Библиотека OSCAT Basic

Документация на русском

Версия библиотеки: 3.33

Версия перевода: 1.0 (01.12.2017)

Оглавление

0. Комментарий переводчиков

Библиотека **OSCAT** хорошо знакома значительному количеству специалистов в области программирования ПЛК. К числу ее несомненных плюсов относится большой набор блоков, открытые исходные коды и аппаратная независимость. В то же время сопроводительная документация для библиотеки доступна только на немецком и английском языках. Разумеется, большинство специалистов владеют английским на достаточном уровне, чтобы читать техническую документацию – но даже при этом требуется дополнительное время, чтобы вникнуть в принцип работы того или иного блока (многие ли смогут перевести слово *abruptly* без обращения к интернету?). Вполне очевидно, что можно воспользоваться онлайн-переводчиком – и в определенном количестве случаев получить бессвязный набор слов. Перечисленные причины сделали задачу перевода документации на **OSCAT** достаточно актуальной.

Более того, попытки решить эту задачу предпринимались и раньше. Например, доменное имя http://oscat.ru/ зарегистрировано (*в 2009 г.*) и содержит единственную страницу, на которой упомянуто, что «*ведется ее (**документации библиотеки**) перевод на английский и русский языки*». Перевод на английский, как известно, был успешно завершен – силами самого сообщества **OSCAT**. Также стоить отметить ресурс http://www.wago.su/oscat, на котором выложен перевод описания нескольких десятков модулей. Можно упомянуть статью, опубликованную на сайте компании ПК Пролог (дистрибьютор **CODESYS** в России), посвященную строковым функциям библиотеки. Наконец, значительная часть функционала **OSCAT** была включена в состав SCADA-системы MasterSCADA 4D – и, соответственно, получила описание в справке на систему.

Как можно заметить, ни в одном из упомянутых случаев задача не была решена полностью. «*Что же*», – сказали мы себе, – «*Show must go on*» – и взялись за работу. Мы не задавались целью дословно перевести англоязычную документацию – она, как уже упоминалось, является дословным переводом с немецкого и соблюдать каждую ее букву было бы не самым разумным решением. Чего нам хотелось – создать предельно понятное описание для всего доступного в библиотеке функционала. Насколько мы справились с этой задачей – решать вам.

Мы приветствуем распространение перевода при соблюдении трех простых условий:

- на некоммерческой основе;
- при отсутствии каких бы то ни было модификаций файла перевода;
- с информированием в случае выкладывания файла (или его фрагментов) на каких-либо ресурсах, включения его в состав ПО и т.д. (адрес электронной почты приведен ниже).

Если вы нашли в переводе ошибку или неточность, то сообщите, пожалуйста, об этом нам, написав на OscatLibRu@gmail.com

Переводчики: Евгений Кислов, Екатерина Чибисова, Войцех Бжезинский

1. Правовые вопросы

1.1. Отказ от ответственности

Программные модули (функции и функциональные блоки), включенные в библиотеку **OSCAT**, предоставляются для использования в качестве шаблонов и примеров для разработки прикладного программного обеспечения ПЛК согласно стандарту МЭК 61131-3. Разработчики библиотеки не принимают на себя никаких обязательств по поводу работоспособности программных модулей. Поскольку библиотека распространяется безвозмездно, в допустимых законодательством пределах отсутствуют какие-либо гарантийные обязательства. Если иное не указано в письменной форме, владельцы авторских прав и/или третьи лица распространяют модули «как есть» («as is)», без каких-либо гарантий, явных или подразумеваемых, в том числе по удобству использования в конкретных ситуациях. Вся полнота риска и ответственность за соответствие критериям качества, функциональности и точности лежит на пользователе. В случае, если библиотека или ее отдельные фрагменты содержат ошибки, затраты на исправление и/или изменение программных модулей лежат на пользователе. Если библиотека или ее отдельные фрагменты используются для создания программного обеспечения или применяются при разработке проектов, то пользователь принимает на себя ответственность за его/их производительность, качество и корректное функционирование. Ответственность для сообщества разработчиков **OSCAT** в явном виде исключается.

Пользователи **OSCAT** должны принимать меры по тестированию и контролю качества приложений, чтобы исключить ущерб от потенциальных ошибок библиотеки. Настоящее лицензионное соглашение и отказ от ответственности имеют одинаковую юридическую силу для библиотеки и документации на нее, даже если это не указывается в явном виде.

1.2. Лицензионное соглашение

Библиотека **OSCAT** распространяется на безвозмездной основе и может быть использована в личных или служебных целях. Распространение библиотеки приветствуется при условии сохранения ее бесплатности и публикации ссылки на веб-страницу www.oscat.de. Если библиотека публикуется для загрузки или распространяется на носителях информации, то должно быть в явном виде указано авторство сообщества **OSCAT** и приведена ссылка на веб-страницу www.oscat.de.

1.3. Зарегистрированные товарные знаки

Все товарные знаки, упомянутые в данной документации, приводятся без ссылок на их регистрацию и владельцев. Существование таких прав не может быть исключено. Все упомянутые товарные знаки являются собственностью их владельцев. По этой причине не допускается коммерческое использование документации или ее фрагментов.

1.4. Использование по назначению

Программные модули (функции и функциональные блоки), включенные в библиотеку **OSCAT**, были разработаны исключительно для использования специалистами, обладающими соответствующей квалификацией в области программирования ПЛК. Пользователи несут ответственность за соблюдение всех действующих стандартов и нормативных актов, которые связаны с использованием программных модулей. Библиотека и ее описание не соответствуют никаким стандартам и нормативным актам.

1.5. Остальное

Все юридически обязательные правовые нормы приведены в главе 1 данного документа. Выдвижение юридических претензий на основании содержимого данного документа полностью исключено, кроме случаев, упомянутых в главе 1.

2. Введение

2.1. Задачи

OSCAT является акронимом названия **O**pen **S**ource **C**ommunity for **A**utomation **T**echnology («Сообщество открытого ПО для систем автоматизации»).

Сообщество **OSCAT** занимается созданием библиотеки с открытым исходным кодом в соответствии со стандартом МЭК 61131-3, которая не имеет привязок к конкретному оборудованию и, соответственно, может использоваться на любых ПЛК, поддерживающих данный стандарт. В настоящее время большинство производителей ПЛК бесплатно предоставляют свои проприетарные библиотеки, но этот подход имеет ряд существенных недостатков:

1. Исходный код большинства проприетарных библиотек закрыт для пользователей, что делает внесение изменений и исправление потенциально возможных ошибок сложным и, зачастую, даже невозможным;

2. Использование проприетарных библиотек при разработке программ на графических языках (LD, FBD, CFC) может быть сложным, неэффективным и приводить к ошибкам, поскольку существующие функции не обязательно соответствуют потребностям пользователей. Исходный код библиотек при этом закрыт для редактирования;

3. Использование проприетарных библиотек затрудняет перенос приложений с одних ПЛК на другие (особенно в случае ПЛК различных производителей), тем самым сводя на нет преимущества стандарта МЭК 61131-3. Замена проприетарных библиотек одного производителя библиотеками другого в большинстве случаев невозможна (или как минимум требует затрат на адаптацию программы) из-за специфических различий;

4. Понимание принципов работы сложных программных модулей без наличия их исходного кода может быть крайне затруднено, что снижает эффективность программ и приводит к различного рода ошибкам.

Сообщество **OSCAT** с помощью разработки одноименной библиотеки планирует создать функциональный и понятный стандарт программирования ПЛК, исходные коды которого открыты и тщательно протестированы в различных приложениях. Использование библиотеки в различных проектах поможет исправлять ошибки и улучшать функционал, что является очень практичным. Сообщество OSCAT позиционирует библиотеку как шаблон для разработки приложений, а не законченный продукт. Пользователь несет полную ответственность за тестирование своих приложений с целью верификации заданным критериям точности, качества и функциональности. По этой причине мы должны сослаться на лицензионное соглашение и отказ от ответственности, опубликованные в главе 1 данного документа.

2.2. Соглашения

1. Прямой доступ к памяти

Названия модулей, изменяющих значения своих входных переменных с помощью указателей (например, _ARRAY_SORT), начинаются с символа нижнего подчеркивания («_»). Функция **_ARRAY_SORT** сортирует массив с помощью прямого доступа к памяти, что является очень эффективным решением, поскольку позволяет обойтись без промежуточных массивов (и, соответственно, сэкономить память), а также уменьшить время сортировки (т.к. не потребуется копировать данные из входного массива в промежуточные). Тем не менее, данные модули должны использоваться только опытными специалистами, поскольку их некорректный вызов может привести к серьезным ошибкам (вплоть до прекращения работы приложения). В частности, при вызове таких функций необходимо делать проверку на то, что значения указателей не являются нулевыми или неопределенными.

2. Названия модулей

Модули, названия которых начинаются с префикса **FT_**, подразумевают использование на некотором отрезке времени (например, ФБ FT_AVG вычисляет среднее значение переменной по заданному количеству срезов, и его однократный вызов не приведет к желаемому результату).

Модули, названия которых начинаются с префикса **F_**, подразумевают однократный вызов (например, функция F_LIN возвращает значение линейной функции с заданными коэффициентами, и ее вызов имеет смысл только в том случае, если значение одной из входных переменных изменилось).

Примечание – данное соглашение распространяется на модули из главы 9 и главы 23.

3. Параметры модулей

Чтобы облегчить процесс разработки и упростить работу со сложными функциями, многие модули библиотеки **OSCAT** имеют параметры. Параметры представляют переменные класса **VAR_INPUT CONSTANT**, которые задаются при вызове модуля и не могут изменить свои значения в процессе его работы. На языке **CFC** для открытия диалога редактирования параметров необходимо дважды нажать **ЛКМ** на вкладку **Параметры**, расположенную внутри элемента. Можно привязать к параметру переменную или задать ему конкретное значение.

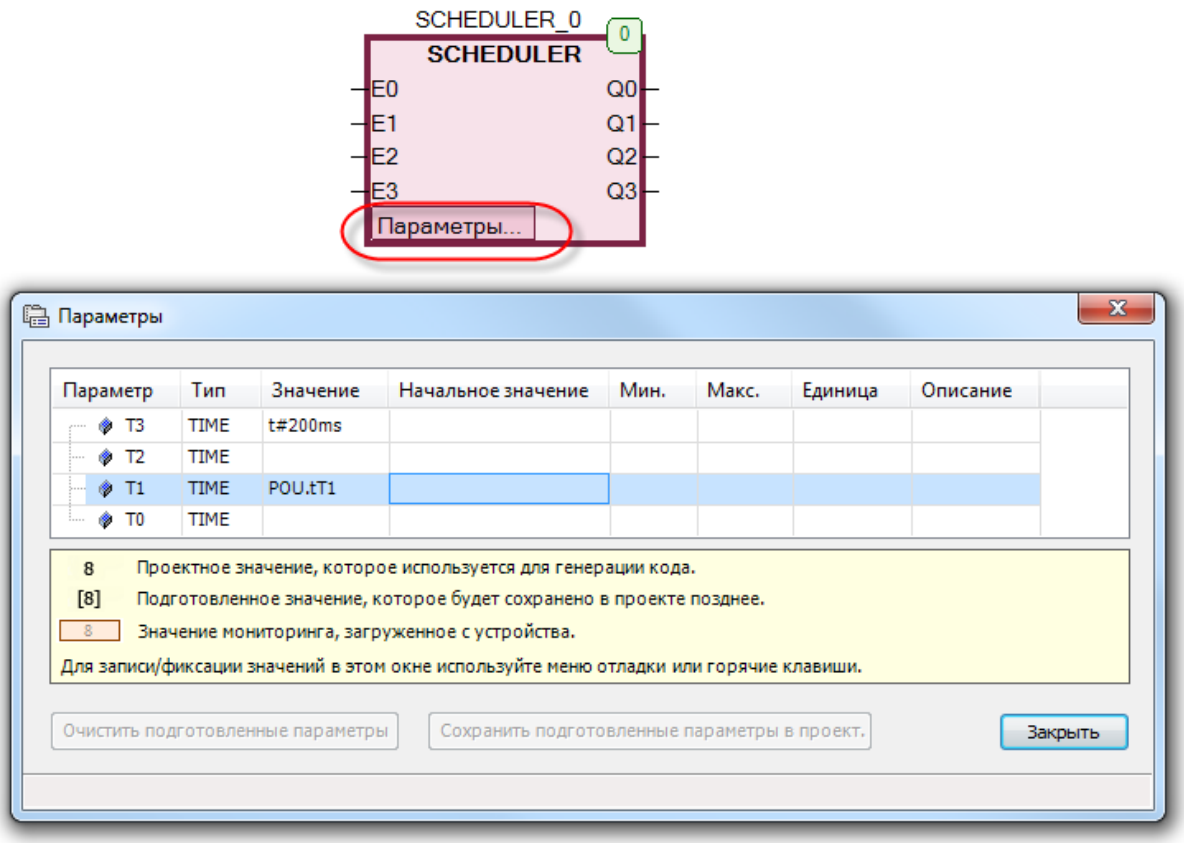


Рис. 4.1. Настройка параметров модуля на языке **CFC**

На языке **ST** значения параметров задаются непосредственно при вызове модуля:

```
3   SCHEDULER_0(T1:=tT1, T3:=T#200MS);
```

Рис. 4.2. Вызов модуля с параметрами на языке **ST**

4. Отчеты о событиях и ошибках (**ESR**)

Модули, выполняющие сложные функции, обычно имеют выход **Error**, который используется для сигнализации о возникших ошибках. Значение **0** соответствует нормальной работе модуля. При возникновении ошибок выход принимает значение из диапазона **1..99**; это значение соответствует коду ошибки. Расшифровка кодов приводится в описании соответствующего модуля.

Выход **Status** выполняет аналогичную функцию, но содержит коды не ошибок, а состояний модуля (например, включен/остановлен/выключен). Диапазон кодов состояний: **100…199**. Диапазон **200…255** зарезервирован для отладочных сообщений.

ESR-модули позволяют собирать информацию об ошибках и состояниях с метками времени, которая при необходимости может быть сохранена в архив или передана на верхний уровень АСУ (реализация архивации и передачи должны быть выполнена пользователем). Это позволяет с помощью стандартных возможностей библиотеки вести аварийные и оперативные журналы, не используя дополнительный функционал системы. Описание ESR-модулей приведено в главе 4.

2.3. Средства тестирования

Библиотека **OSCAT** изначально создана в среде **CoDeSys 2.x** и портирована в другие среды разработки. Библиотека протестирована на следующем оборудовании:

ПЛК	Среда разработки
Beckhof BX 9000	TwinCAT PLC Control Version 2.10.0
Beckhof CX 9001-1001	TwinCAT PLC Control Version 2.10.0
Wago 750-841	CoDeSys 2.3.9.31
Möller EC4P222	
CoDeSys Simulation on I386	
CODESYS Control Win V3	CODESYS 3.4
-	S7 and STEP 7: Библиотека поддерживается и тестируется на STEP7 начиная с версии 1.5.
-	PCWORX / MULTIPROG: Библиотека поддерживается и тестируется на MULTIPROG начиная с версии 2.6.
Bosch Rexroth IndraLogic XLC L25/L45/L65	Indraworks 12VRS
Bosch Rexroth IndraMotion MLC L25/L45/L65	
Bosch Rexroth IndraMotion MTX L45/L65/L85	

Примечание переводчиков – в процессе перевода библиотека тестировалась на следующих ПЛК:

ПЛК	Среда разработки
CODESYS Control Win V3	CODESYS 3.5 SP7 Patch4
Berghof DC2007	CODESYS 3.5 SP7 Patch4
Овен СП207 [M04] / [M05]	CODESYS 3.5 SP5 Patch5

2.4. Глобальные переменные

Разработчики **OSCAT** старались избежать использования глобальных переменных для упрощения интеграции библиотеки в различные среды разработки. Реализация пользовательских алгоритмов и обмена данными между устройствами может быть осуществлена без применения глобальных переменных. Переменные параметров функций и ФБ являются локальными, что обеспечивает модульность и переносимость программы.

Для прозрачности использования физических и математических констант (таких, как скорость света, число Пи и т.д.) было решено сделать их глобальными переменными. Ниже приведен их список.

Название переменной	Тип	Описание
Глобальные переменные		
MATH	CONSTANTS_MATH	Структура CONSTANTS_MATH определяет математические константы.
PHYS	CONSTANTS_PHYS	Структура CONSTANTS_PHYS определяет физические константы.
LANGUAGE	CONSTANTS_LANGUAGE	Структура CONSTANTS_LANGUAGE определяет языковые константы.
SETUP	CONSTANTS_SETUP	Структура CONSTANTS_SETUP определяет вспомогательные константы для работы со строками и временем.
LOCATION	CONSTANTS_LOCATION	Структура CONSTANTS_LOCATION определяет языки регионов.
Глобальные константы		
String_Length	INT	Используется в функциях работы со строками для определения их размеров. Значение по умолчанию – 250. Рекомендуется использовать эту константу при объявлении всех строковых переменных приложения.
List_Length	INT	Используется для определения размеров списков. Значение по умолчанию – 250.

2.5. Версионность

Данное руководство постоянно актуализируется. Рекомендуется загрузить последнюю версию руководства с сайта www.oscat.de. Помимо руководства, на сайте доступен документ с историей версий всех модулей библиотеки (даты добавления, внесенные изменения и т.д.).

Примечание переводчиков – последняя на данный момент (*начало 2018 г.)* версия библиотеки OSCAT и ее документации были выпущены зимой 2012 года (версия **3.33**). На основе этой версии и составлено данное описание.

2.6. Техподдержка

Техподдержка осуществляется пользователями форума www.oscat.de. При этом техподдержка в каждом конкретном случае не гарантируется, даже при наличии ошибок в библиотеке или отдельных модулях. Техподдержка осуществляется силами пользователей библиотеки и на добровольной основе. Обновление библиотеки и документации обычно происходит раз в месяц, актуальные версии выкладываются на сайте www.oscat.de. Никакие претензии по техническому обслуживанию, устранению ошибок и сопровождению не принимаются. Пожалуйста, не отправляйте запросы по технической поддержке на электронную почту. Ответ последует значительно быстрее, если они будут размещены на форуме.

2.7. Структура библиотек OSCAT

Изначально библиотека OSCAT представляла собой единую сущность. По мере развития число модулей стало настолько большим, что разработчики приняли решение разделить их на три отдельные библиотеки:

1. **OSCAT Basic** – библиотека базовых, низкоуровневых модулей. Именно ей посвящен данный документ.

2. **OSCAT Building** – библиотека модулей для автоматизации зданий (например, модули управления жалюзи).

3. **OSCAT Network** – библиотека модулей коммуникации и архивации.

Библиотеки OSCAT Building и OSCAT Network основаны на OSCAT Basic и используют ее модули.

3. Типы данных

3.1. CALENDAR

Структура **CALENDAR** содержит подробную информацию о текущей дате и времени. Она используется при работе с ФБ CALENDAR_CALC.

Переменная структуры	Тип	Описание
UTC	DT	Дата и время по UTC.
LDT	DT	Местные дата и время.
LDATE	DATE	Дата по LDT.
LTOD	TOD	Время по LDT.
YEAR	INT	Год по LDT.
MONTH	INT	Месяц по LDT.
DAY	INT	День по LDT.
WEEKDAY	INT	День недели по LDT (**1** – понедельник, **7** – воскресенье).
OFFSET	INT	Смещение LDT относительно UTC в минутах.
DST_EN	BOOL	Управление режимом летнего времени (**TRUE** – включить, **FALSE** – выключить).
DST_ON	BOOL	Флаг режима летнего времени (**TRUE** – включен, **FALSE** – отключен).
NAME	STRING(5)	Название часового пояса.
LANGUAGE	INT	Язык региона. Используется для определения названий месяцев и дней.
LONGITUDE	REAL	Долгота.
LATITUDE	REAL	Широта.
SUN_RISE	TOD	Время восхода солнца по LDT.
SUN_SET	TOD	Время заката солнца по LDT.
SUN_MIDDAY	TOD	Время солнечного полудня по UTC.
SUN_HEIGHT	REAL	Склонение солнца над горизонтом в градусах.
SUN_HOR	REAL	Азимут.
SUN_VER	REAL	Высота солнца над горизонтом в градусах с учетом рефракции.
NIGHT	BOOL	Флаг «Сейчас ночь».
HOLIDAY	BOOL	Флаг «Сегодня праздник».
HOLY_NAME	STRING(30)	Название праздника.
WORK_WEEK	INT	Номер недели в году (начиная с **1**).

3.2. COMPLEX

Структура **COMPLEX** используется для представления комплексных чисел. Модули работы с комплексными числами описаны в главе 7.

Переменная структуры	Тип	Описание
re	REAL	Действительная часть комплексного числа.
im	REAL	Мнимая часть комплексного числа.

3.3. CONSTANTS_LANGUAGE

Структура **CONSTANTS_LANGUAGE** используется для задания параметров языков проекта. По умолчанию структура определяет три языка – английский, немецкий и французский. Для работы со структурой используется глобальная переменная **LANGUAGE**.

Переменная структуры	Тип	Описание
DEFAULT	INT	Язык по умолчанию. Языковые модули библиотеки в качестве входа принимают код языка. Значение **0** соответствует языку по умолчанию, значения от **1** до **LMAX** – одному из языков проекта.
LMAX	INT	Количество языков проекта. Значение по умолчанию – **3** (см. размерность следующих массивов).
WEEKDAYS	ARRAY [1..**3**, 1..7] OF STRING(10)	Полные названия дней недели (**Monday**) для всех языков проекта.
WEEKDAYS2	ARRAY [1..**3**, 1..7] OF STRING(2)	Сокращенные названия дней недели (**Mo**) для всех языков проекта.
MONTHS	ARRAY [1..**3**, 1..12] OF STRING(10)	Полные названия месяцев (**January**) для всех языков проекта.
MONTHS3	ARRAY [1..**3**, 1..12] OF STRING(3)	Сокращенные названия месяцев (**Jan**) для всех языков проекта.
DIRS	ARRAY [1..**3**, 0..15] OF STRING(3)	Обозначения румбов для всех языков проекта.

3.4. CONSTANTS_LOCATION

Структура **CONSTANTS_LOCATION** используется для задания языков регионов. По умолчанию структура определяет пять регионов, в каждом из которых используется немецкий язык. Для работы со структурой применяется глобальная переменная **LOCATION**.

Переменная структуры	Тип	Описание
DEFAULT	INT	Регион по умолчанию.
LMAX	INT	Количество языков проекта. Значение по умолчанию – **5**.
LANGUAGE	ARRAY [1..**5**] OF INT	Коды языков для регионов проекта.

3.5. CONSTANTS_MATH

Структура **CONSTANTS_MATH** определяет математические константы. Для работы со структурой применяется глобальная переменная **MATH**.

Переменная структуры	Тип	Точность (количество знаков после запятой)	Описание
PI	REAL	35	Число Пи
PI2	REAL	35	Число Пи · 2
PI4	REAL	35	Число Пи · 4
PI05	REAL	13	Число Пи / 2
PI025	REAL	15	Число Пи / 4
PI_INV	REAL	15	1 / число Пи
E	REAL	35	Число Эйлера
E_INV	REAL	15	1 / число Эйлера
SQ2	REAL	13	Квадратный корень из 2.
FACTS	ARRAY [0..12] OF DINT	-	Факториалы чисел от 0 до 12.

3.6. CONSTANTS_PHYS

Структура **CONSTANTS_PHYS** определяет физические константы. Для работы со структурой применяется глобальная переменная **PHYS**.

Переменная структуры	Тип	Точность (количество знаков после запятой)	Описание
C	REAL	-	Скорость света, м/с.
E	REAL	8	Элементарный электрический заряд, Кл.
G	REAL	5	Ускорение свободного падения, м/с^2.
T0	REAL	2	Абсолютный нуль температуры, °C.
RU	REAL	6	Универсальная газовая постоянная, Дж/(моль· К).
PN	REAL	-	Нормальное атмосферное давление, Па.

3.7. CONSTANTS_SETUP

Структура **CONSTANTS_SETUP** определяет вспомогательные константы для работы со строками и временем. Для работы со структурой применяется глобальная переменная **SETUP**.

Переменная структуры	Тип	Описание
EXTENDED_ASCII	BOOL	Флаг использования верхней половины таблицы ASCII.
CHARNAME	ARRAY [1..4] OF STRING(253)	Содержит символы Юникода.
MTH_OFS	ARRAY [1..12] OF INT	Содержит номера последних дней месяцев календарного года (0, 31, 59, 70 и т.д.).
DECADES	ARRAY [0..8] OF REAL	Массив результатов возведения в степень числа 10.

3.8. ESR_DATA

Структура **ESR_DATA** используется ESR-модулями для сбора информации об ошибках и изменении состояний модулей, поддерживающих данный функционал.

Переменная структуры	Тип	Описание
TYP	BYTE	Тип события.
ADRESS	STRING(10)	Имя контролируемого параметра.
DS	DT	Дата и время события.
TS	TIME	Относительное время события (в мс после старта проекта).
DATA	ARRAY [0..7] OF BYTE	Информация о событии.

3.9. FRACTION

Структура **FRACTION** используется для представления дробей.

Переменная структуры	Тип	Описание
NUMERATOR	INT	Числитель дроби.
DENUMERATOR	INT	Знаменатель дроби.

3.10. HOLIDAY_DATA

Структура **HOLIDAY_DATA** содержит информацию о праздниках. Крайне сложно дать емкое описание переменных структуры – гораздо проще объяснить принцип их использования на примерах.

Переменная структуры	Тип	Описание
NAME	STRING(30)	Название праздника.
DAY	SINT	См. ниже.
MONTH	SINT	См. ниже.
USE	SINT	См. ниже.

С календарной точки зрения праздники можно разделить на две группы:

1. праздники с фиксированной датой;

2. переходящие праздники.

Для праздников с фиксированной датой переменные **DAY** и **MONTH** определяют дату проведения, а переменная **USE** характеризует, следует ли учитывать праздник модулям библиотеки (**1** – следует, **0** – не следует).

Пример объявления:

ALL_SAINTS: HOLIDAY_DATA := (NAME:='All Saints Day' , DAY:=1 , MONTH:=11, USE:=1);

Для переходящих праздников возможно два варианта определения:

1. Относительно заданного опорного праздника. Опорным считается праздник с **DAY**=**0** и **MONTH**=**0**:

*EASTER_SUNDAY: HOLIDAY_DATA := (NAME:='Пасха' , DAY:=**0** , MONTH:=**0**, USE:=1);*

День Святой Троицы является переходящим праздником – он отмечается спустя **50 дней** после Пасхи (при этом Пасха считается **1**-м днем). Тогда объявить его можно следующим образом:

*SAINT_TRINITY_DAY: HOLIDAY_DATA := (NAME:='Святая троица' , DAY:=**49** , MONTH:=**0**, USE:=1);*

В переменной **DAY** указывается смещение (положительное или отрицательное) относительно опорного праздника, переменная **MONTH** задается **нулем**.

2. Относительно заданной даты. Например, немецкий праздник День покаяния и молитвы празднуется в последнюю среду перед началом Адвента. Соответственно, если дата Адвента известна (в 2016-м году это **27-ое ноября**), то праздник можно определить следующим образом:

*BUSS_UND_BETAG: HOLIDAY_DATA := (NAME:='Buss und Betag' , DAY:=**27** , MONTH:=**11**, USE:=**-3**);*

Переменная **USE** в данном случае определяет день недели перед заданной датой (где **-1** соответствует понедельнику … -7 соответствует воскресенью).

В приведенных выше примерах праздники объявляются по одному, но в большинстве случаев удобнее представить их сразу в виде массива:

HOLIDAYS_RU: ARRAY [0..29] OF HOLIDAY_DATA;

3.11. REAL2

Структура **REAL2** используется для представления чисел двойной точности на системах без поддержки переменных типа **LREAL** – но только при работе с определенными функциями и рядом ограничений. Модули работы с числами двойной точности описаны в главе 8.

Пример представления: 11.2222222233 = RX + R1, где:

RX=11.2222222,

R1=0.0000000033.

Переменная структуры	Тип	Описание
R1	REAL	Разность между числом двойной точности и R1.
RX	REAL	Число двойной точности, округленное до REAL.

3.12. SDT

Структура **SDT** используется для определения разрядов даты и времени.

Переменная структуры	Тип	Описание
YEAR	INT	Год.
MONTH	INT	Месяц.
DAY	INT	День.
WEEKDAY	INT	День недели (**1** – понедельник, **7** – воскресенье).
HOUR	INT	Часы.
MINUTE	INT	Минуты.
SECOND	INT	Секунды.
MS	INT	Миллисекунды.

3.13. TIMER_EVENT

Структура **TIMER_EVENT** не используется ни одним из модулей библиотеки **OSCAT Basic**, поэтому ее описание здесь не приводится. Оно будет приведено в описании библиотеки **OSCAT Building**, поскольку используется ее модулем **TIMER_P4**.

3.14. VECTOR_3

Структура **VECTOR_3** определяет радиус-вектор в трехмерном пространстве. Модули работы с векторами описаны в главе 11.

Переменная структуры	Тип	Описание
X	REAL	Координата X.
Y	REAL	Координата Y.
Z	REAL	Координата Z.

4. Специальные функции

4.1. ESR_COLLECT

Тип модуля: ФБ	**Переменная**	**Тип**	**Описание**
Входы	ESR_0…ESR_7	ARRAY [0…3] OF ESR_DATA	Массив входных данных.
	rst	BOOL	Сигнал удаления данных.
Выходы	ESR_OUT	ARRAY [0…31] OF ESR_DATA	Массив выходных данных.
Входы-выходы	pos	INT	Номер последнего записанного элемента выходного массива.
Используемые модули	INC1		

Обратите внимание, что в текущей версии библиотеки ФБ работает **некорректно** (пруф). Для корректной работы необходимо отредактировать код ФБ следующим образом:

```
IF rst OR cnt < 0 THEN
    pos := -1;
    cnt:=0;                          // bug-fix
ELSE
    FOR cnt := 0 TO max_in DO
    IF esr_0[cnt].typ > 0 THEN pos := INC1(pos, max_out); esr_out[pos] := esr_0[cnt]; END_I
    IF esr_1[cnt].typ > 0 THEN pos := INC1(pos, max_out); esr_out[pos] := esr_1[cnt]; END_I
    IF esr_2[cnt].typ > 0 THEN pos := INC1(pos, max_out); esr_out[pos] := esr_2[cnt]; END_I
    IF esr_3[cnt].typ > 0 THEN pos := INC1(pos, max_out); esr_out[pos] := esr_3[cnt]; END_I
    IF esr_4[cnt].typ > 0 THEN pos := INC1(pos, max_out); esr_out[pos] := esr_4[cnt]; END_I
    IF esr_5[cnt].typ > 0 THEN pos := INC1(pos, max_out); esr_out[pos] := esr_5[cnt]; END_I
    IF esr_6[cnt].typ > 0 THEN pos := INC1(pos, max_out); esr_out[pos] := esr_6[cnt]; END_I
    IF esr_7[cnt].typ > 0 THEN pos := INC1(pos, max_out); esr_out[pos] := esr_7[cnt]; END_I
   END_FOR;
END_IF;
```

Рис. 4.1. Исправление исходного кода ФБ **ESR_COLLECT** для обеспечения корректной работы

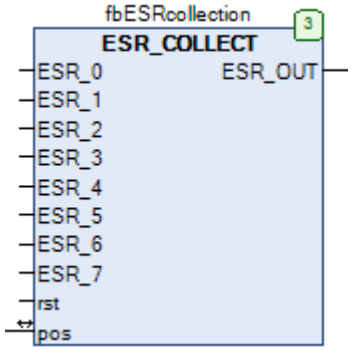
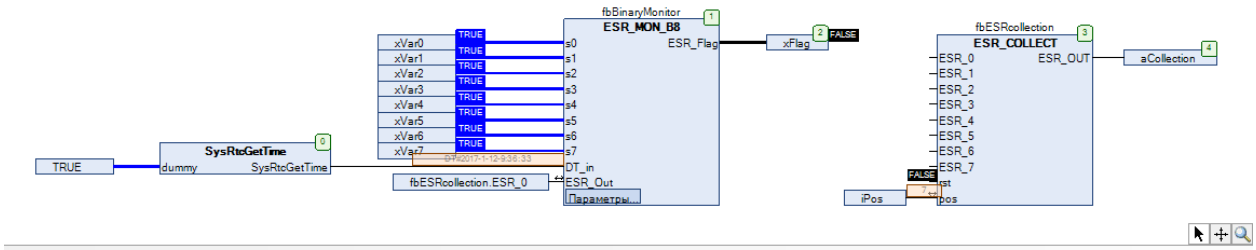


Рис. 4.2. Внешний вид ФБ **ESR_COLLECT** на языке CFC

Функциональный блок **ESR_COLLECT** используется для сбора данных других ESR-модулей и сохранения их в массив. Входы **ESR_0...ESR_7** используются как значения для входов-выходов **ESR_Out** других ESR-модулей (см. рис. 4.3). При обновлении информации ESR-модулей, данные заносятся в массив **ESR_COLLECT.ESR_OUT**, при этом значение входа-выхода **pos** инкрементируется. Массив **ESR_COLLECT.ESR_OUT** может хранить до 32-х элементов типа ESR_DATA. При превышении этого количества начинается перезапись элементов (начиная с нулевого). По переднему фронту на входе **rst** происходит удаление всех данных модуля, после чего **pos** принимает значение **-1**.



Watch 1

Выражение	Приложение	Тип	Значение	Подготовленн...	Точка трассировки
PLC_PRG.aCollection	Device.Application	ARRAY [0..31] O...			Циклический мониторинг
aCollection[0]		ESR_DATA			Циклический мониторинг
TYP		BYTE	11		Циклический мониторинг
ADRESS		STRING(10)	'Сигнал 0'		Циклический мониторинг
DS		DATE_AND_TIME	DT#2017-1-12-9:36:...		Циклический мониторинг
TS		TIME	T#1d4h44m14s643ms		Циклический мониторинг
DATA		ARRAY [0..7] OF...			Циклический мониторинг
aCollection[1]		ESR_DATA			Циклический мониторинг
aCollection[2]		ESR_DATA			Циклический мониторинг
aCollection[3]		ESR_DATA			Циклический мониторинг
aCollection[4]		ESR_DATA			Циклический мониторинг
aCollection[5]		ESR_DATA			Циклический мониторинг
aCollection[6]		ESR_DATA			Циклический мониторинг
aCollection[7]		ESR_DATA			Циклический мониторинг
TYP		BYTE	11		Циклический мониторинг
ADRESS		STRING(10)	'Сигнал 7'		Циклический мониторинг
DS		DATE_AND_TIME	DT#2017-1-12-9:36:...		Циклический мониторинг

Рис. 4.3. Использование ФБ **ESR_COLLECT** на языке CFC для сохранения данных ФБ **ESR_MON_B8**

4.2. ESR_MON_B8

Тип модуля: ФБ	**Переменная**	**Тип**	**Описание**
Входы	s0…s7	BOOL	Сигналы для мониторинга.
	DT_in	DT	Текущие дата и время.
Выходы	ESR_Flag	BOOL	Признак обновления данных.
Входы-выходы	ESR_out	ARRAY [0…3] OF ESR_DATA	Информация мониторинга.
Параметры	a0…a7	STRING(10)	Названия сигналов.
Используемые модули	T_PLC_MS		

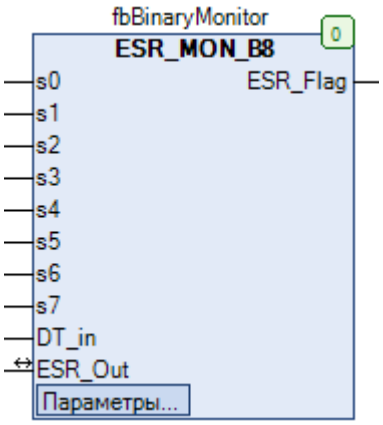


Рис. 4.4. Внешний вид ФБ **ESR_MON_B8** на языке CFC

Функциональный блок **ESR_MON_B8** применяется для мониторинга от одной до восьми переменных типа BOOL (**s0...s7**). На вход **DT_in** подается значение системного времени в формате DT (считанного с помощью одной из доступных пользователю библиотек). При изменении значения переменных **s0…s7** данные мониторинга сохраняются в массив **ESR_Out**, элементы которого являются экземплярами структуры типа ESR_DATA. Массив содержит 4 элемента и, соответственно, позволяет хранить данные мониторинга для четырех одновременно изменившихся переменных. Иными словами, при одновременном (произошедшем в одном цикле ПЛК) изменении всех восьми переменных будет сохранена информация только о четырех (приоритет отдается сигналам с меньшим номером). Для ведения архива мониторинга можно использовать ФБ ESR_Collect. Выход **ESR_flag** принимает значение **TRUE** на один цикл при обновлении данных мониторинга.

Структура ESR_DATA включает следующую информацию:

Переменная структуры	**Тип**	**Описание**
TYP	BYTE	Тип изменения переменной типа BOOL. **11** – с FALSE на TRUE, 1**0** – с TRUE на FALSE Изменяется только на один цикл, после чего сбрасывается в 0.
ADRESS	STRING(10)	Описание переменной (указывается в параметрах модуля).
DS	DT	Дата и время изменения.
TS	TIME	Относительное время изменения (в мс после старта проекта).
DATA	ARRAY [0..7] OF BYTE	Не используется данным ФБ.

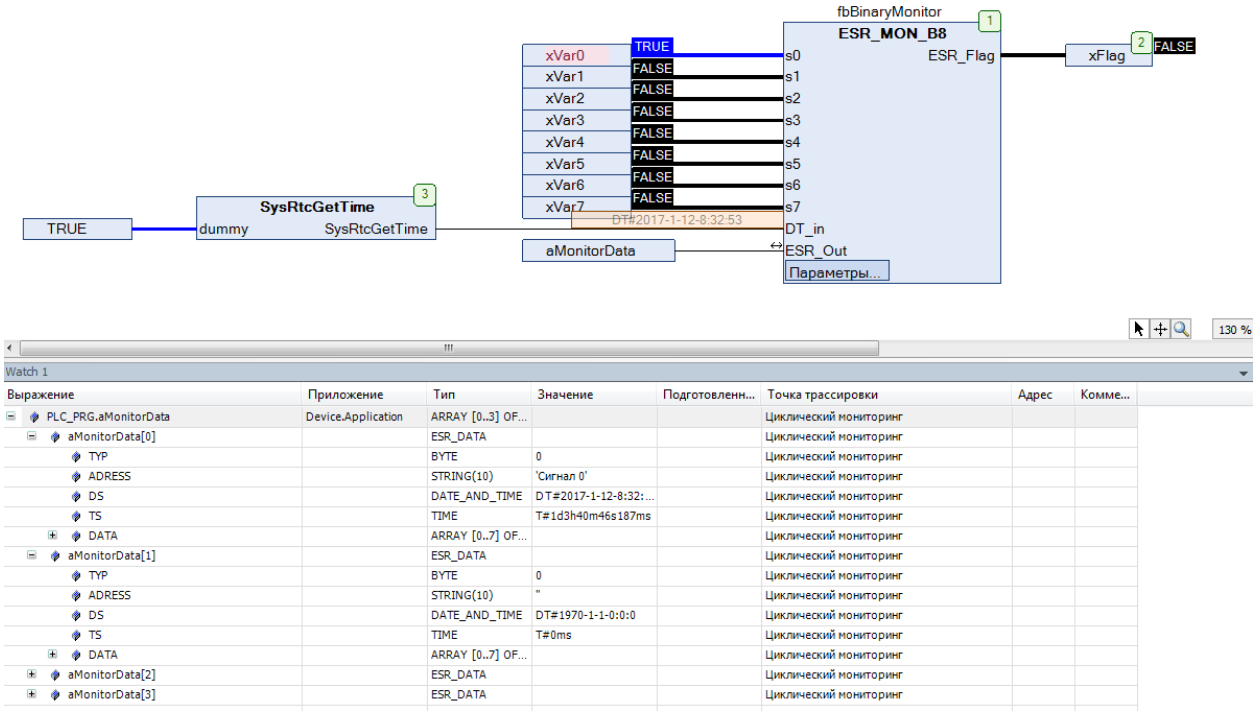


Рис. 4.5. Пример работы с ФБ **ESR_MON_B8** на языке CFC. Изменение дискретного сигнала **xVar0** (‘Сигнал 0’)

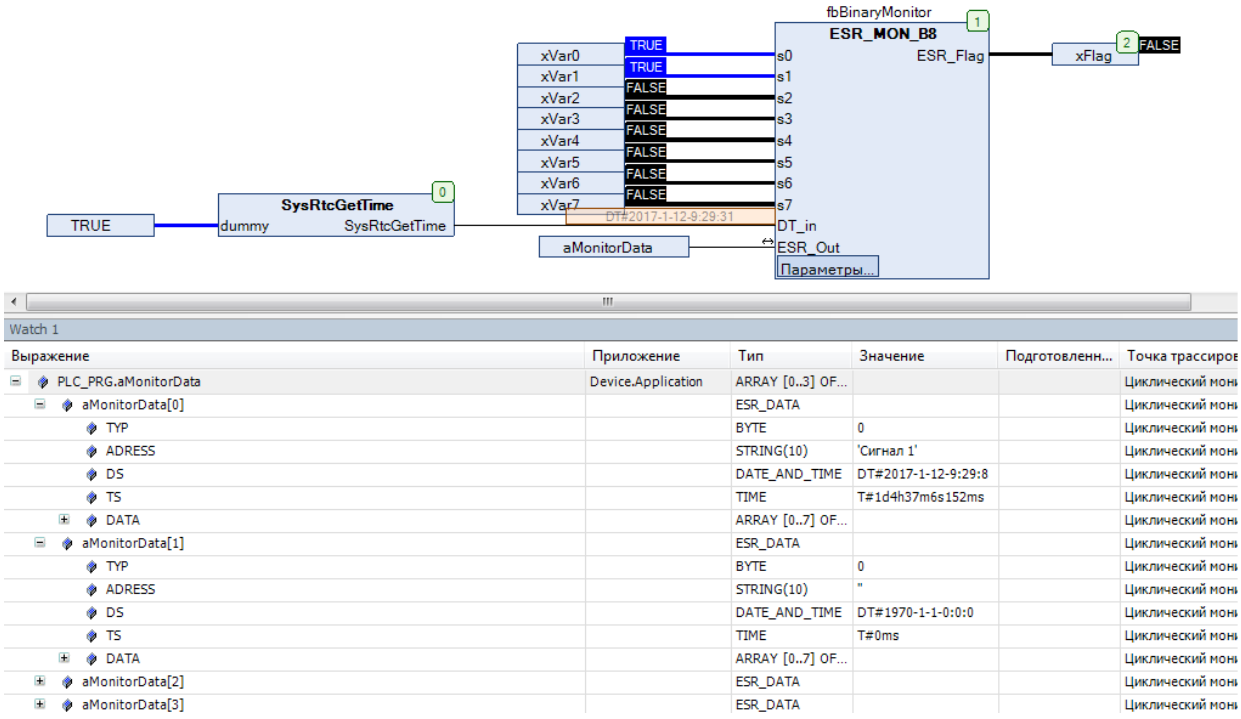
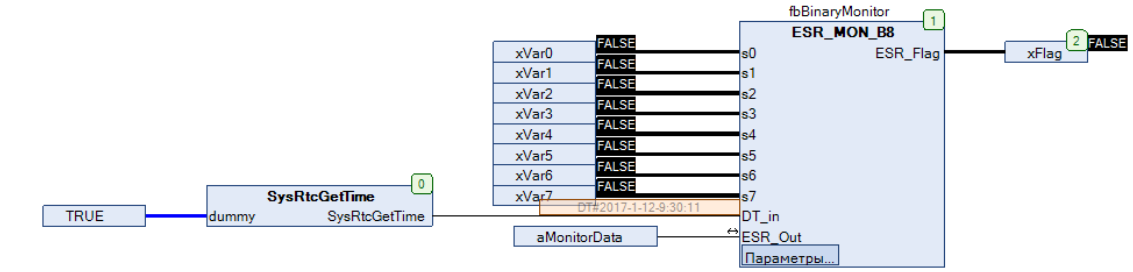


Рис. 4.6. Пример работы с ФБ **ESR_MON_B8 на** языке CFC. Спустя некоторое время изменился сигнал **xVar1** (‘Сигнал 1’)



Watch 1

Выражение	Приложение	Тип	Значение	Подготовленн...	Точка трассировки
PLC_PRG.aMonitorData	Device.Application	ARRAY [0..3] OF...			Циклический мониторинг
aMonitorData[0]		ESR_DATA			Циклический мониторинг
TYP		BYTE	0		Циклический мониторинг
ADRESS		STRING(10)	'Сигнал 0'		Циклический мониторинг
DS		DATE_AND_TIME	DT#2017-1-12-9:30:2		Циклический мониторинг
TS		TIME	T#1d4h38m612ms		Циклический мониторинг
DATA		ARRAY [0..7] OF...			Циклический мониторинг
aMonitorData[1]		ESR_DATA			Циклический мониторинг
TYP		BYTE	0		Циклический мониторинг
ADRESS		STRING(10)	'Сигнал 1'		Циклический мониторинг
DS		DATE_AND_TIME	DT#2017-1-12-9:30:2		Циклический мониторинг
TS		TIME	T#1d4h38m612ms		Циклический мониторинг
DATA		ARRAY [0..7] OF...			Циклический мониторинг
aMonitorData[2]		ESR_DATA			Циклический мониторинг
aMonitorData[3]		ESR_DATA			Циклический мониторинг

Рис. 4.7. Пример работы с ФБ **ESR_MON_B8** на языке CFC. Спустя некоторое время оба сигнала одновременно вернулись к состоянию **FALSE**

4.3. ESR_MON_R4

Тип модуля: ФБ	**Переменная**	**Тип**	**Описание**
Входы	R0…R3	BOOL	Сигналы для мониторинга.
	DT_in	DT	Текущие дата и время.
Выходы	ESR_Flag	BOOL	Признак обновления данных.
Входы-выходы	ESR_out	ARRAY [0…3] OF ESR_DATA	Информация мониторинга.
Параметры	a0…a3	STRING(10)	Названия сигналов.
	s0…s3	REAL	Гистерезис сигналов.
Используемые модули	T_PLC_MS, DIFFER		

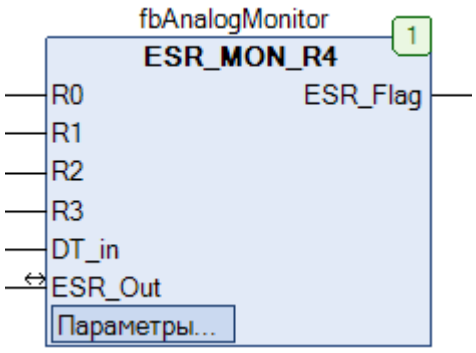
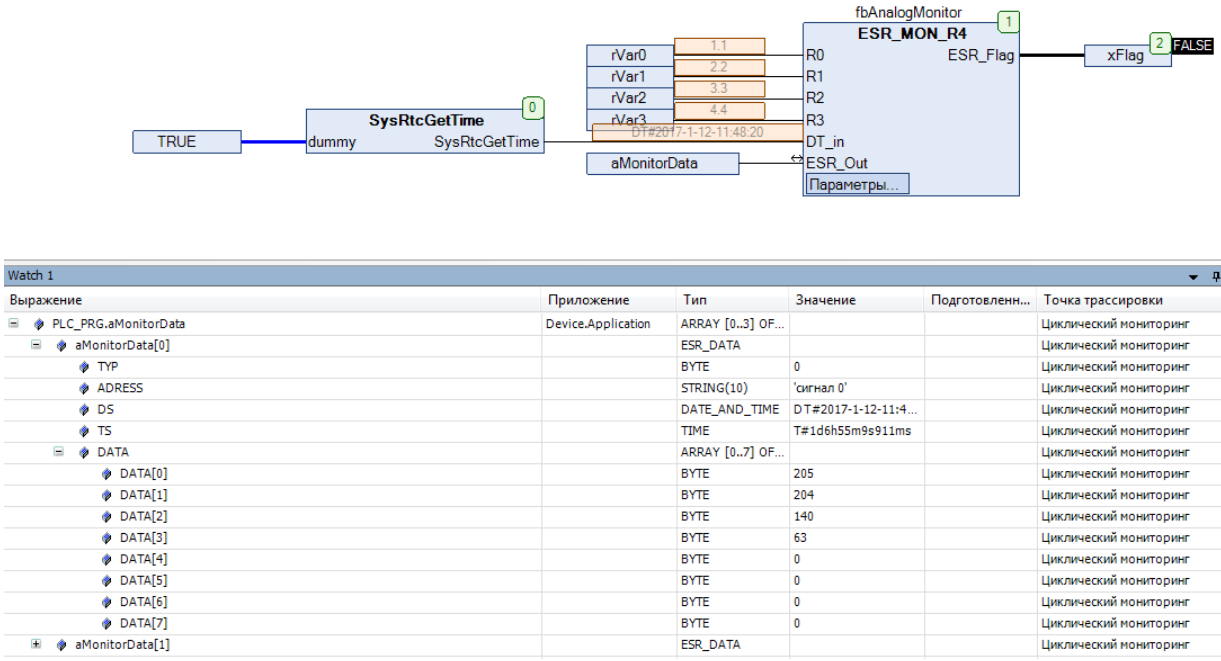


Рис. 4.8. Внешний вид ФБ **ESR_MON_R4** на языке CFC

Функциональный блок **ESR_MON_R4** применяется для мониторинга от одной до четырех переменных типа REAL (**R0...R3**). На вход **DT_in** подается значение системного времени в формате DT (считанного с помощью одной из доступных пользователю библиотек). При изменении значения переменных **R0…R3** на величину **s0…s3** или выше данные мониторинга сохраняются в массив **ESR_Out**, элементы которого являются экземплярами структуры типа ESR_DATA. Массив содержит 4 элемента и, соответственно, позволяет хранить данные о последнем изменении для каждой переменной. Для ведения архива мониторинга можно использовать ФБ ESR_Collect. Выход **ESR_flag** принимает значение **TRUE** на один цикл при обновлении данных мониторинга.

Структура ESR_DATA включает следующую информацию:

Переменная структуры	**Тип**	**Описание**
TYP	BYTE	Принимает значение **20** при изменении переменной. Изменяется только на один цикл, после чего сбрасывается в 0.
ADRESS	STRING(10)	Описание переменной (указывается в параметрах модуля).
DS	DT	Дата и время изменения.
TS	TIME	Относительное время изменения (в мс после старта проекта).
DATA	ARRAY [0..7] OF BYTE	В первых четырех байтах содержится новое значение переменной в формате IEEE 754.



Watch 1

Выражение	Приложение	Тип	Значение	Подготовленн...	Точка трассировки
PLC_PRG.aMonitorData	Device.Application	ARRAY [0..3] OF...			Циклический мониторинг
aMonitorData[0]		ESR_DATA			Циклический мониторинг
TYP		BYTE	0		Циклический мониторинг
ADRESS		STRING(10)	'сигнал 0'		Циклический мониторинг
DS		DATE_AND_TIME	DT#2017-1-12-11:4...		Циклический мониторинг
TS		TIME	T#1d6h55m9s911ms		Циклический мониторинг
DATA		ARRAY [0..7] OF...			Циклический мониторинг
DATA[0]		BYTE	205		Циклический мониторинг
DATA[1]		BYTE	204		Циклический мониторинг
DATA[2]		BYTE	140		Циклический мониторинг
DATA[3]		BYTE	63		Циклический мониторинг
DATA[4]		BYTE	0		Циклический мониторинг
DATA[5]		BYTE	0		Циклический мониторинг
DATA[6]		BYTE	0		Циклический мониторинг
DATA[7]		BYTE	0		Циклический мониторинг
aMonitorData[1]		ESR_DATA			Циклический мониторинг

Рис. 4.9. Пример работы с ФБ **ESR_MON_R4** на языке CFC

4.4. ESR_MON_X8

Тип модуля: ФБ	**Переменная**	**Тип**	**Описание**
Входы	s0…s7	BYTE	Сигналы для мониторинга.
	DT_in	DT	Текущие дата и время.
	Mode	BYTE	Режим работы блока.
Выходы	ESR_Flag	BOOL	Признак обновления данных.
Входы-выходы	ESR_out	ARRAY [0…3] OF ESR_DATA	Информация мониторинга.
Параметры	a0…a7	STRING(10)	Названия сигналов.
Используемые модули	T_PLC_MS, STATUS_TO_ESR		

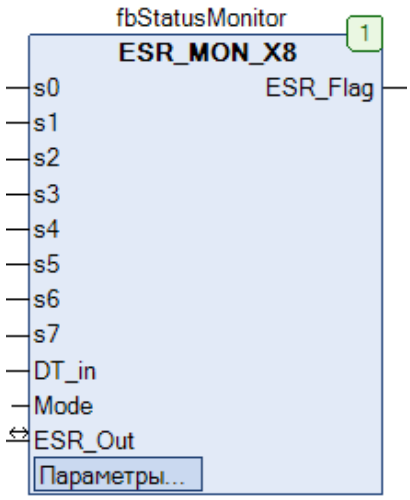


Рис. 4.10. Внешний вид **ФБ ESR_MON_X8** на языке CFC

Функциональный блок **ESR_MON_X8** применяется для мониторинга от одной до восьми переменных типа BYTE (**s0...s7**), которые содержат информацию о состоянии других ФБ. Такие переменные обычно называются **Status**. В качестве примера можно привести выход **Status** ФБ SEQUENCE_4. На вход **DT_in** подается значение системного времени в формате DT (считанного с помощью одной из доступных пользователю библиотек). Вход **Mode** определяет режим работы блока:

- Mode=1 – обрабатываются только сообщения об ошибках;
- Mode=2 – обрабатываются сообщения об ошибках и состоянии ФБ;
- Mode=3 – обрабатываются сообщения об ошибках, состоянии ФБ.

При изменении значения переменных **s0…s7** данные мониторинга сохраняются в массив **ESR_Out**, элементы которого являются экземплярами структуры типа ESR_DATA. Массив содержит 4 элемента и, соответственно, позволяет хранить данные мониторинга для четырех одновременно изменившихся переменных. Иными словами, при одновременном (произошедшем в одном цикле ПЛК) изменении всех восьми переменных будет сохранена информация только о четырех (приоритет отдается сигналам с меньшим номером). Для ведения архива мониторинга можно использовать ФБ ESR_Collect. Выход **ESR_flag** принимает значение **TRUE** на один цикл при обновлении данных мониторинга.

Структура ESR_DATA включает следующую информацию:

Переменная структуры	Тип	Описание
TYP	BYTE	Код статуса. Изменяется только на один цикл, после чего сбрасывается в 0.
ADRESS	STRING(10)	Описание переменной (указывается в параметрах модуля).
DS	DT	Дата и время изменения.
TS	TIME	Относительное время изменения (в мс после старта ПЛК).
DATA	ARRAY [0..7] OF BYTE	В нулевом байте содержится код статуса.

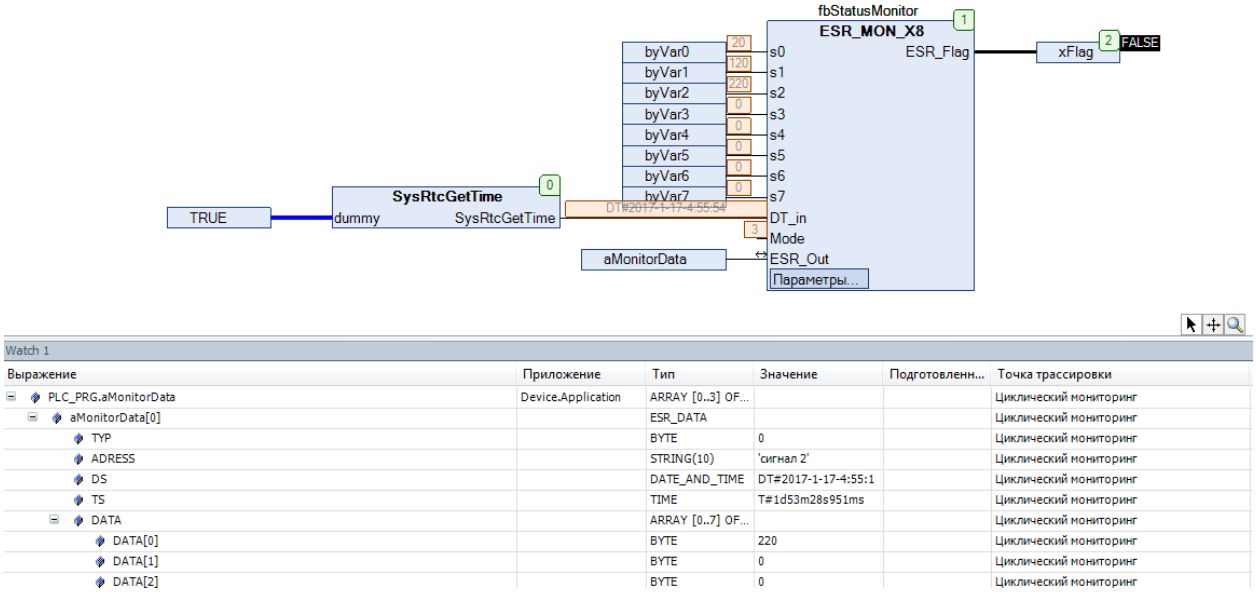


Рис. 4.11. Пример работы с ФБ **ESR_MON_X8** на языке CFC. Изменение переменной byVar2 (‘Сигнал 0’)

4.5. OSCAT_VERSION

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	IN	BOOL	Вход функции.
Выходы	OSCAT_VERSION	DWORD	Версия библиотеки/дата релиза.

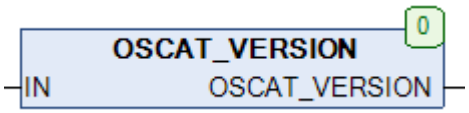


Рис. 4.12. Внешний вид функции **OSCAT_VERSION** на языке CFC

Функция **OSCAT_VERSION** возвращает версию библиотеки (в формате DWORD) при значении IN=FALSE и дату релиза данной версии (в формате DWORD) при значении IN=TRUE.

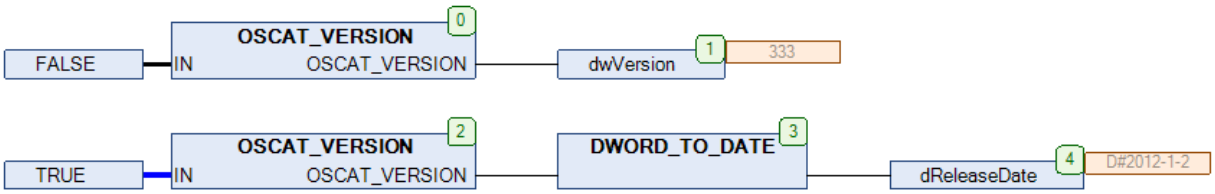


Рис. 4.13. Пример работы с функцией **OSCAT_VERSION** на языке CFC

4.6. STATUS_TO_ESR

Тип модуля: функция	Переменная	Тип	Описание
Входы	status	BYTE	Код статуса.
	adress	STRING(10)	Описание переменной.
	DT_in	DT	Дата и время изменения.
	TS	TIME	Относительное время изменения (в мс после старта ПЛК).
Выходы	STATUS_TO_ESR	ESR_DATA	Информация мониторинга.

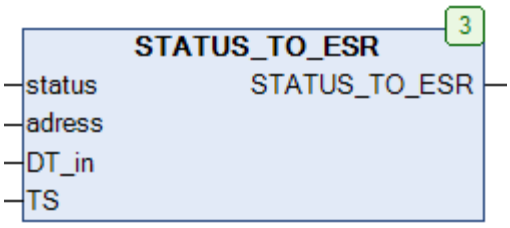


Рис. 4.14. Внешний вид функции **STATUS_TO_ESR** на языке CFC

Функция **STATUS_TO_ESR** собирает элемент структуры ESR_DATA из отдельных переменных (их описание приведено в п. 3.8).

Коды статуса **1...99** соответствуют сообщениям об ошибках, **100…199** – состояниям ФБ, **200…255** – отладочным сообщениям.

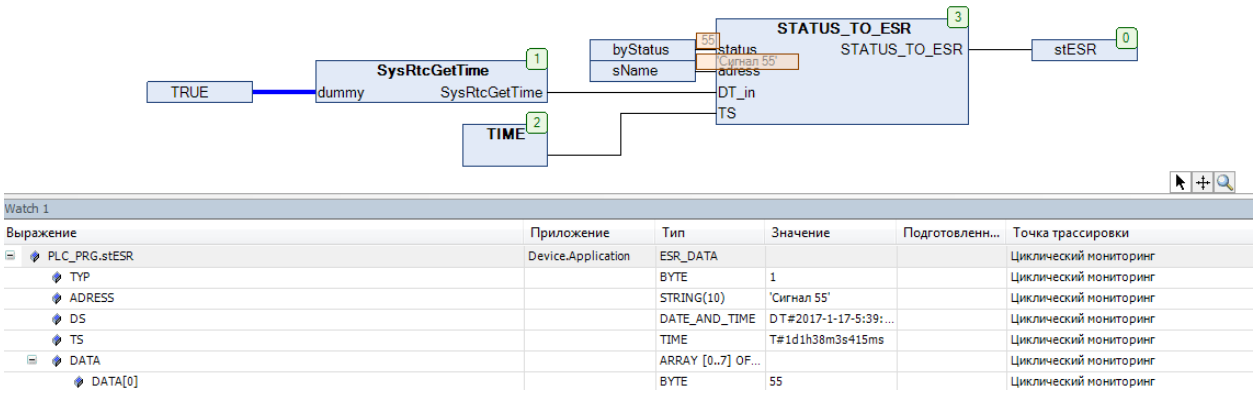


Рис. 4.15. Пример работы с функцией **STATUS_TO_ESR** на языке CFC

5. Математические функции

5.1. ACOSH

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	X	REAL	Аргумент.
Выходы	ACOSH	REAL	Значение гиперболического арккосинуса.

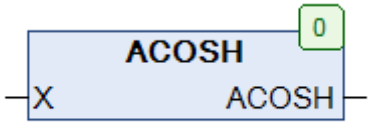


Рис. 5.1. Внешний вид функции **ACOSH** на языке CFC

Функция **ACOSH** возвращает значение гиперболического арккосинуса, вычисленное по формуле

$$\mathrm{ACOSH}(X) = \ln(X + \sqrt{X^2 - 1})$$

Область определения: $1 \leq X < +\infty$

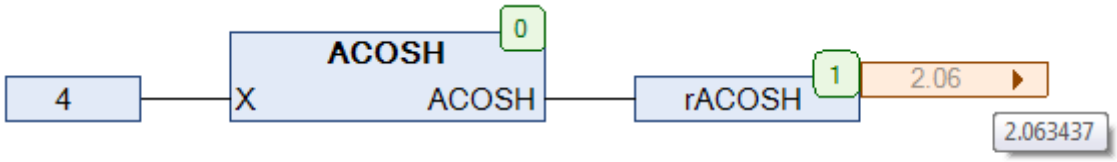


Рис. 5.2. Пример работы с функцией **ACOSH** на языке CFC

5.2. ACOTH

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	X	REAL	Аргумент.
Выходы	ACOTH	REAL	Значение гиперболического арккотангенса.

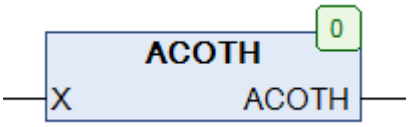


Рис. 5.3. Внешний вид функции **ACOTH** на языке CFC

Функция **ACOTH** возвращает значение гиперболического арккотангенса, вычисленное по формуле

$$\mathrm{ACOTH}(\mathrm{X}) = \frac{1}{2}\ln(\frac{\mathrm{X}+1}{\mathrm{X}-1})$$

Область определения: $|\mathrm{X}| > 1$

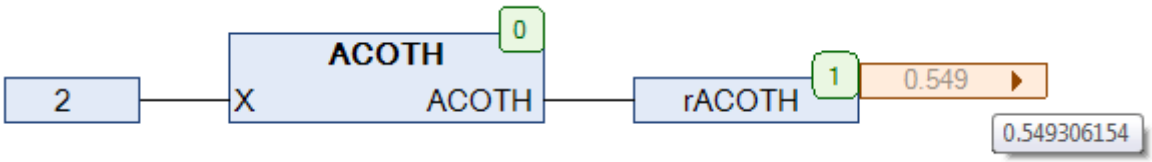


Рис. 5.4. Пример работы с функцией **ACOTH** на языке CFC

5.3. AGDF

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	X	REAL	Аргумент.
Выходы	AGDF	REAL	Значение обратной функции Гудермана.

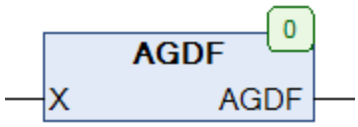


Рис. 5.5. Внешний вид функции **AGDF** на языке CFC

Функция **AGDF** возвращает значение обратной функции Гудермана, вычисленное по формуле

$$\mathrm{AGDF(X)} = \ln\left(\frac{1+\sin(\mathrm{X})}{\cos(\mathrm{X})}\right)$$

Текущая реализация функции гарантирует приемлемую точность вычислений только для x, принадлежащих интервалу $-\pi/2 \leq \mathrm{X} \leq \pi/2$.

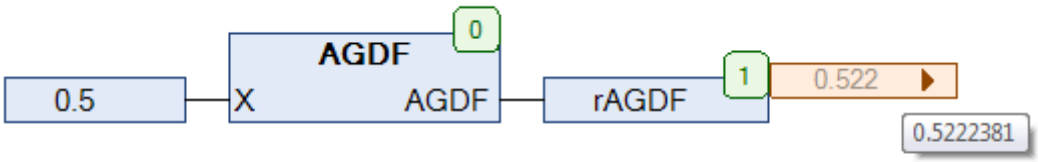


Рис. 5.6. Пример работы с функцией **AGDF** на языке CFC

5.4. ASINH

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	X	REAL	Аргумент.
Выходы	ASINH	REAL	Значение гиперболического арксинуса.

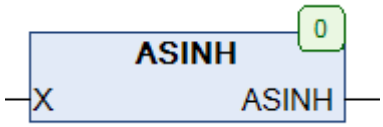


Рис. 5.7. Внешний вид функции **ASINH** на языке CFC

Функция **ASINH** возвращает значение гиперболического арксинуса, вычисленное по формуле

$$\mathrm{ASINH}(X) = \ln(X + \sqrt{X^2 + 1})$$

Область определения: $-\infty < X < +\infty$

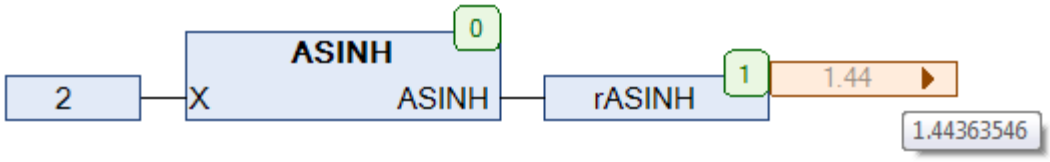


Рис. 5.8. Пример работы с функцией **ASINH** на языке CFC

5.5. ATAN2

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	Y	REAL	Координата Y.
	X	REAL	Координата X.
Выходы	ATAN	REAL	Значение арктангенса для отношения Y/X.

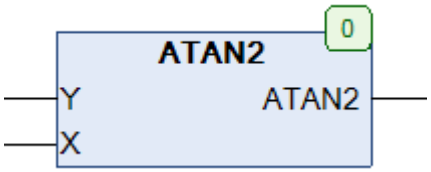


Рис. 5.9. Внешний вид функции **ATAN2** на языке CFC

Функция **ATAN2** возвращает значение арктангенса для заданных координат X и Y. Арктангенс – это угол между осью X и линией, проведенной из начала координат (0,0) в точку с координатами (X,Y). Угол определяется в радианах в диапазоне $-\pi/2 \leq \mathrm{ATAN2} \leq \pi/2$.

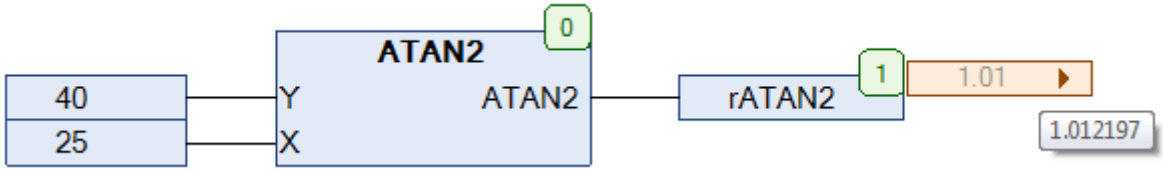


Рис. 5.10. Пример работы с функцией **ATAN2** на языке CFC

5.6. ATANH

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	X	REAL	Аргумент.
Выходы	ATANH	REAL	Значение гиперболического арктангенса.

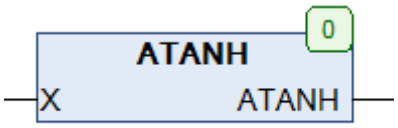


Рис. 5.11. Внешний вид функции **ATANH** на языке CFC

Функция **ATANH** возвращает значение гиперболического арктангенса, вычисленное по формуле

$$\mathrm{ATANH}(\mathrm{X}) = \frac{1}{2}\ln(\frac{1+\mathrm{X}}{1-\mathrm{X}})$$

Область определения: $-1 < \mathrm{X} < 1$

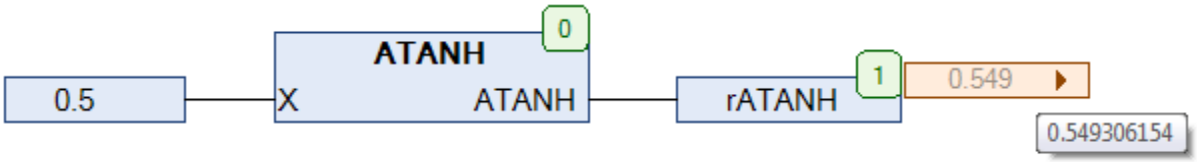


Рис. 5.12. Пример работы с функцией **ATANH** на языке CFC

5.7. BETA

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	X	REAL	Аргумент X.
	Y	REAL	Аргумент Y.
Выходы	BETA	REAL	Значение бета-функции.
Используемые модули	GAMMA		

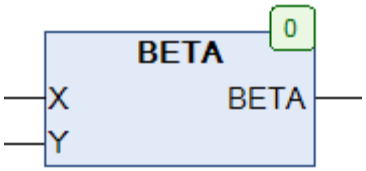


Рис. 5.13. Внешний вид функции **BETA** на языке CFC

Функция **BETA** возвращает значение бета-функции, вычисленное по формуле

$$\mathrm{BETA}(X, Y) = \frac{\Gamma(X) \cdot \Gamma(Y)}{\Gamma(X + Y)},$$

где Г – гамма-функция

Область определения: $X > 1,\ Y > 1$

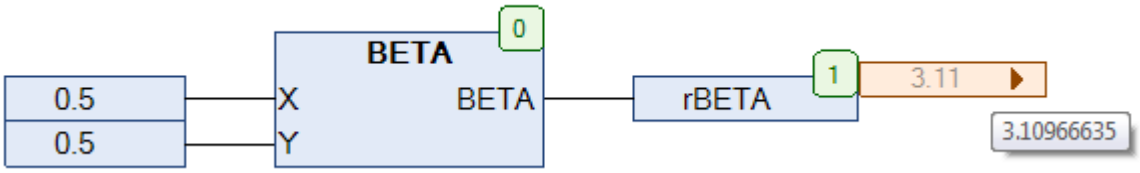


Рис. 5.14. Пример работы с функцией **BETA** на языке CFC

5.8. BINOM

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	N	INT	Аргумент N.
	K	INT	Аргумент K.
Выходы	BINOM	DINT	Биномиальный коэффициент из N по K.

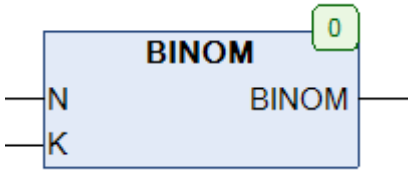


Рис. 5.15. Внешний вид функции **BINOM** на языке CFC

Функция **BINOM** возвращает значение биномиального коэффициента из N по K, вычисленное по формуле

$$\mathrm{BINOM}(\mathrm{N},\mathrm{K}) = \frac{\mathrm{N}!}{\mathrm{K}! \cdot (\mathrm{N}-\mathrm{K})!}$$

Функция возвращает корректные значения только для $N \geq K > 0$.

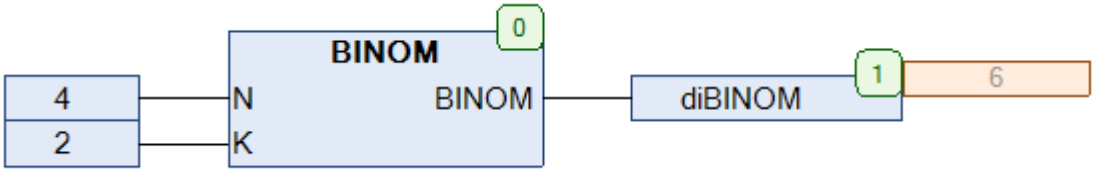


Рис. 5.16. Пример работы с функцией **BINOM** на языке CFC

5.9. CAUCHY

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	X	REAL	Аргумент.
	T	REAL	Коэффициент сдвига.
	U	REAL	Коэффициент масштаба.
Выходы	CAUCHY	REAL	Плотность вероятности для распределения Коши.

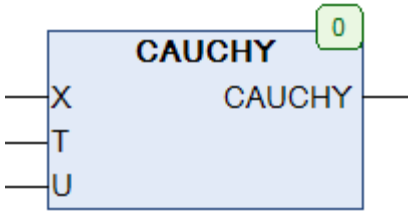


Рис. 5.17. Внешний вид функции CAUCHY на языке CFC

Функция **CAUCHY** возвращает значение плотности вероятности для распределения Коши, вычисленное по формуле

$$\mathrm{CAUCHY(X)} = \frac{1}{\pi} \cdot \frac{\mathrm{U}}{\mathrm{U}^2 + (\mathrm{X} - \mathrm{T})^2}, \qquad \mathrm{U} > 0$$

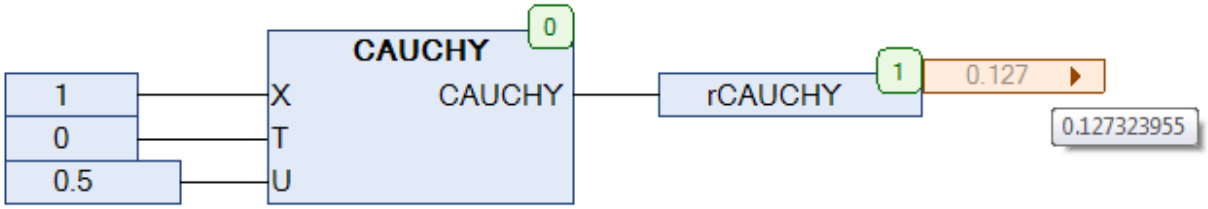


Рис. 5.18. Пример работы с функцией **CAUCHY** на языке CFC

5.10. CAUCHYCD

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	X	REAL	Аргумент.
	T	REAL	Коэффициент сдвига.
	U	REAL	Коэффициент масштаба.
Выходы	CAUCHYCD	REAL	Функция распределения Коши.

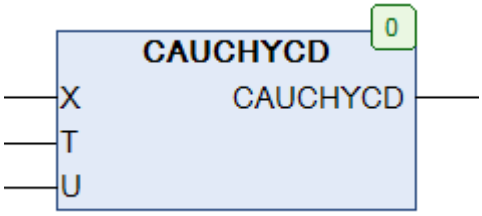


Рис. 5.19. Внешний вид функции **CAUCHYCD** на языке CFC

Функция **CAUCHYCD** возвращает значение функции распределения Коши по формуле

$$\mathrm{CAUCHYCD(X)} = \frac{1}{\pi} \cdot \mathrm{arctg}\left(\frac{\mathrm{X}-\mathrm{T}}{\mathrm{U}}\right) + \frac{1}{2}, \qquad \mathrm{U} > 0$$

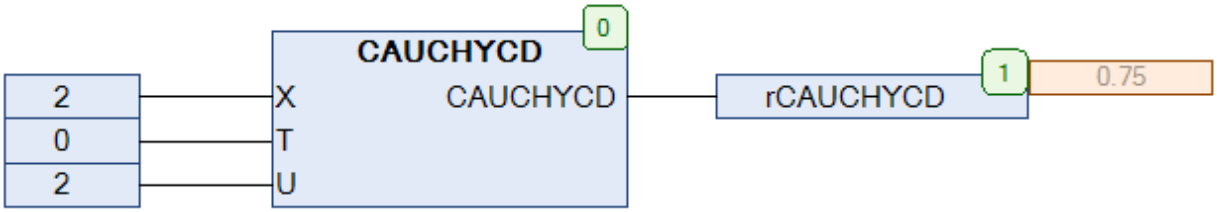


Рис. 5.20. Пример работы с функцией **CAUCHYCD** на языке CFC

5.11. CEIL

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	X	REAL	Округляемое значение.
Выходы	CEIL	INT	Округленное значение.

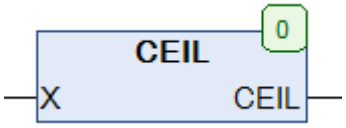


Рис. 5.21. Внешний вид функции **CEIL** на языке CFC

Функция **CEIL** округляет переменную типа REAL в сторону ближайшего большего целого значения. ***Обратите внимание***, что функция возвращает значение типа INT, что накладывает соответствующие ограничения на величину входной переменной (диапазон возможных значений для типа INT составляет -32768..32767).

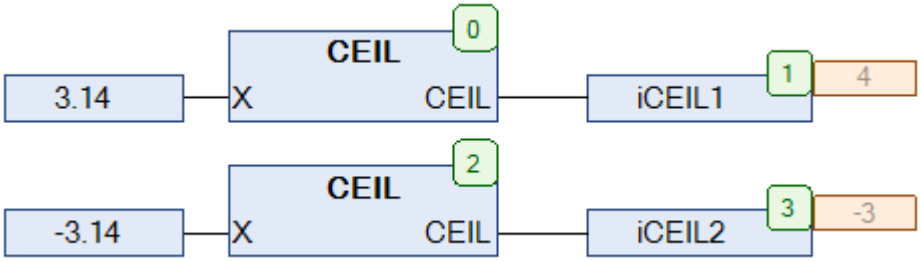


Рис. 5.22. Пример работы с функцией **CEIL** на языке CFC

5.12. CEIL2

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	X	REAL	Округляемое значение.
Выходы	CEIL2	DINT	Округленное значение.

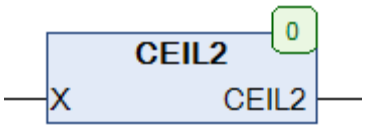


Рис. 5.23. Внешний вид функции **CEIL2** на языке CFC

Функция **CEIL2** округляет переменную типа REAL в сторону ближайшего большего целого значения. В отличие от функции CEIL, данная функция возвращает значение типа DINT, что увеличивает диапазон возможных значений входной переменной.

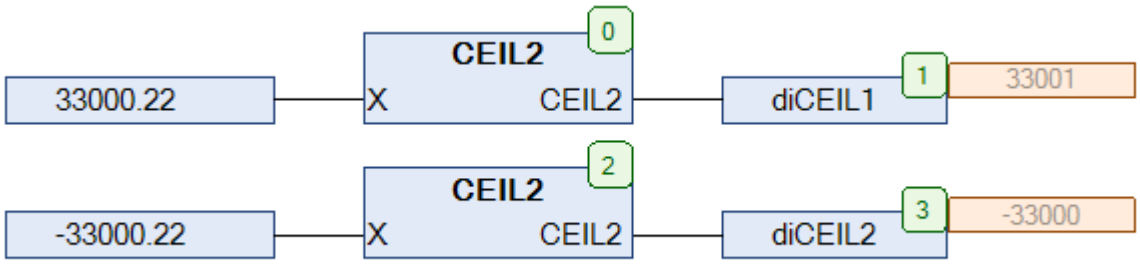


Рис. 5.24. Пример работы с функцией **CEIL2** на языке CFC

5.13. CMP

Тип модуля: функция	Переменная	Тип	Описание
Входы	X	REAL	1-я сравниваемая величина.
	Y	REAL	2-я сравниваемая величина.
	N	INT	Кол-во сравниваемых знаков.
Выходы	CMP	BOOL	Результат сравнения.

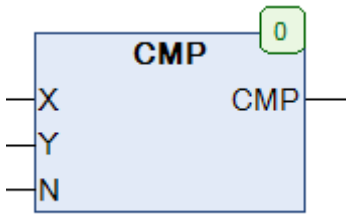


Рис. 5.25. Внешний вид функции **CMP** на языке CFC

Функция **CMP** сравнивает значения X и Y типа REAL и возвращает **TRUE,** если их первые N символов совпадают. В противном случае функция возвращает **FALSE**.

Необходимо отметить, что тип REAL обеспечивает относительную точность в 7-8 десятичных цифр после запятой в заданном диапазоне, поэтому нельзя гарантировать корректную работу функции в 100% случаев.

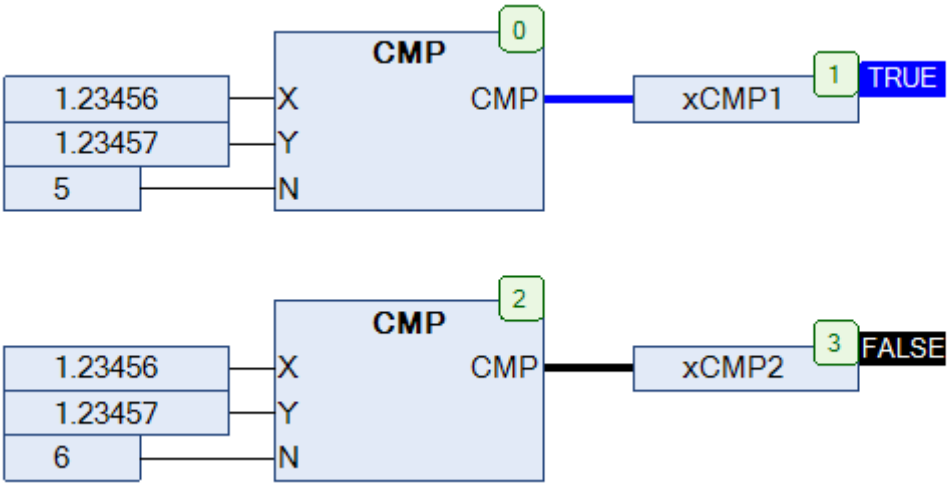


Рис. 5.26. Пример работы с функцией **CMP** на языке CFC

5.14. COSH

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	X	REAL	Аргумент.
Выходы	COSH	REAL	Значение гиперболического косинуса.

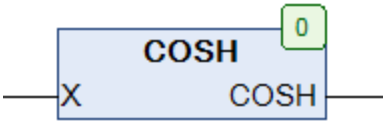


Рис. 5.27. Внешний вид функции **COSH** на языке CFC

Функция **COSH** возвращает значение гиперболического косинуса, вычисленное по формуле

$$\mathrm{COSH}(X) = \frac{e^{X} + e^{-X}}{2}$$

Область определения: $-\infty < X < +\infty$

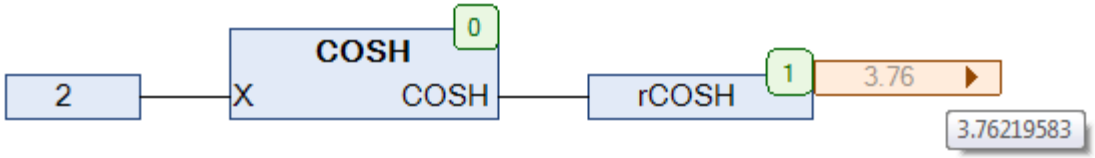


Рис. 5.28. Пример работы с функцией **COSH** на языке CFC

5.15. COTH

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	X	REAL	Аргумент.
Выходы	COTH	REAL	Значение гиперболического котангенса.

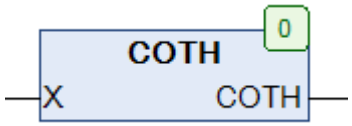


Рис. 5.29. Внешний вид функции **COTH** на языке CFC

Функция **COTH** возвращает значение гиперболического котангенса, вычисленное по формуле

$$\mathrm{COTH}(X) = \frac{e^{2X} + 1}{e^{2X} - 1}$$

Область определения: $X \neq 0$

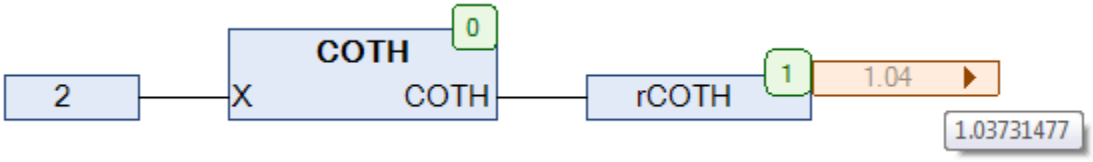


Рис. 5.30. Пример работы с функцией **COTH** на языке CFC

5.16. D_TRUNC

Тип модуля: функция	Переменная	Тип	Описание
Входы	X	REAL	Округляемое значение.
Выходы	D_TRUNC	DINT	Округленное значение.

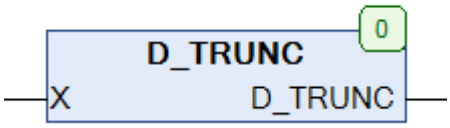


Рис. 5.31. Внешний вид функции **D_TRUNC** на языке CFC

Функция **D_TRUNC** возвращает целую часть переменной типа REAL (***обратите внимание,*** что речь идет не об округлении до ближайшего целочисленного значения). Функция **TRUNC()** из стандарта МЭК 61131-3 поддерживает тип DINT не во всех средах программирования; более того, даже преобразование **REAL_TO_DINT** на разных системах может приводить к разному результату. Функция **D_TRUNC** анализирует работу преобразования и возвращает корректное значение.

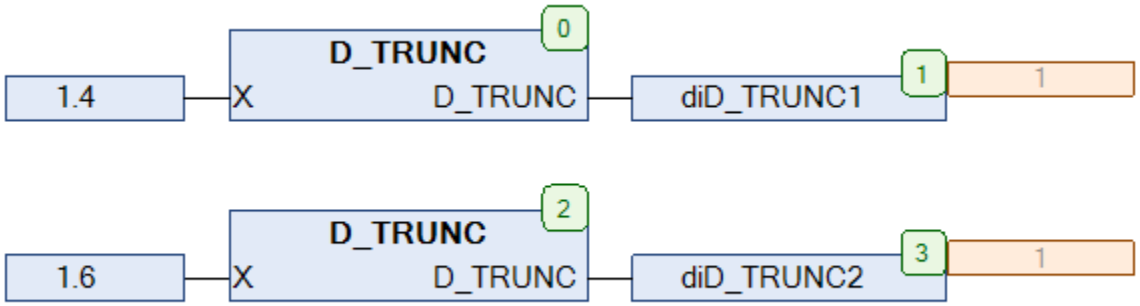


Рис. 5.32. Пример работы с функцией **D_TRUNC** на языке CFC

5.17. DEC1

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	X	INT	Декрементируемое значение 1.
	N	INT	Декрементируемое значение 2.
Выходы	DEC1	INT	Результат декремента.

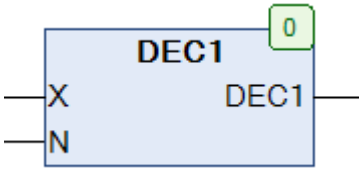


Рис. 5.33. Внешний вид функции **DEC1** на языке CFC

Функция **DEC1** декрементирует значение переменной X. Если X = 0, то функция возвращает декрементированное значение переменной N.

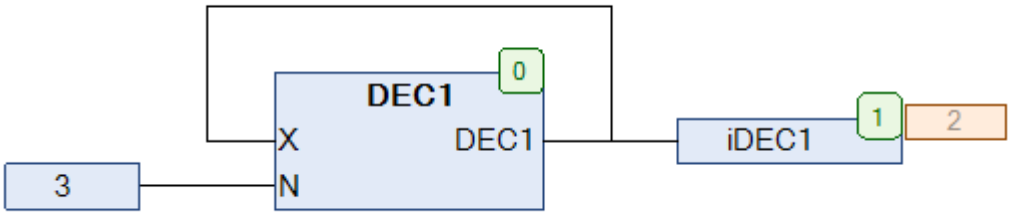


Рис. 5.34. Пример работы с функцией **DEC1** на языке CFC

При работе согласно рис. 5.34 функция циклически возвращает последовательность **2,1,0**.

5.18. DEG

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	rad	REAL	Значение угла в радианах.
Выходы	DEG	REAL	Значение угла в градусах.

Рис. 5.35. Внешний вид функции **DEG** на языке CFC

Функция **DEG** преобразует значение угла из радиан в градусы. Если входная переменная rad превышает 2π, то из нее будет вычитаться по π до тех пор, пока ее значение не окажется в интервале $0 < \mathrm{rad} < 2\pi$.

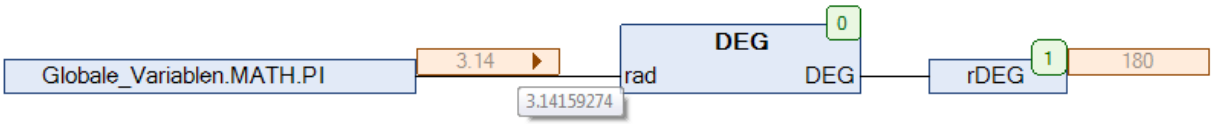


Рис. 5.36. Пример работы с функцией **DEG** на языке CFC

5.19. DIFFER

Тип модуля: функция	Переменная	Тип	Описание
Входы	in1	REAL	Сравниваемое значение 1.
	in2	REAL	Сравниваемое значение 2.
	X	REAL	Допустимая разность значений.
Выходы	DIFFER	TRUE	Флаг сравнения.

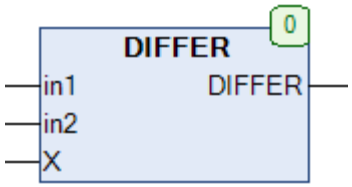


Рис. 5.37. Внешний вид функции **DIFFER** на языке CFC

Функция **DIFFER** возвращает значение **TRUE**, если значения переменных **in1** и **in2** отличаются на величину, превышающую **X**. В противном случае функция возвращает **FALSE**.

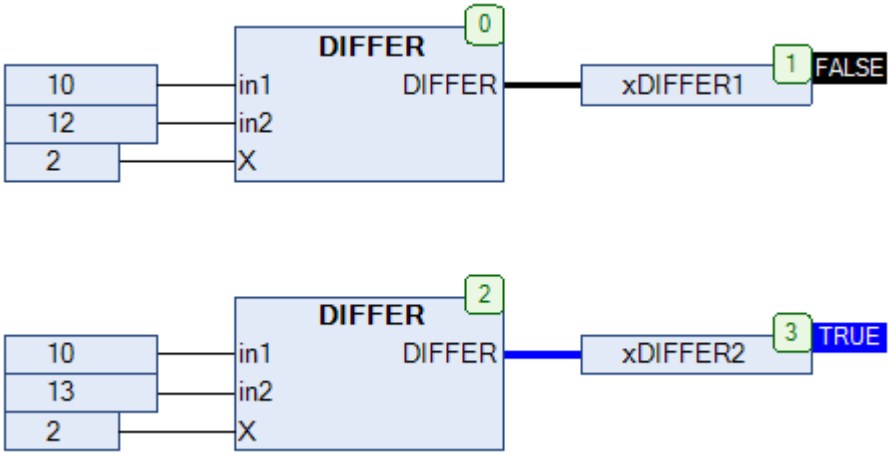


Рис. 5.38. Пример работы с функцией **DIFFER** на языке CFC

5.20. ERF

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	X	REAL	Аргумент.
Выходы	ERF	REAL	Функция ошибок.
Используемые модули	SGN		

Рис. 5.39. Внешний вид функции **ERF** на языке CFC

Функция **ERF** возвращает значение функции ошибок для аргумента **X**, вычисленное по формуле

$$\mathrm{ERF}(X) = \pm\sqrt{1 - \exp\left(-x^2 \cdot \frac{\frac{4}{\pi} + ax^2}{1 + +ax^2}\right)} \quad \text{где } a = \frac{8}{3\pi} \cdot \frac{3-\pi}{\pi-4}$$

Относительная ошибка вычисления не превышает 0.00013.

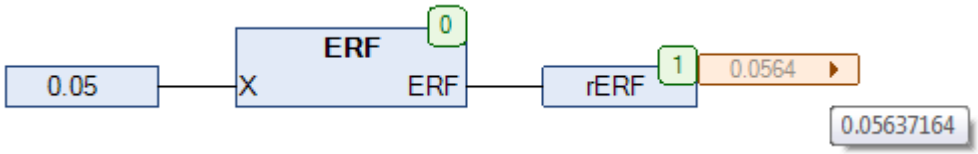


Рис. 5.40. Пример работы с функцией **ERF** на языке CFC

5.21. ERFC

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	X	REAL	Аргумент.
Выходы	ERFC	REAL	Дополнительная функция ошибок.
Используемые модули	ERF		

Рис. 5.41. Внешний вид функции **ERFC** на языке CFC

Функция **ERFC** возвращает значение дополнительной функции ошибок для аргумента **X**, вычисленное по формуле

$$\mathrm{ERFC}(\mathrm{X}) = 1 - \mathrm{ERF}(\mathrm{X})$$

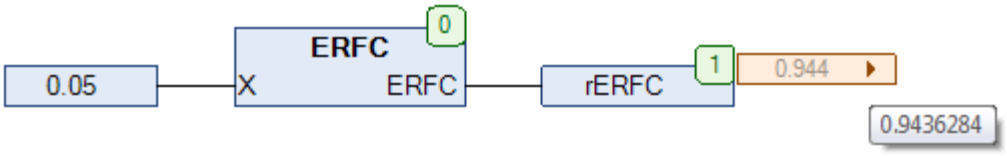


Рис. 5.42. Пример работы с функцией **ERFC** на языке CFC

5.22. EVEN

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	in	DINT	Анализируемое значение.
Выходы	EVEN	BOOL	Флаг четности.

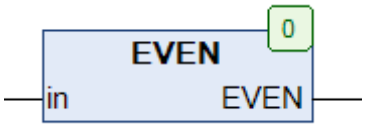


Рис. 5.43. Внешний вид функции **EVEN** на языке CFC

Функция **EVEN** возвращает **TRUE,** если значение переменной **in** является четным, и **FALSE** – если нечетным.

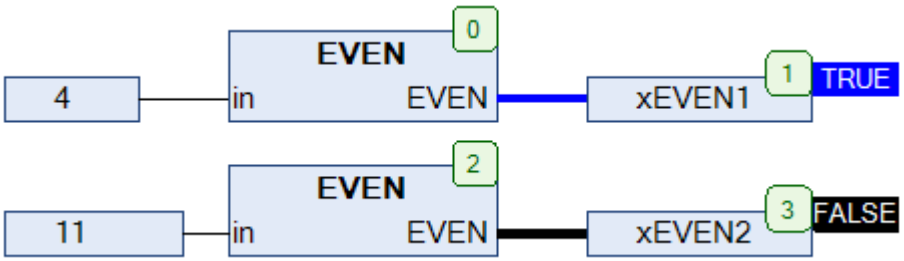


Рис. 5.44. Пример работы с функцией **EVEN** на языке CFC

5.23. EXP10

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	X	REAL	Показатель степени.
Выходы	EXP10	REAL	Показательная функция.

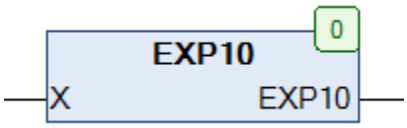


Рис. 5.45. Внешний вид функции **EXP10** на языке CFC

Функция **EXP10** возвращает значение показательной функции с основанием степени 10, вычисленное по формуле

$$\mathrm{EXP10(X)} = 10^{\mathrm{X}}$$

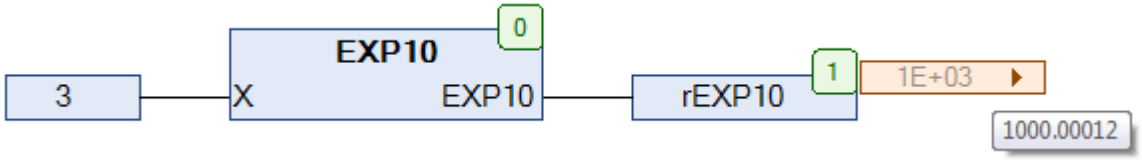


Рис. 5.46. Пример работы с функцией **EXP10** на языке CFC

5.24. EXPN

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	X	REAL	Основание степени.
	N	INT	Показатель степени.
Выходы	EXPN	REAL	Показательная функция.

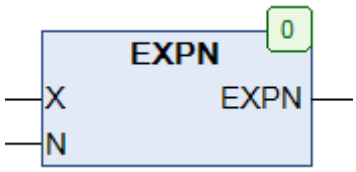


Рис. 5.47. Внешний вид функции **EXPN** на языке CFC

Функция **EXPN** возвращает значение показательной функции с целочисленным основанием степени, вычисленное по формуле

$$\mathrm{EXPN(X)} = \mathrm{X^N}$$

Функция **EXPN** разработана специально для ПЛК без FPU (математического сопроцессора) и выполняется приблизительно в 30 раз быстрее функции **EXPT**, определенной стандартом МЭК 61131-3.

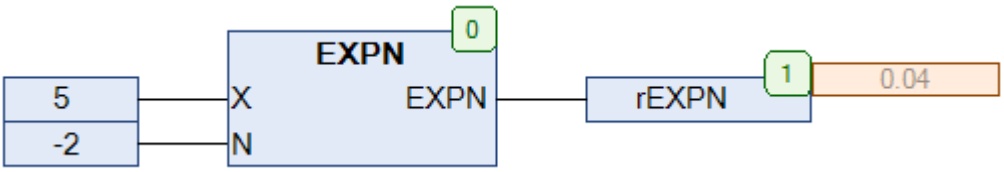


Рис. 5.48. Пример работы с функцией **EXPN** на языке CFC

5.25. FACT

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	X	INT	Аргумент.
Выходы	FACT	DINT	Значение факториала.

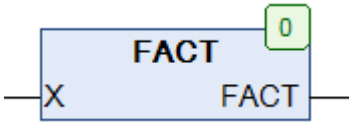


Рис. 5.49. Внешний вид функции **FACT** на языке CFC

Функция **FACT** возвращает значение факториала **X**. Функция возвращает корректные значения при X, принадлежащем интервалу $0 \leq X \leq 12$. Во всех остальных случаях функция возвращает **-1**.

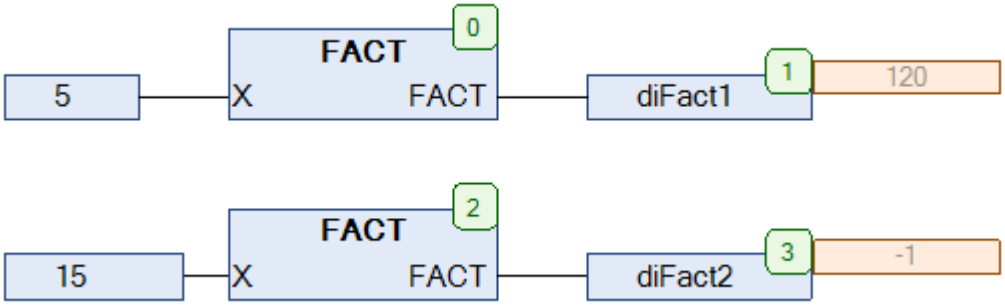


Рис. 5.50. Пример работы с функцией **FACT** на языке CFC

5.26. FIB

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	X	INT	Номер элемента последовательности.
Выходы	FIB	DINT	Значение элемента последовательности.

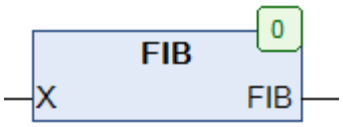


Рис. 5.51. Внешний вид функции **FIB** на языке CFC

Функция **FIB** возвращает значение **X**-го элемента последовательности Фибоначчи. Функция возвращает корректные значения при **X**, принадлежащему интервалу $0 \leq X \leq 46$. Во всех остальных случаях функция возвращает **-1**.

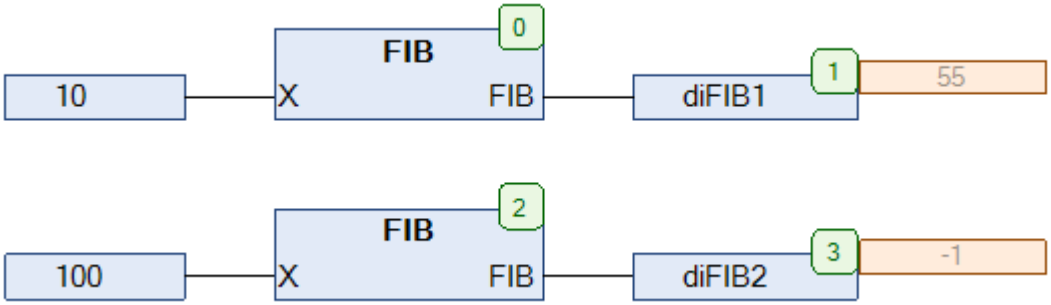


Рис. 5.52. Пример работы с функцией **FIB** на языке CFC

5.27. FLOOR

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	X	REAL	Округляемое значение.
Выходы	FLOOR	INT	Округленное значение.

Рис. 5.53. Внешний вид функции **FLOOR** на языке CFC

Функция **FLOOR** округляет переменную типа REAL в сторону ближайшего меньшего целого значения. ***Обратите внимание***, что функция возвращает значение типа INT, что накладывает соответствующие ограничения на величину входной переменной (диапазон возможных значений для типа INT составляет -32768..32767).

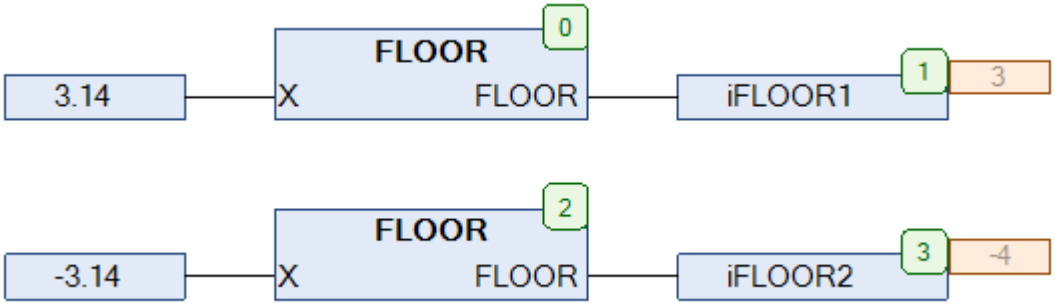


Рис. 5.54. Пример работы с функцией **FLOOR** на языке CFC

5.28. FLOOR2

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	X	REAL	Округляемое значение.
Выходы	FLOOR2	DINT	Округленное значение.

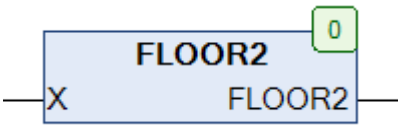


Рис. 5.55. Внешний вид функции **FLOOR2** на языке CFC

Функция **FLOOR2** округляет переменную типа REAL в сторону ближайшего меньшего целого значения. В отличие от функции FLOOR, данная функция возвращает значение типа DINT, что увеличивает диапазон возможных значений входной переменной.

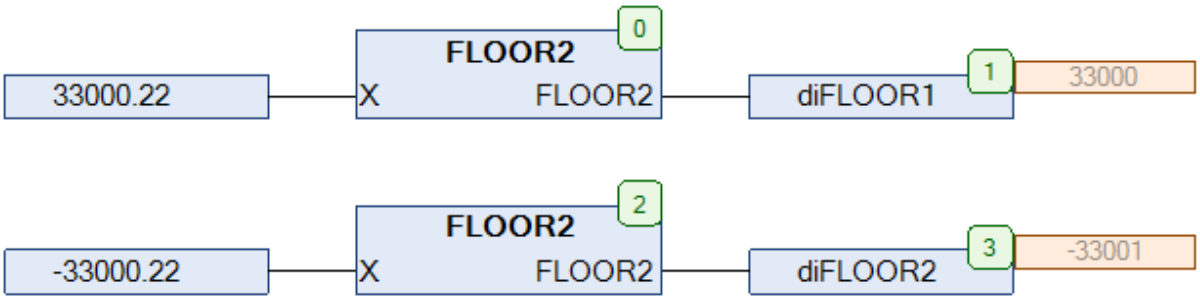


Рис. 5.56. Пример работы с функцией **FLOOR2** на языке CFC

5.29. FRACT

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	X	REAL	Значение с плавающей точкой.
Выходы	FRACT	REAL	Дробная часть значения.
Используемые модули	D_TRUNC		

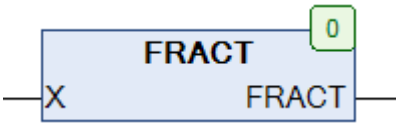


Рис. 5.57. Внешний вид функции **FRACT** на языке CFC

Функция **FRACT** возвращает дробную часть значения с плавающей точкой. Необходимо отметить, что тип REAL обеспечивает относительную точность в 7-8 десятичных цифр после запятой в заданном диапазоне, поэтому нельзя гарантировать корректную работу функции в 100% случаев.

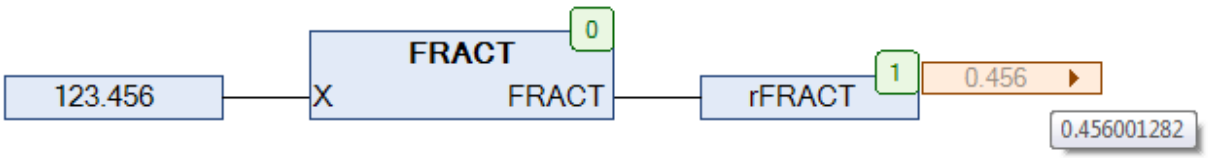


Рис. 5.58. Пример работы с функцией **FRACT** на языке CFC

5.30. GAMMA

Тип модуля: функция	Переменная	Тип	Описание
Входы	X	REAL	Аргумент.
Выходы	GAMMA	REAL	Значение гамма-функции.

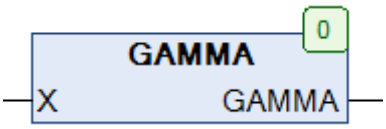


Рис. 5.59. Внешний вид функции GAMMA на языке CFC

Функция **GAMMA** возвращает значение гамма-функции, вычисленное по аппроксимирующей формуле Герго Немеса:

$$\mathrm{GAMMA}(X) = \sqrt{\frac{2\pi}{X}} \cdot \left[\frac{1}{e} \cdot \left(X + \frac{1}{12X - \frac{1}{10X}}\right)\right]$$

Область определения: $X > 0$

Функция может использоваться для приближенного вычисления факториалов чисел, что будет полезным в тех случаях, когда не может быть использована функция FACT. Следует помнить, что гамма-функция возвращает не факториал X, а факториал его декремента.

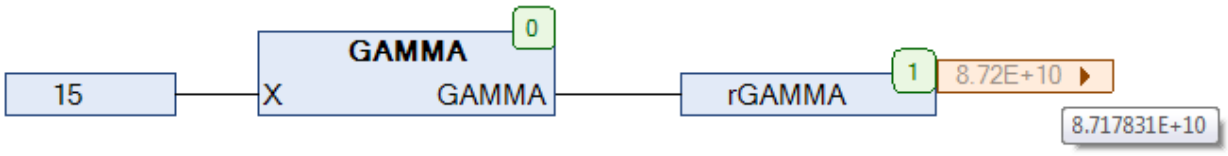


Рис. 5.60. Пример работы с функцией GAMMA на языке CFC

5.31. GAUSS

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	X	REAL	Аргумент.
	U	REAL	Коэффициент сдвига.
	SI	REAL	Коэффициент масштаба.
Выходы	GAUSS	REAL	Плотность вероятности для распределения Гаусса.

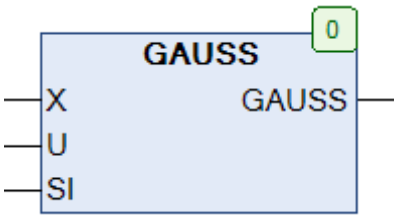


Рис. 5.61. Внешний вид функции **GAUSS** на языке CFC

Функция **GAUSS** возвращает значение плотности вероятности для распределения Гаусса (нормального распределения), вычисленное по формуле

$$\mathrm{GAUSS(X)} = \frac{1}{\sigma\sqrt{2\pi}} \cdot \mathrm{e}^{-\frac{(X-\mu)^2}{2\sigma^2}}, \quad \sigma > 0$$

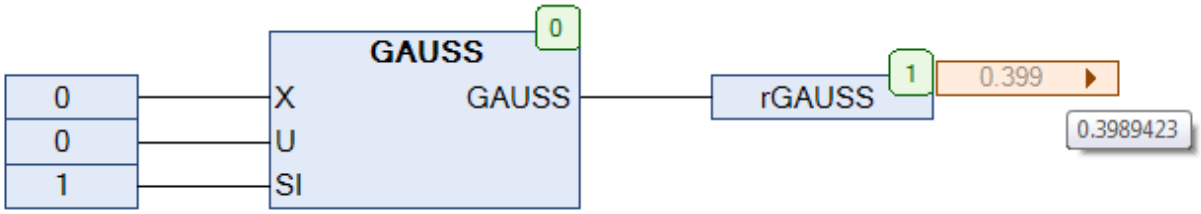


Рис. 5.62. Пример работы с функцией **GAUSS** на языке CFC

5.32. GAUSSCD

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	X	REAL	Аргумент.
	U	REAL	Параметр сдвига.
	SI	REAL	Параметр масштаба.
Выходы	GAUUSCD	REAL	Функция распределения Гаусса.
Используемые модули	ERF		

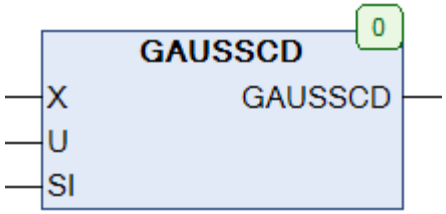


Рис. 5.63. Внешний вид функции **GAUSSCD** на языке CFC

Функция **GAUSSCD** возвращает значение функции распределения Гаусса (нормального распределения), вычисленное по формуле

$$\mathrm{GAUSSCD(X)} = \frac{1}{2}\left[1 + ERF\left(\frac{X - \mu}{\sigma\sqrt{2}}\right)\right], \qquad \mathrm{U} > 0$$

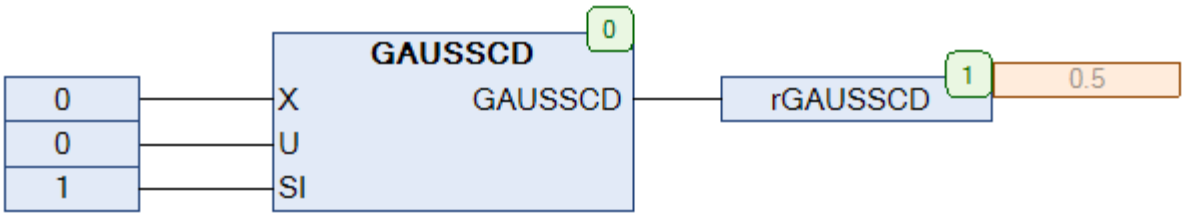


Рис. 5.64. Пример работы с функцией **GAUSSCD** на языке CFC

5.33. GCD

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	A	DINT	1-ое значение.
	B	DINT	2-ое значение.
Выходы	GCD	INT	Наибольший общий делитель для A и B.

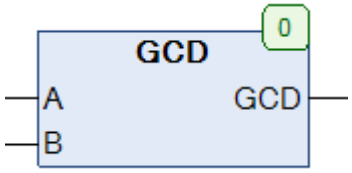


Рис. 5.65. Внешний вид функции **GCD** на языке CFC

Функция GCD возвращает значение [наибольшего общего делителя](#) для чисел A и B.

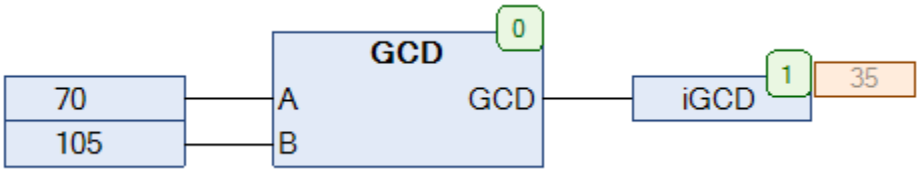


Рис. 5.66. Пример работы с функцией **GCD** на языке CFC

5.34. GDF

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	X	REAL	Аргумент.
Выходы	GDF	REAL	Значение функции Гудермана.

Рис. 5.67. Внешний вид функции **GDF** на языке CFC

Функция **GDF** возвращает значение функции Гудермана, вычисленное по формуле

$$\mathrm{GDF}(X) = 2\,\mathrm{atan}\left(e^{X}\right) - \frac{\pi}{2}$$

Область определения: $-\infty \leq X < +\infty$

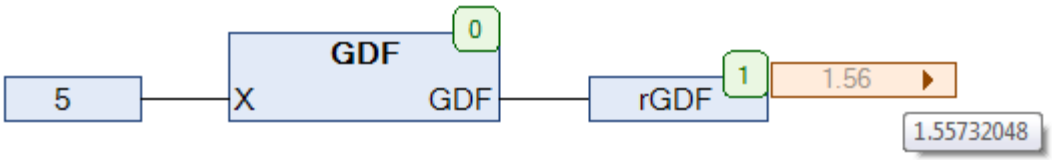


Рис. 5.68. Пример работы с функцией **GDF** на языке CFC

5.35. GOLD

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	X	REAL	Аргумент.
Выходы	GOLD	REAL	Значение функции.

Рис. 5.69. Внешний вид функции **GOLD** на языке CFC

Функция **GOLD** возвращает значение, вычисленное по формуле

$$\mathrm{GOLD}(X) = \frac{X + \sqrt{X^2 + 4}}{2}$$

При X=1 функция возвращает значение [золотого сечения](#).

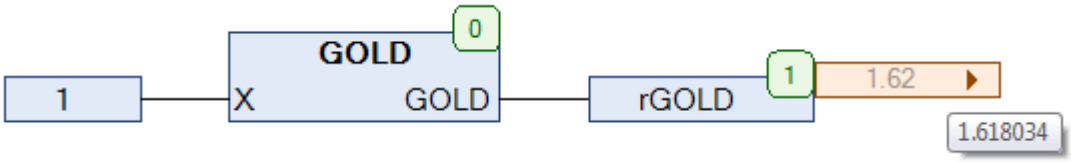


Рис. 5.70. Пример работы с функцией **GOLD** на языке CFC

5.36. HYPOT

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	X	REAL	Длина катета X.
	Y	REAL	Длина катета Y.
Выходы	HYPOT	REAL	Длина гипотенузы.

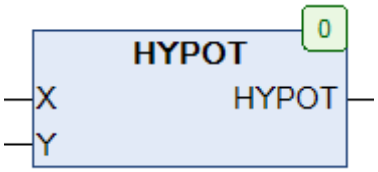


Рис. 5.71. Внешний вид функции **HYPOT** на языке CFC

Функция **HYPOT** возвращает значение длины гипотенузы прямоугольного треугольника, вычисленное по теореме Пифагора.

$$\mathrm{HYPOT} = \sqrt{\mathrm{X}^2 + \mathrm{Y}^2}$$

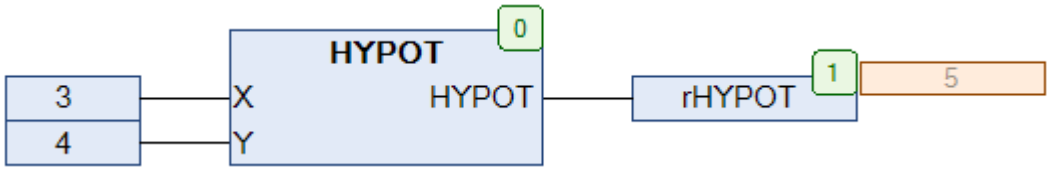


Рис. 5.72. Пример работы с функцией **HYPOT** на языке CFC

5.37. INC

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	X	INT	Входное значение.
	D	INT	Прибавляемое значение.
	M	INT	Максимальное возвращаемое значение.
Выходы	INC	INT	Возвращаемое значение.

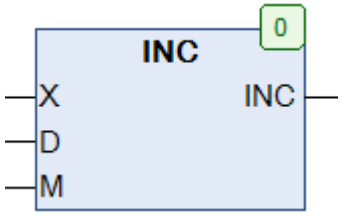


Рис. 5.73. Внешний вид функции **INC** на языке CFC

Функция **INC** складывает значения **X** и **D**, и проверяет результат. Если результат сложения превышает **M**, то функция возвращает **0**. Если результат сложения отрицателен, то функция возвращает **M**. Во всех остальных случаях функция возвращает сумму **X** и **D**. Функция может быть полезной при работе с массивами и буферами, а также абсолютными энкодерами.

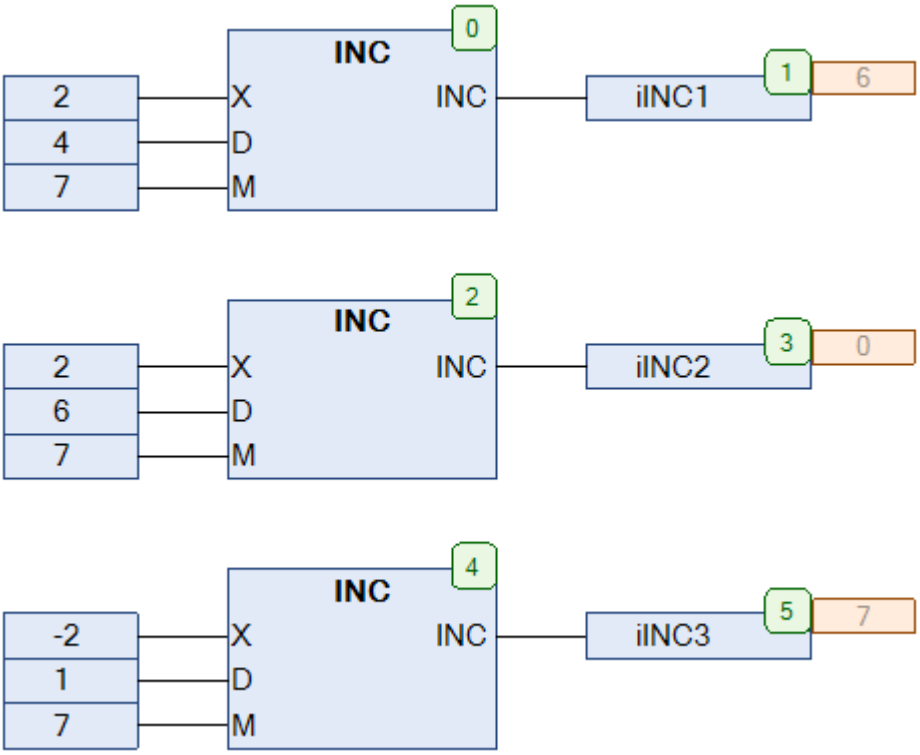


Рис. 5.74. Пример работы с функцией **INC** на языке CFC

5.38. INC1

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	X	INT	Инкрементируемое значение 1.
	N	INT	Инкрементируемое значение 2.
Выходы	INC1	INT	Результат инремента.

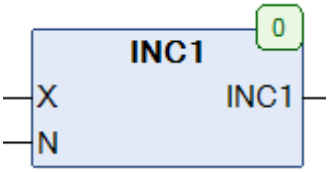


Рис. 5.75. Внешний вид функции **INC1** на языке CFC

Функция **INC1** инкрементирует значение переменной **X** до **N-1**, после чего на выход подается **0**, и продолжается циклический инкремент.

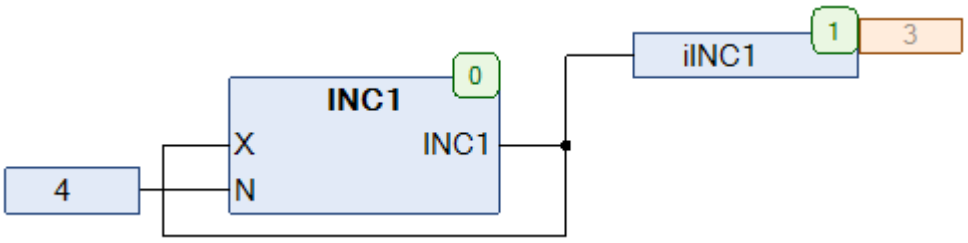


Рис. 5.76. Пример работы с функцией INC1 на языке CFC

При работе согласно рис. 5.76 функция циклически возвращает последовательность **0,1,2,3**.

5.39. INC2

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	X	INT	Входное значение.
	D	INT	Прибавляемое значение.
	L	INT	Нижний предел.
	U	INT	Верхний предел.
Выходы	INC2	INT	Возвращаемое значение.

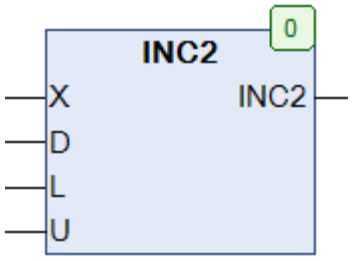


Рис. 5.77. Внешний вид функции **INC2** на языке CFC

Функция **INC2** складывает значения **X** и **D**, и проверяет результат. Если результат сложения превышает **U**, то функция возвращает **L**. Если результат сложения меньше **L**, то функция возвращает **U**. Во всех остальных случаях функция возвращает сумму **X** и **D**. Функция может оказаться полезной при работе с массивами и буферами, а также абсолютными энкодерами.

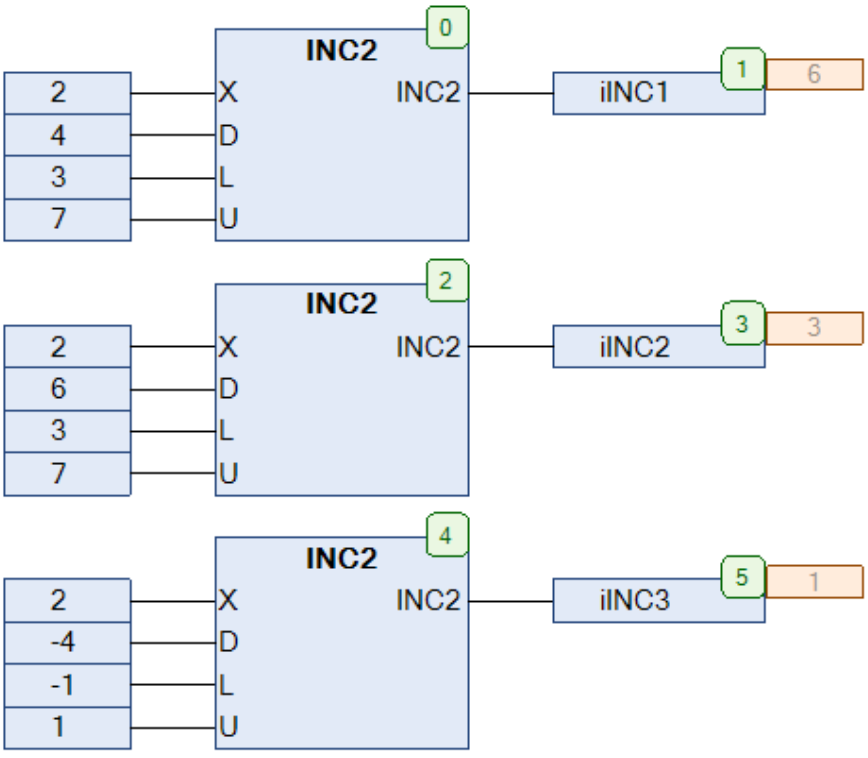


Рис. 5.78. Пример работы с функцией **INC2** на языке CFC

5.40. INV

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	X	INT	Аргумент.
Выходы	INV	INT	Обратное число.

Рис. 5.79. Внешний вид функции **INV** на языке CFC

Функция **INV** возвращает обратное число для **X**.

$$\mathrm{INV}(X) = \frac{1}{X}$$

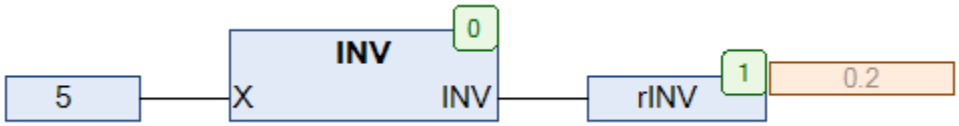


Рис. 5.80. Пример работы с функцией **INV** на языке CFC

5.41. LAMBERT_W

Тип модуля: функция	Переменная	Тип	Описание
Входы	X	REAL	Аргумент.
Выходы	LAMBERT_W	REAL	Значение W-функции Ламберта.

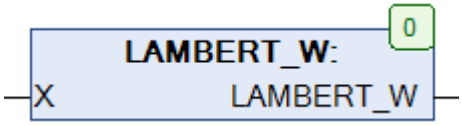


Рис. 5.81. Внешний вид функции **LAMBERT_W** на языке CFC

Функция **LAMBERT_W** возвращает значение W-функции Ламберта, вычисленное по аппроксимационной формуле

$$\mathrm{LAMBERT\_W}(X) = \begin{cases} 0.665 \cdot (1 + 0.0195 \cdot \ln(X+1)) \cdot \ln(X+1) + 0.04, & 0 < X \le 500 \\ \ln(X-4) - \left(1 - \dfrac{1}{\ln(X)}\right) \cdot \ln(\ln(X)), & X > 500 \end{cases}$$

Относительная точность вычисления по данной формуле превышает 10%.

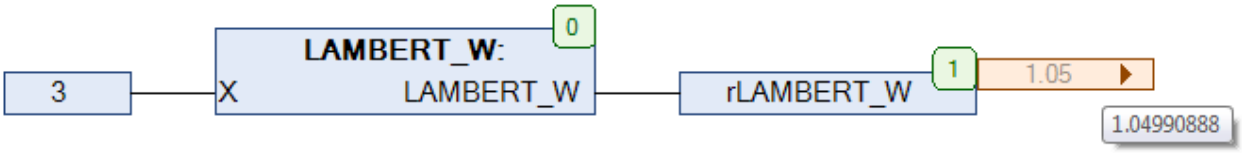


Рис. 5.82. Пример работы с функцией **LAMBERT_W** на языке CFC

5.42. LANGEVIN

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	X	REAL	Аргумент.
Выходы	LANGEVIN	REAL	Значение функции Ланжевена.
Используемые модули	COTH		

Рис. 5.83. Внешний вид функции **LANGEVIN** на языке CFC

Функция **LANGEVIN** возвращает значение функции Ланжевена, вычисленное по формуле

$$\mathrm{LANGEVIN}(X) = \mathrm{COTH}(X) - \frac{1}{X}$$

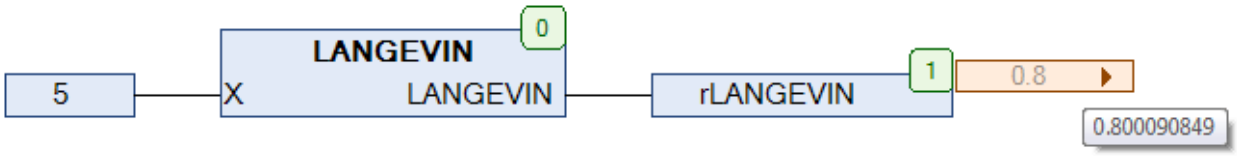


Рис. 5.84. Пример работы с функцией **LANGEVIN** на языке CFC

5.43. MAX3

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	in1	REAL	Сравниваемое значение 1.
	in2	REAL	Сравниваемое значение 2.
	in3	REAL	Сравниваемое значение 3.
Выходы	MAX3	REAL	Наибольшее из сравниваемых значений.

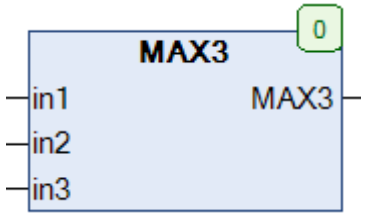


Рис. 5.85. Внешний вид функции **MAX3** на языке CFC

Функция **MAX3** возвращает наибольшее из трех сравниваемых значений. Согласно стандарту МЭК 61131-3 оператор **MAX()** должен иметь настраиваемое число входов, но в некоторых средах разработки их число ограничено двумя – по этой причине в библиотеку **OSCAT** включена функция **MAX3**.

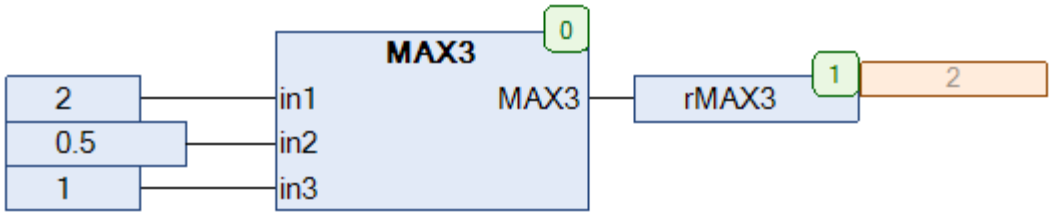


Рис. 5.86. Пример работы с функцией **MAX3** на языке CFC

5.44. MID3

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	in1	REAL	Сравниваемое значение 1.
	in2	REAL	Сравниваемое значение 2.
	in3	REAL	Сравниваемое значение 3.
Выходы	MID3	REAL	Среднее из сравниваемых значений.

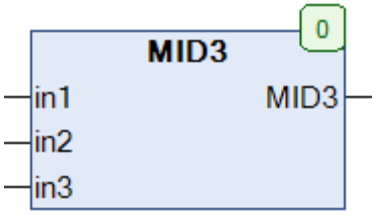


Рис. 5.87. Внешний вид функции **MID3** на языке CFC

Функция **MID3** возвращает среднее (*но не усредненное*) из трех сравниваемых значений.

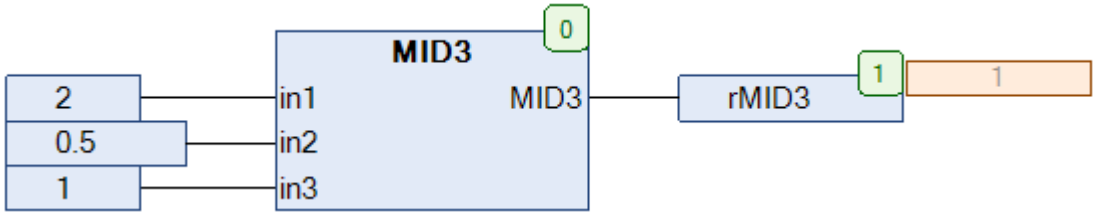


Рис. 5.88. Пример работы с функцией **MID3** на языке CFC

5.45. MIN3

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	in1	REAL	Сравниваемое значение 1.
	in2	REAL	Сравниваемое значение 2.
	in3	REAL	Сравниваемое значение 3.
Выходы	MIN3	REAL	Наименьшее из сравниваемых значений.

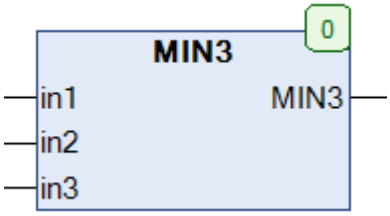


Рис. 5.89. Внешний вид функции **MIN3** на языке CFC

Функция **MIN3** возвращает наименьшее из трех сравниваемых значений. Согласно стандарту МЭК 61131-3 оператор **MIN()** должен иметь настраиваемое число входов, но в некоторых средах разработки их число ограничено двумя – по этой причине в библиотеку включена функция **MIN3**.

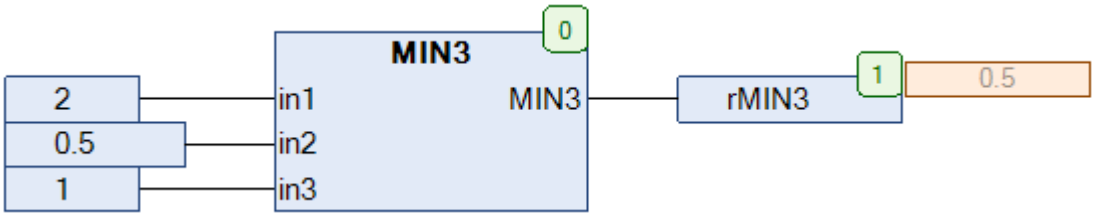


Рис. 5.90. Пример работы с функцией **MIN3** на языке CFC

5.46. MODR

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	IN	REAL	Делимое.
	DIVI	REAL	Делитель.
Выходы	MODR	REAL	Остаток.
Используемые модули	FLOOR2		

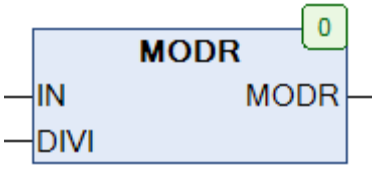


Рис. 5.91. Внешний вид функции **MODR** на языке CFC

Функция **MODR** возвращает остаток от деления значения **IN** на значение **DIVI**. В отличие от стандартного оператора **MOD()**, в качестве аргументов используются переменные типа **REAL.** При **DIVI=0** функция возвращает **0**.

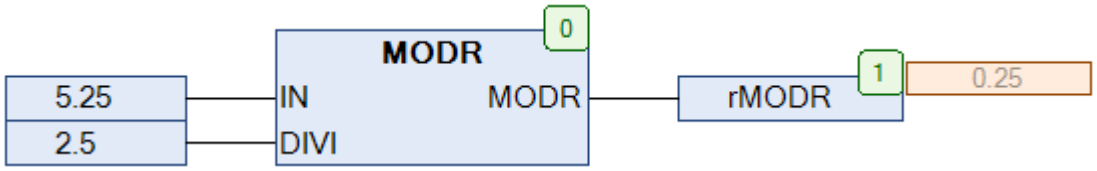


Рис. 5.92. Пример работы с функцией **MODR** на языке CFC

5.47. MUL_ADD

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	X	REAL	Входное значение.
	K	REAL	Коэффициент масштабирования.
	O	REAL	Сдвиг.
Выходы	MUL_ADD	REAL	Отмасштабированное значение.

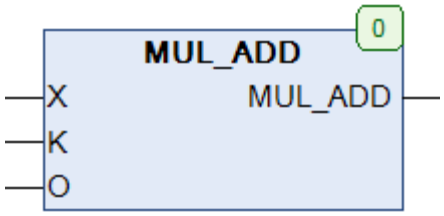


Рис. 5.93. Внешний вид функции **MUL_ADD** на языке CFC

Функция **MUL_ADD** возвращает значение [линейной функции](#) с коэффициентом **K** и сдвигом **O** для аргумента **X**.

$$\mathrm{MUL\_ADD} = \mathrm{K} \cdot \mathrm{X} + \mathrm{O}$$

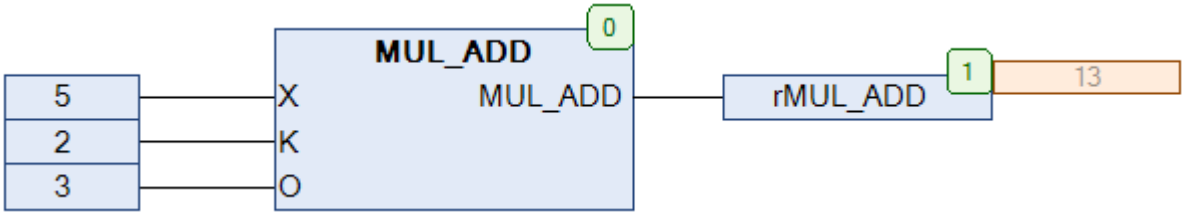


Рис. 5.94. Пример работы с функцией **MUL_ADD** на языке CFC

5.48. NEGX

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	X	REAL	Аргумент.
Выходы	NEGX	REAL	Значение аргумента с противоположным знаком.

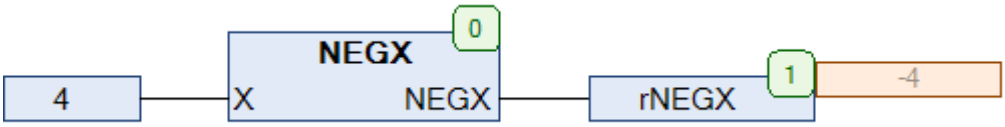


Рис. 5.95. Внешний вид функции **NEGX** на языке CFC

Функция **NEGX** возвращает значение аргумента с противоположным знаком.

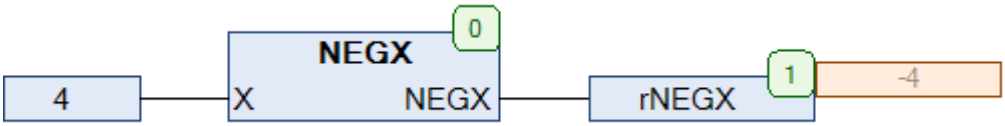


Рис. 5.96. Пример работы с функцией **NEGX** на языке CFC

5.49. RAD

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	DEG	REAL	Значение угла в градусах.
Выходы	RAD	REAL	Значение угла в радианах.
Используемые модули	MODR, DEG		

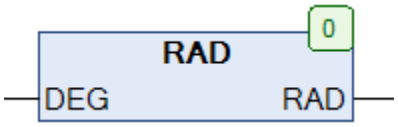


Рис. 5.97. Внешний вид функции **RAD** на языке CFC

Функция **RAD** преобразует значение угла в градусах в радианы. Если входная переменная **DEG** превышает 2π, то из нее будет вычитаться по π до тех пор, пока ее значение не окажется в интервале $0 < \mathrm{DEG} < 2\pi$.

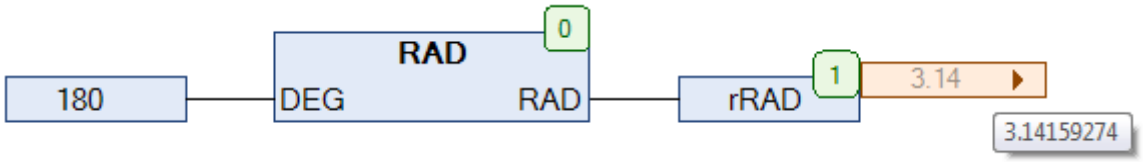


Рис. 5.98. Пример работы с функцией **RAD** на языке CFC

5.50. RDM

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	last	REAL	Начальное значение (зерно).
Выходы	RDM	REAL	Псевдослучайное число.
Используемые модули	T_PLC_MS, BIT_COUNT, FRACT		

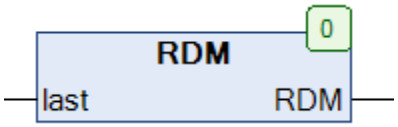


Рис. 5.99. Внешний вид функции **RDM** на языке CFC

Функция **RDM** используется для генерации псевдослучайных чисел с плавающей точкой в диапазоне **[0,1).** В качестве источника энтропии используется системный таймер ПЛК. Поскольку модуль **RDM** представляет собой функцию, а не ФБ, он не содержит информацию о своем предыдущем вызове, и поэтому должен использоваться с осторожностью. Если функция вызывается более одного раза в течение цикла, то возвращает одинаковые числа, т.к. значение системного таймера остается тем же. Этого можно избежать, вызывая функцию с разным значением зерна **last**. В качестве зерна можно использовать значение функции, возвращенное при предыдущем вызове.

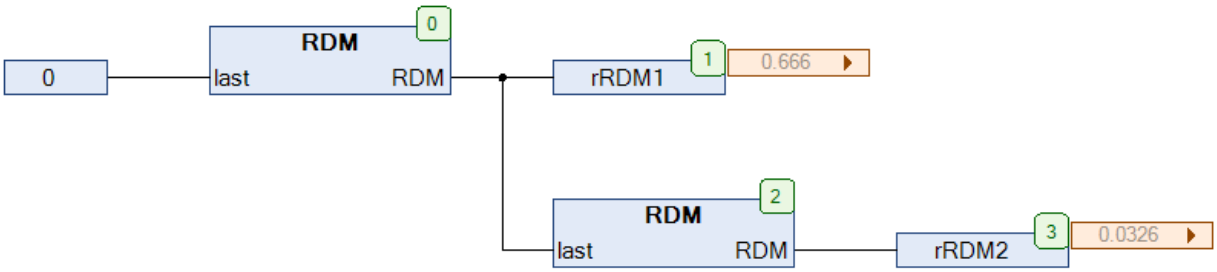


Рис. 5.100. Пример работы с функцией **RDM** на языке CFC

На рис. 5.101 приведена трассировка для 100 циклов программы с рис. 5.100, выполняемой в задаче с временем цикла = 1 с.

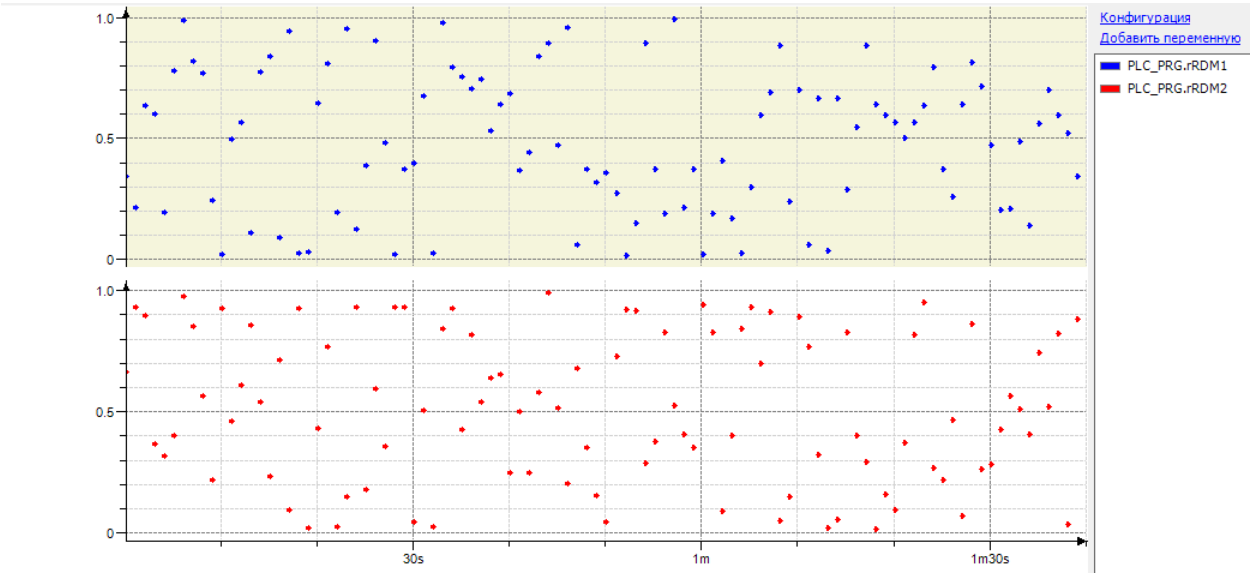


Рис. 5.101. Трассировка для 100 циклов программы с рис. 5.100, время цикла = 1 с

5.51. RDM2

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	last	INT	Начальное значение (зерно).
	low	INT	Нижний предел.
	high	INT	Верхний предел.
Выходы	RDM2	REAL	Псевдослучайное число.
Используемые модули	RDM, FRACT		

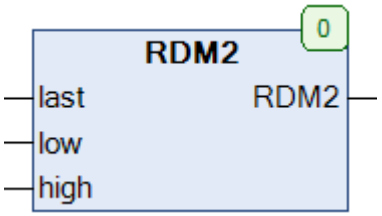


Рис. 5.102. Внешний вид функции **RDM2** на языке CFC

Функция **RDM2** используется для генерации псевдослучайных целых чисел в диапазоне **[low,high].** В качестве источника энтропии используется системный таймер ПЛК. Поскольку модуль **RDM2** представляет собой функцию, а не ФБ, он не содержит информацию о своем предыдущем вызове, и поэтому должен использоваться с осторожностью. Если функция вызывается более одного раза в течение цикла, то возвращает одинаковые числа, т.к. значение системного таймера остается тем же. Этого можно избежать, вызывая функцию с разным значением зерна **last**. В качестве зерна можно использовать значение функции, возвращенное при предыдущем вызове.

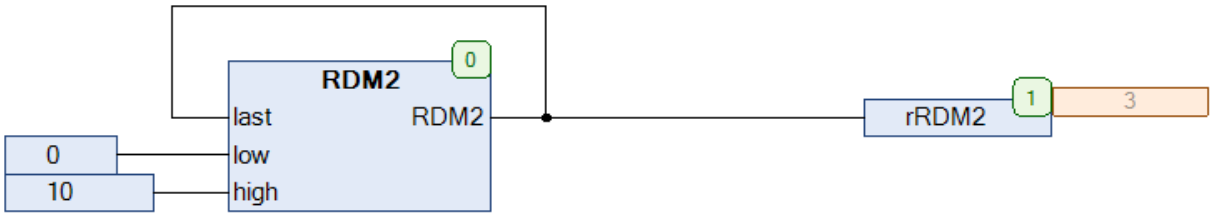


Рис. 5.103. Пример работы с функцией RDM на языке CFC

На рис. 5.104 приведена трассировка для 100 циклов программы с рис. 5.103, выполняемой в задаче с временем цикла = 1 с.

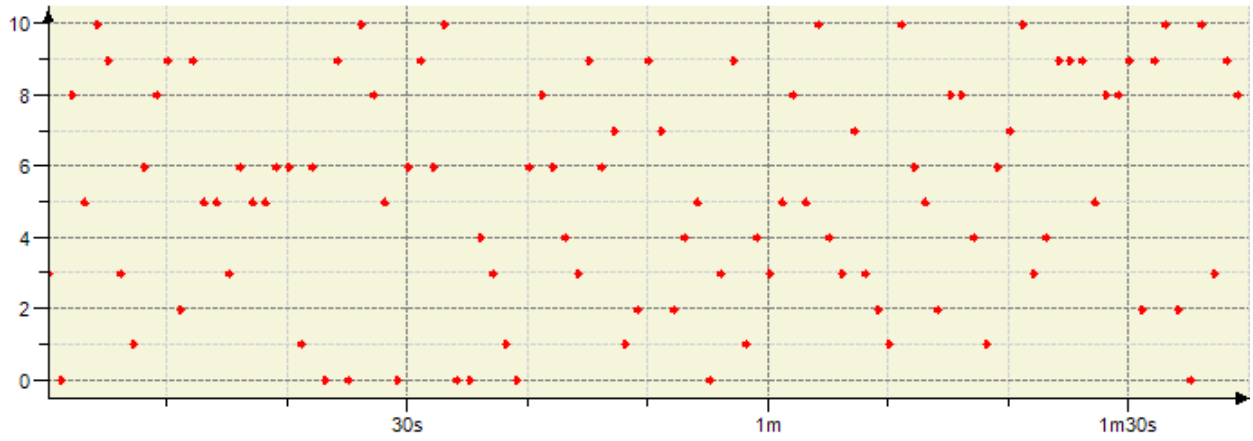


Рис. 5.104. Трассировка для 100 циклов программы с рис. 5.100, время цикла = 1 с

5.52. RDMDW

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	last	REAL	Начальное значение (зерно).
Выходы	RDMDW	DWORD	Псевдослучайный набор бит.
Используемые модули	T_PLC_MS, RDM, BIT_COUNT, FRACT		

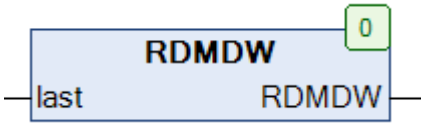


Рис. 5.105. Внешний вид функции **RDMDW** на языке CFC

Функция **RDMDW** используется для генерации псевдослучайных 32-битных наборов бит. В качестве источника энтропии используется системный таймер ПЛК. Поскольку модуль **RDMDW** представляет собой функцию, а не ФБ, он не содержит информацию о своем предыдущем вызове, и поэтому должен использоваться с осторожностью. Если функция вызывается более одного раза в течение цикла, то возвращает одинаковые числа, т.к. значение системного таймера остается тем же. Этого можно избежать, вызывая функцию с разным значением зерна **last**. В качестве зерна можно использовать значение функции, возвращенное при предыдущем вызове.

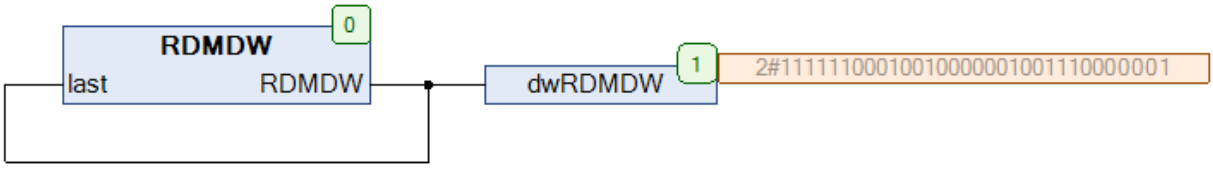


Рис. 5.106. Пример работы с функцией **RDMDW** на языке CFC

5.53. REAL_TO_FRAC

Тип модуля: функция	Переменная	Тип	Описание
Входы	X	REAL	Число с плавающей точкой.
	N	INT	Максимальное значение знаменателя.
Выходы	REAL_TO_FRAC	FRACTION	Число с плавающей точкой в виде дроби.

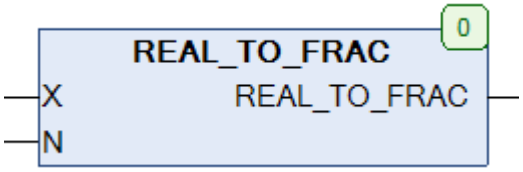


Рис. 5.107. Внешний вид функции **REAL_TO_FRAC** на языке CFC

Функция **REAL_TO_FRAC** представляет значение переменной **X** типа REAL в виде дроби, знаменатель которой не превышает **N**. Возвращаемое значение представляет собой структуру типа FRACTION.

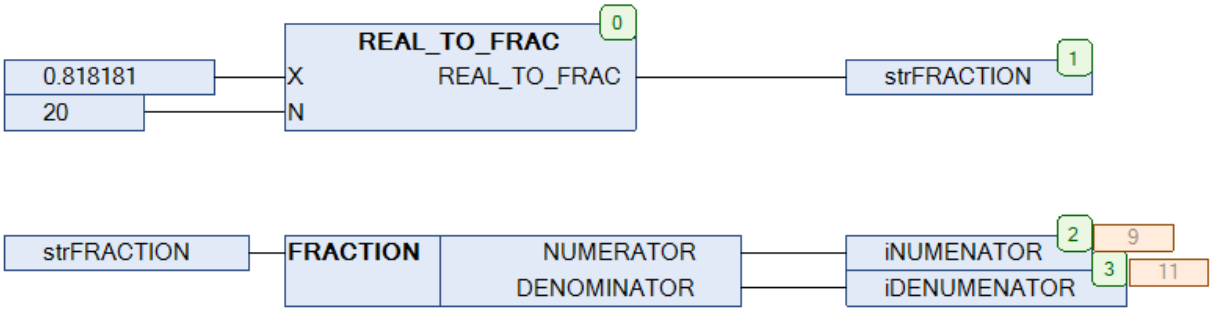


Рис. 5.108. Пример работы с функцией **REAL_TO_FRAC** на языке CFC

5.54. RND

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	X	REAL	Округляемое значение.
	N	INT	Число значащих цифр.
Выходы	RND	REAL	Округленное значение.
Используемые модули	EXPN, CEIL		

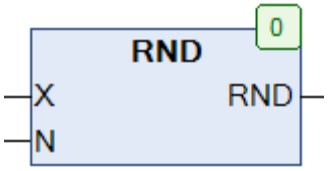


Рис. 5.109. Внешний вид функции **RND** на языке CFC

Функция **RND** возвращает число с плавающей точкой **X**, округленное до **N** значащих цифр. Для чисел 0-4 округление выполняется в меньшую сторону, для 5-9 – в большую. Для округления до заданного количества десятичных знаков может использоваться функция ROUND.

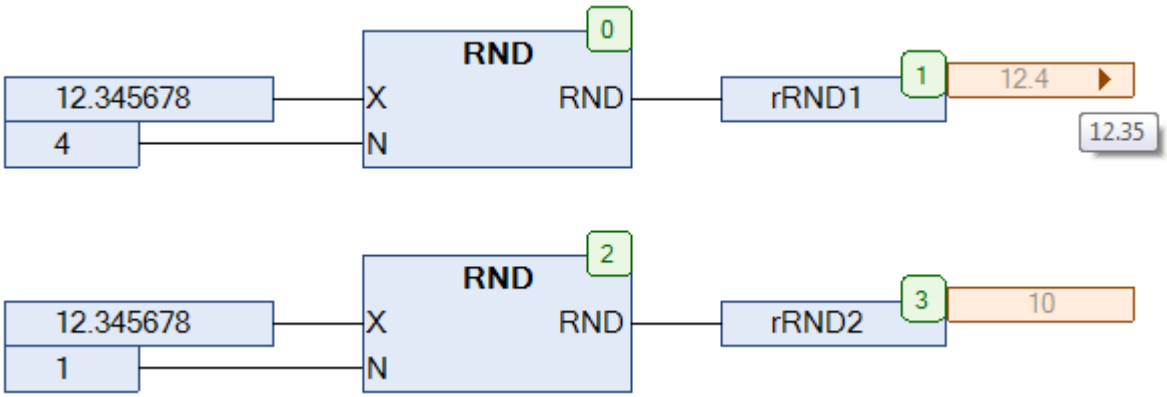


Рис. 5.110. Пример работы с функцией **RND** на языке CFC

5.55. ROUND

Тип модуля: функция	Переменная	Тип	Описание
Входы	in	REAL	Округляемое значение.
	N	INT	Число знаков после запятой.
Выходы	ROUND	REAL	Округленное значение.

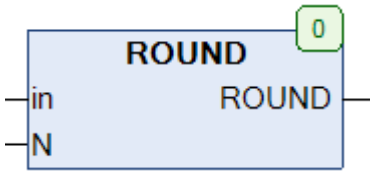


Рис. 5.111. Внешний вид функции **ROUND** на языке CFC

Функция **ROUND** возвращает число с плавающей точкой **X**, округленное до **N** знаков после запятой. Для чисел 0-4 округление выполняется в меньшую сторону, для 5-9 – в большую. Для округления до заданного количества значащих цифр может использоваться функция RND.

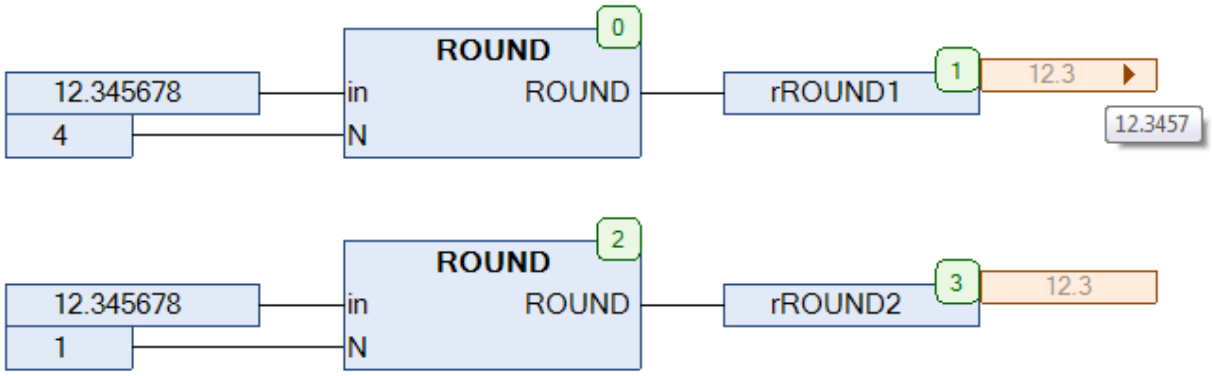


Рис. 5.112. Пример работы с функцией **ROUND** на языке CFC

5.56. SGN

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	X	REAL	Анализируемое значение.
Выходы	SGN	INT	Код знака.

Рис. 5.113. Внешний вид функции **SGN** на языке CFC

Функция **SGN** возвращает код знака переменной **X**.

$$\begin{cases} \mathrm{SGN(X)} = 1 \text{ при } X > 0 \\ \mathrm{SGN(X)} = 0 \text{ при } X = 0 \\ \mathrm{SGN(X)} = -1 \text{ при } X < 0 \end{cases}$$

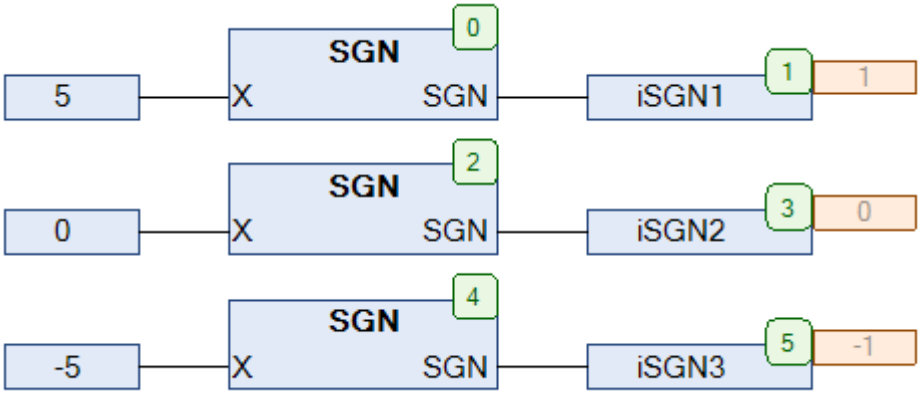


Рис. 5.114. Пример работы с функцией **SGN** на языке CFC

5.57. SIGMOID

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	X	REAL	Аргумент.
Выходы	SIGMOID	REAL	Значение сигмоиды.

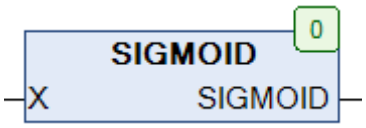


Рис. 5.115. Внешний вид функции **SIGMOID** на языке CFC

Функция SIGMOID возвращает значение сигмоиды, вычисленное по формуле

$$\mathrm{SIGMOID}(X) = \frac{1}{1 + e^{-X}}$$

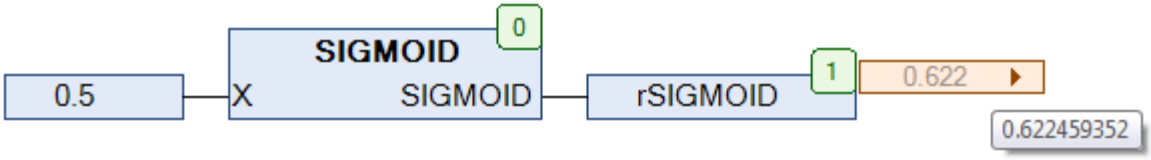


Рис. 5.116. Пример работы с функцией **SIGMOID** на языке CFC

5.58. SIGN_I

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	in	DINT	Анализируемое значение.
Выходы	SIGN_I	BOOL	Флаг знака.

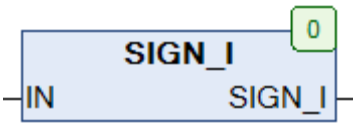


Рис. 5.117. Внешний вид функции **SIGN_I** на языке CFC

Функция **SIGN_I** возвращает **TRUE,** если значение переменной **in** типа DINT является отрицательным, и **FALSE** – если положительным.

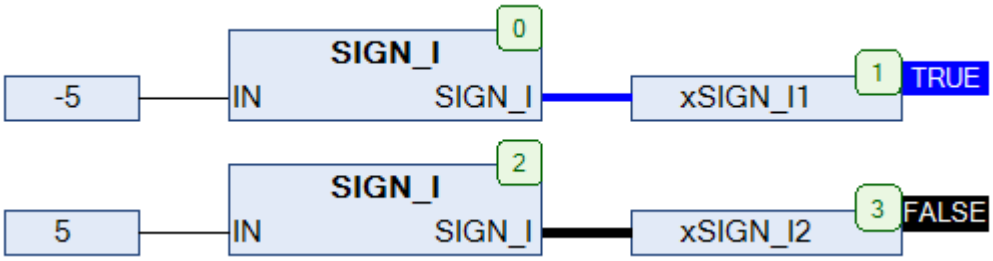


Рис. 5.118. Пример работы с функцией **SIGN_I** на языке CFC

5.59. SIGN_R

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	in	REAL	Анализируемое значение.
Выходы	SIGN_R	BOOL	Флаг знака.

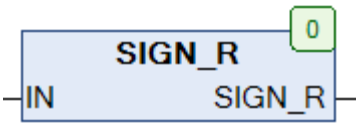


Рис. 5.119. Внешний вид функции **SIGN_R** на языке CFC

Функция **SIGN_R** возвращает **TRUE**, если значение переменной **in** типа REAL является отрицательным, и **FALSE** – если положительным.

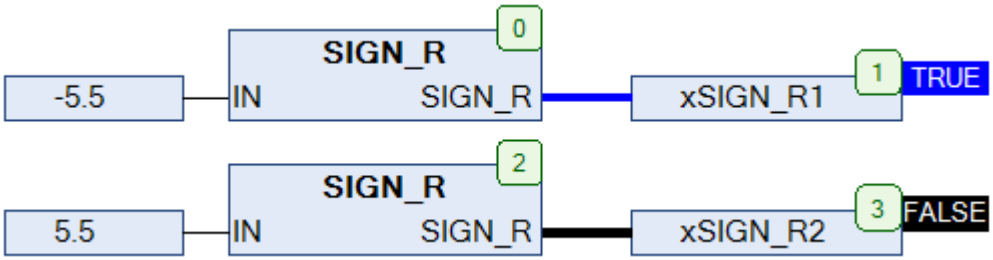


Рис. 5.120. Пример работы с функцией **SIGN_R** на языке CFC

5.60. SINC

Тип модуля: функция	Переменная	Тип	Описание
Входы	X	REAL	Аргумент.
Выходы	SINC	REAL	Значение кардинального синуса.

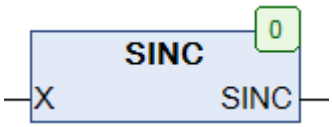


Рис. 5.121. Внешний вид функции **SINC** на языке CFC

Функция **SINC** возвращает значение [ненормированной функции кардинального синуса](), вычисленное по формуле

$$\begin{cases} \mathrm{SINC}(X) = \dfrac{\mathrm{SIN}(X)}{X} & X \neq 0 \\ 1 & X = 0 \end{cases}$$

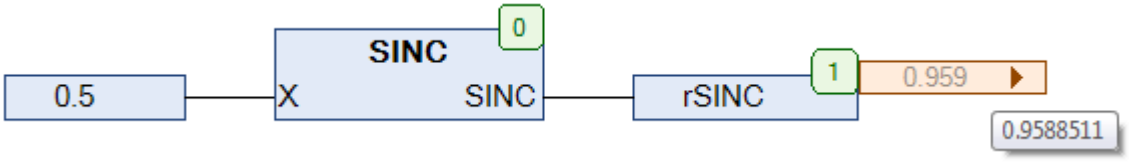


Рис. 5.122. Пример работы с функцией **SINC** на языке CFC

5.61. SINH

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	X	REAL	Аргумент.
Выходы	SINH	REAL	Значение гиперболического синуса.

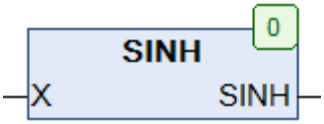


Рис. 5.123. Внешний вид функции **SINH** на языке CFC

Функция **SINH** возвращает значение гиперболического синуса, вычисленное по формуле

$$\mathrm{SINH}(X) = \frac{e^{X} - e^{-X}}{2}$$

Область определения: $-\infty < X < +\infty$

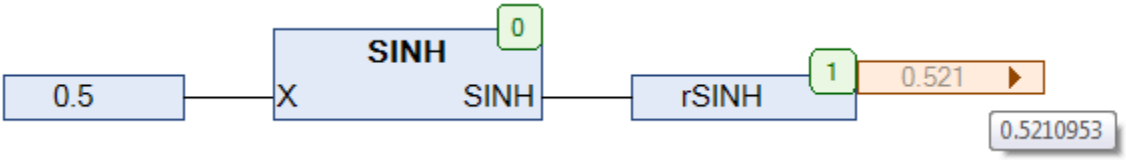


Рис. 5.124. Пример работы с функцией **SINH** на языке CFC

5.62. SQRTN

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	X	REAL	Аргумент.
	N	REAL	Степень корня.
Выходы	SQRTN	REAL	Корень N-й степени.

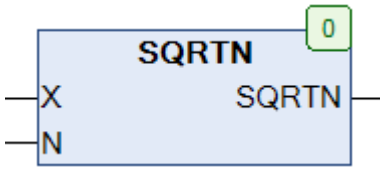


Рис. 5.125. Внешний вид функции **SQRTN** на языке CFC

Функция **SQRTN** возвращает значение корня **N**-й степени из **X**.

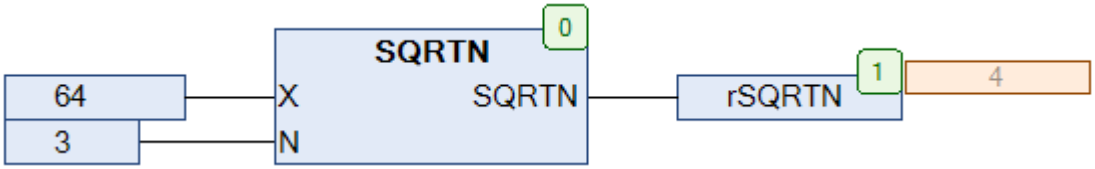


Рис. 5.126. Пример работы с функцией **SQRTN** на языке CFC

5.63. TANC

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	X	REAL	Аргумент.
Выходы	TANC	REAL	Значение функции.

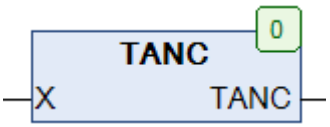


Рис. 5.127. Внешний вид функции **TANC** на языке CFC

Функция **TANC** возвращает значение одноименной математической функции, вычисленное по формуле

$$\begin{cases} \mathrm{TANC}(X) = \dfrac{\mathrm{TANC}(X)}{X} & X \neq 0 \\ 1 & X = 0 \end{cases}$$

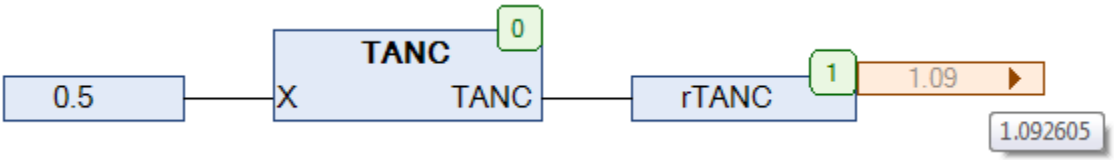


Рис. 5.128. Пример работы с функцией **TANC** на языке CFC

5.64. TANH

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	X	REAL	Аргумент.
Выходы	TANH	REAL	Значение гиперболического тангенса.

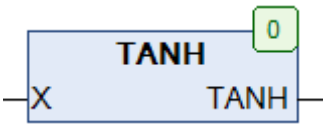


Рис. 5.129. Внешний вид функции **TANH** на языке CFC

Функция **TANH** возвращает значение гиперболического тангенса, вычисленное по формуле

$$\mathrm{TANH}(\mathrm{X}) = 1 - \frac{2}{\mathrm{e}^{2\mathrm{X}} + 1}$$

Область определения: $-\infty < \mathrm{X} < +\infty$

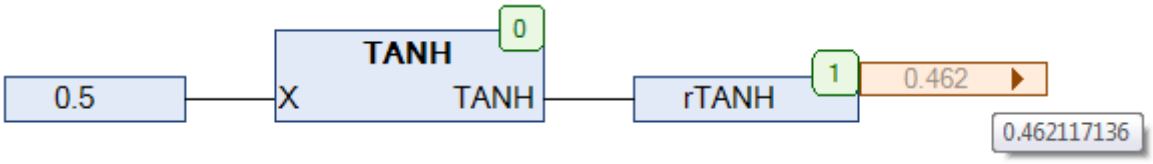


Рис. 5.130. Пример работы с функцией **TANH** на языке CFC

5.65. WINDOW

Тип модуля: функция	Переменная	Тип	Описание
Входы	low	REAL	Нижний предел.
	in	REAL	Анализируемое значение.
	high	REAL	Верхний предел.
Выходы	WINDOW	BOOL	Флаг принадлежности интервалу.

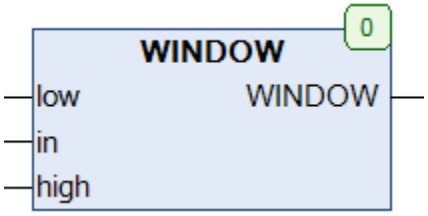


Рис. 5.131. Внешний вид функции **WINDOW** на языке CFC

Функция **WINDOW** возвращает **TRUE**, если значение **X** принадлежит интервалу **(low, high).** В противном случае функция возвращает **FALSE**.

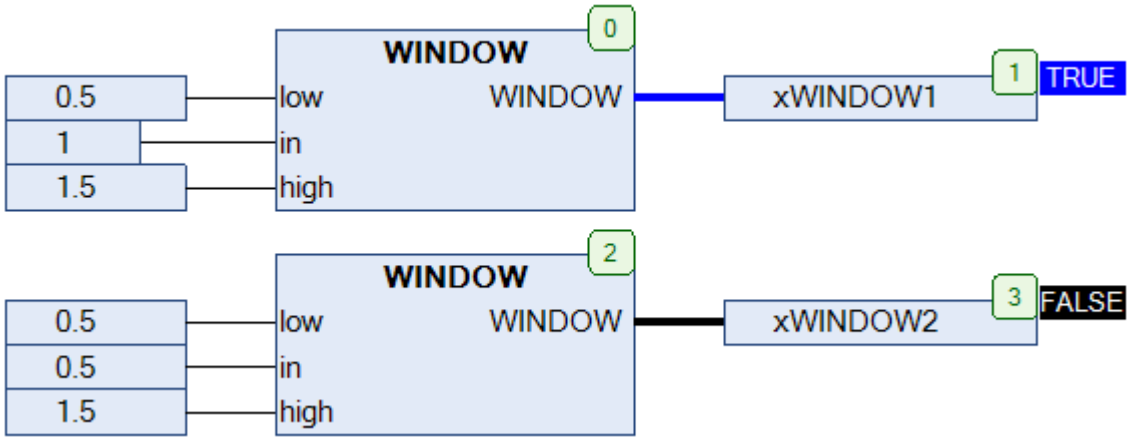


Рис. 5.132. Пример работы с функцией **WINDOW** на языке CFC

5.65. WINDOW2

Тип модуля: функция	Переменная	Тип	Описание
Входы	low	REAL	Нижний предел.
	in	REAL	Анализируемое значение.
	high	REAL	Верхний предел.
Выходы	WINDOW2	BOOL	Флаг принадлежности интервалу.

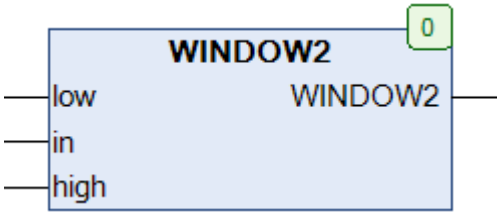


Рис. 5.133. Внешний вид функции **WINDOW2** на языке CFC

Функция **WINDOW2** возвращает **TRUE**, если значение **X** принадлежит интервалу **[low, high].** В противном случае функция возвращает **FALSE**.

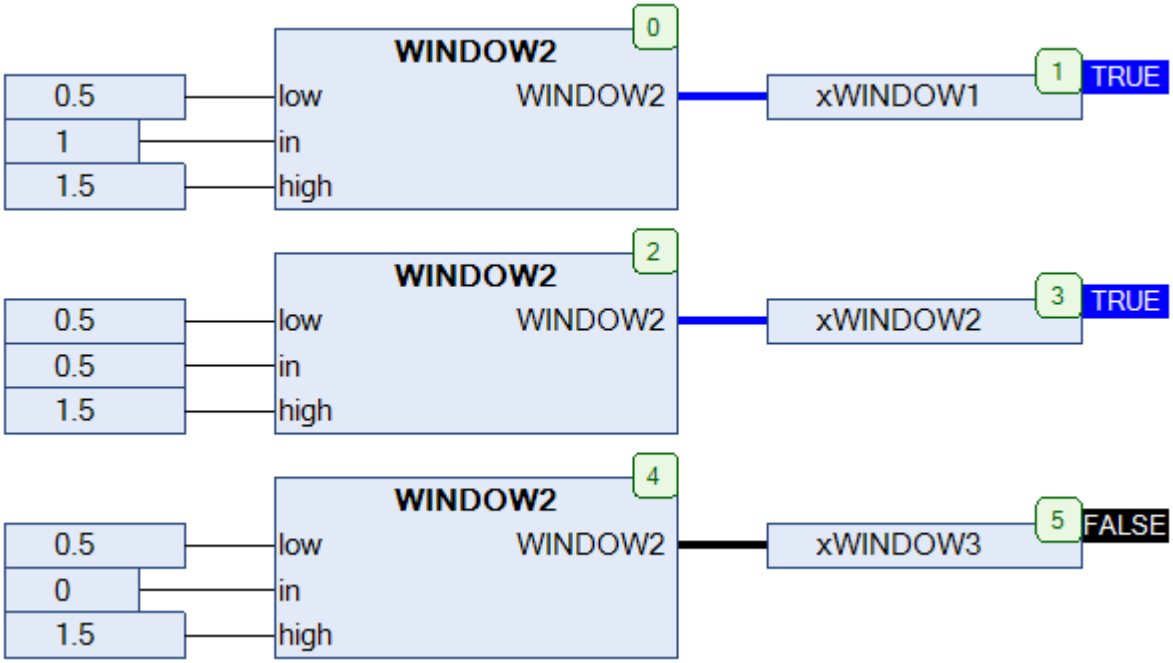


Рис. 5.134. Пример работы с функцией **WINDOW2** на языке CFC

6. Работа с массивами

6.0. Вступление

Функции, описанные в данной главе, используются для операций над массивами значений типа **REAL**. Входными переменными для каждой функции являются указатель на массив (**pt**) и размер массива (**size**). В большинстве случаев представляется удобным использовать операторы **ADR()** и **SIZEOF(),** которые возвращают соответственно адрес и размер массива в байтах. Тогда вызов функции для обработки массива **aArr** можно представить следующим образом:

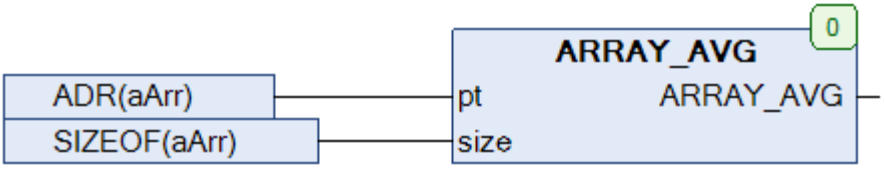


Рис. 6.1. Вызов функции работы с массивами на языке CFC

Максимальный размер обрабатываемого массива: **ARRAY [0..32000] OF REAL**.

Обработка массива, полученного по указателю, производится путем прямых манипуляций с памятью ПЛК. Этот тип обработки является крайне эффективным, так как не требует копирования содержимого массива.

6.1. _ARRAY_ABS

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	pt	POINTER TO ARRAY [...] OF REAL	Указатель на массив.
	size	UINT	Размер массива.
Выходы	_ARRAY_ABS	BOOL	Флаг окончания обработки массива.

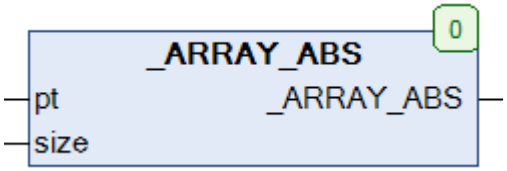


Рис. 6.2. Внешний вид функции **_ARRAY_ABS** на языке CFC

Функция **_ARRAY_ABS** вычисляет модули элементов входного массива. Результаты записываются в тот же массив по указателю. После окончания обработки выход функции принимает значение **TRUE.**

Пример: массив [-2, 12, -4, 8, 6] после обработки функцией примет вид [2, 12, 4, 8, 6].

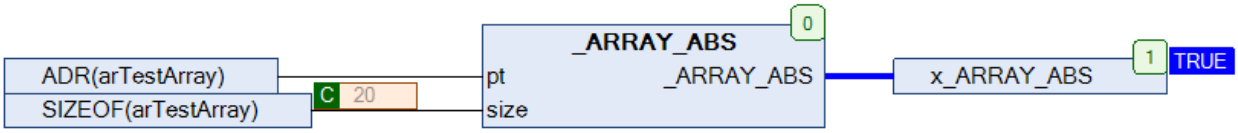


Рис. 6.3. Пример работы с функцией **_ARRAY_ABS** на языке CFC

6.2. _ARRAY_ADD

Тип модуля: функция	Переменная	Тип	Описание
Входы	pt	POINTER TO ARRAY [...] OF REAL	Указатель на массив.
	size	UINT	Размер массива.
	X	REAL	Прибавляемое значение.
Выходы	_ARRAY_ADD	BOOL	Флаг окончания обработки массива.

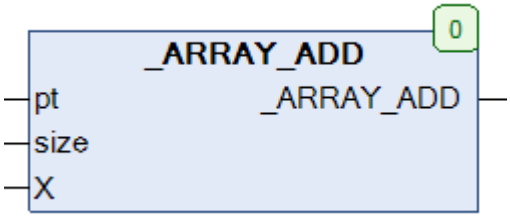


Рис. 6.4. Внешний вид функции **_ARRAY_ADD** на языке CFC

Функция **_ARRAY_ADD** увеличивает значение каждого из элементов входного массива на заданное значение **X**. Результаты записываются в тот же массив по указателю. После окончания обработки выход функции принимает значение **TRUE**.

Пример: массив [-2, 12, -4, 8, 6] при **X=10** после однократной обработки функцией примет вид [8, 22, 6, 18, 16].

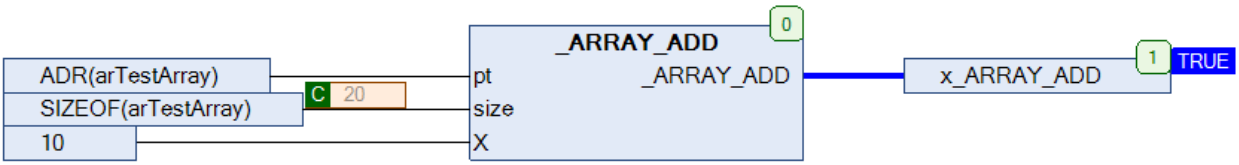


Рис. 6.5. Пример работы с функцией **_ARRAY_ADD** на языке CFC

6.3. _ARRAY_INIT

Тип модуля: функция	Переменная	Тип	Описание
Входы	pt	POINTER TO ARRAY [...] OF REAL	Указатель на массив.
	size	UINT	Размер массива.
	init	REAL	Присваиваемое значение.
Выходы	_ARRAY_INIT	BOOL	Флаг окончания обработки массива.

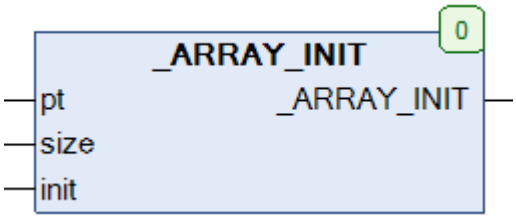


Рис. 6.6. Внешний вид функции **_ARRAY_INIT** на языке CFC

Функция **_ARRAY_INIT** присваивает каждому из элементов входного массива заданное значение **init**. Результаты записываются в тот же массив по указателю. После окончания обработки выход функции принимает значение **TRUE**.

Пример: массив [-2, 12, -4, 8, 6] при **init=0** после обработки функцией примет вид [0, 0, 0, 0, 0].

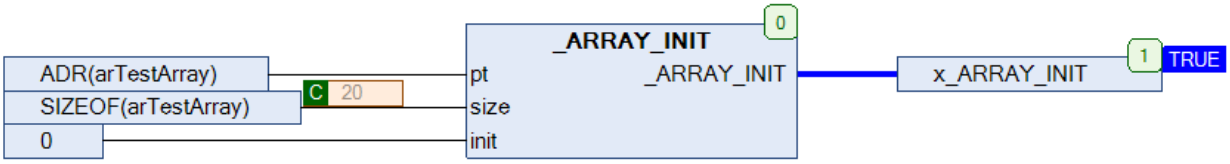


Рис. 6.7. Пример работы с функцией **_ARRAY_INIT** на языке CFC

6.4. _ARRAY_MEDIAN

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	pt	POINTER TO ARRAY [...] OF REAL	Указатель на массив.
	size	UINT	Размер массива.
Выходы	_ARRAY_MEDIAN	REAL	Медиана массива.
Используемые модули	EVEN		

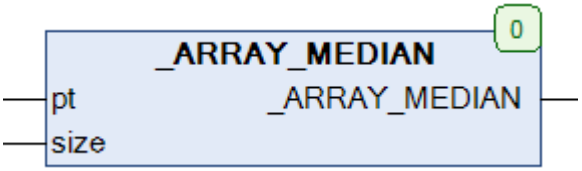


Рис. 6.8. Внешний вид функции **_ARRAY_MEDIAN** на языке CFC

Функция **_ARRAY_MEDIAN** возвращает медиану массива, а также сортирует его по возрастанию.

Пример: массив [-2, 12, -4, 8, 6] после обработки функцией примет вид [-4, -2, 6, 8, 12].

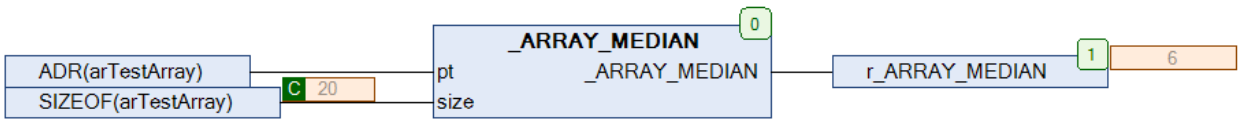


Рис. 6.9. Пример работы с функцией **_ARRAY_MEDIAN** на языке CFC

6.5. _ARRAY_MUL

Тип модуля: функция	Переменная	Тип	Описание
Входы	pt	POINTER TO ARRAY [...] OF REAL	Указатель на массив.
	size	UINT	Размер массива.
	X	REAL	Множитель.
Выходы	_ARRAY_MUL	BOOL	Флаг окончания обработки массива.

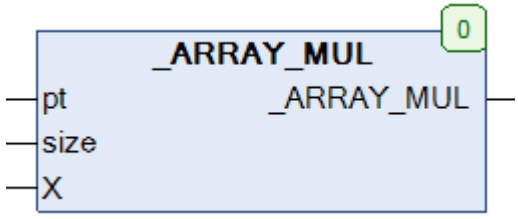


Рис. 6.10. Внешний вид функции _ARRAY_MUL на языке CFC

Функция **_ARRAY_MUL** умножает значение каждого из элементов входного массива на заданное значение **X**. Результаты записываются в тот же массив по указателю. После окончания обработки выход функции принимает значение **TRUE**.

Пример: массив [-2, 12, -4, 8, 6] при **X=3** после однократной обработки функцией примет вид [-6, 36, -12, 24, 18].

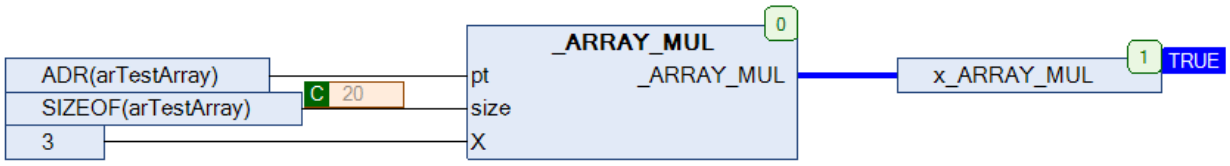


Рис. 6.11. Пример работы с функцией **_ARRAY_MUL** на языке CFC

6.6. _ARRAY_SHUFFLE

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	pt	POINTER TO ARRAY […] OF REAL	Указатель на массив.
	size	UINT	Размер массива.
Выходы	_ARRAY_SHUFFLE	BOOL	Флаг окончания обработки массива.
Используемые модули	RDM2		

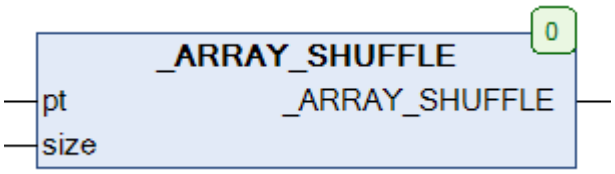


Рис. 6.12. Внешний вид функции **_ARRAY_SHUFFLE** на языке CFC

Функция **_ARRAY_SHUFFLE** переставляет элементы входного массива местами в случайном порядке. Результаты записываются в тот же массив по указателю. После окончания обработки выход функции принимает значение **TRUE**.

Пример: массив [-2, 12, -4, 8, 6] после однократной обработки функцией в нашем случае принял вид [-6, 36, -12, 24, 18].

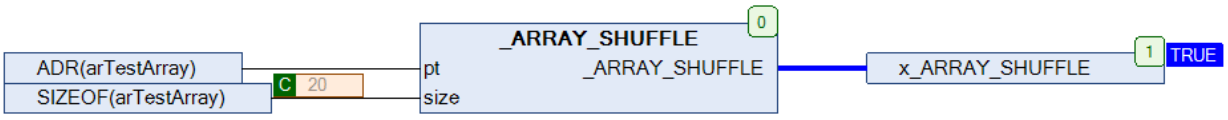


Рис. 6.13. Пример работы с функцией **_ARRAY_SHUFFLE** на языке CFC

6.7. _ARRAY_SORT

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	pt	POINTER TO ARRAY […] OF REAL	Указатель на массив.
	size	UINT	Размер массива.
Выходы	_ARRAY_SORT	BOOL	Флаг окончания обработки массива.

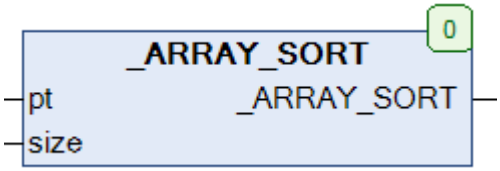


Рис. 6.14. Внешний вид функции **_ARRAY_SORT** на языке CFC

Функция **_ARRAY_SORT** сортирует входной массив, располагая его элементы в порядке возрастания. Результаты записываются в тот же массив по указателю. После окончания обработки выход функции принимает значение **TRUE**.

Пример: массив [-2, 12, -4, 8, 6] после обработки функцией примет вид [-4, -2, 6, 8, 12].

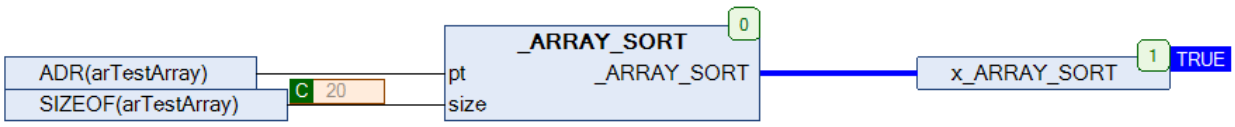


Рис. 6.15. Пример работы с функцией **_ARRAY_SORT** на языке CFC

6.8. ARRAY_AVG

Тип модуля: функция	Переменная	Тип	Описание
Входы	pt	POINTER TO ARRAY […] OF REAL	Указатель на массив.
	size	UINT	Размер массива.
Выходы	ARRAY_AVG	REAL	Среднее арифметическое массива.

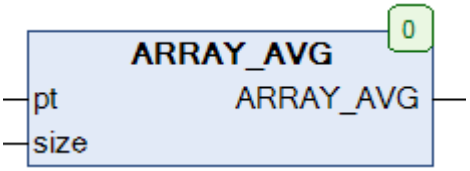


Рис. 6.16. Внешний вид функции **ARRAY_AVG** на языке CFC

Функция **ARRAY_AVG** возвращает среднее арифметическое входного массива. Функция не изменяет содержимое входное массива.

Пример: для массива [-2, 12, -4, 8, 6] функция вернет значение **4**.

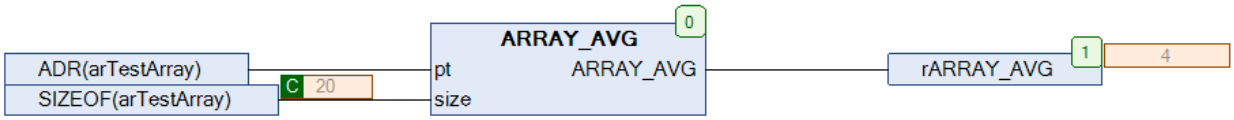


Рис. 6.17. Пример работы с функцией **ARRAY_AVG** на языке CFC

6.9. ARRAY_GAV

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	pt	POINTER TO ARRAY […] OF REAL	Указатель на массив.
	size	UINT	Размер массива.
Выходы	ARRAY_GAV	REAL	Среднее геометрическое массива.
Используемые модули	SQRTN		

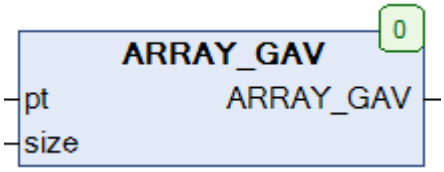


Рис. 6.18. Внешний вид функции **ARRAY_GAV** на языке CFC

Функция **ARRAY_GAV** возвращает среднее геометрическое входного массива. Функция не изменяет содержимое входное массива.

Пример: для массива [2, 12, 4, 8, 6] функция вернет значение **5.40383959**.

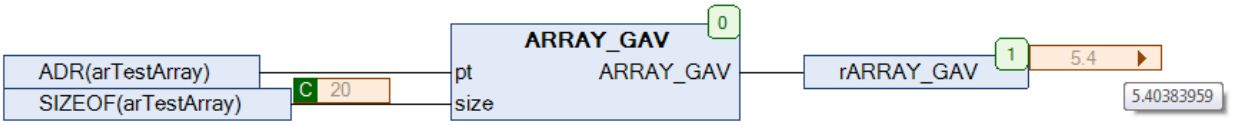


Рис. 6.19. Пример работы с функцией **ARRAY_GAV** на языке CFC

6.10. ARRAY_HAV

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	pt	POINTER TO ARRAY […] OF REAL	Указатель на массив.
	size	UINT	Размер массива.
Выходы	ARRAY_HAV	REAL	Среднее гармоническое массива.

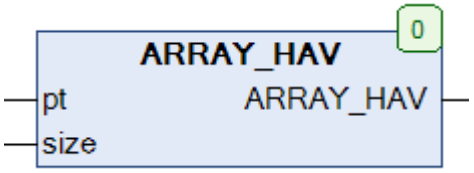


Рис. 6.20. Внешний вид функции **ARRAY_HAV** на языке CFC

Функция **ARRAY_HAV** возвращает среднее гармоническое входного массива. Функция не изменяет содержимое массива.

Пример: для массива [-2, 12, -4, 8, 6] функция вернет значение **-13.333334**.

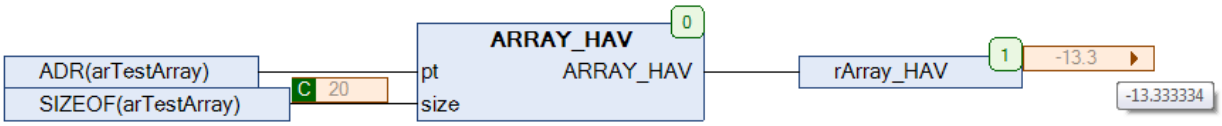


Рис. 6.21. Пример работы с функцией **ARRAY_HAV** на языке CFC

6.11. ARRAY_MIN

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	pt	POINTER TO ARRAY […] OF REAL	Указатель на массив.
	size	UINT	Размер массива.
Выходы	ARRAY_MIN	REAL	Значение наименьшего элемента.

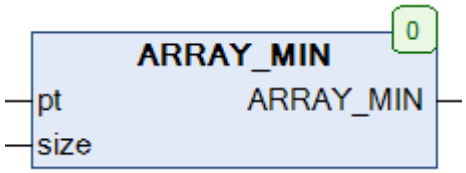


Рис. 6.22. Внешний вид функции **ARRAY_MIN** на языке CFC

Функция **ARRAY_MIN** возвращает значение наименьшего элемента входного массива. Функция не изменяет содержимое массива.

Пример: для массива [-2, 12, -4, 8, 6] функция вернет значение **-4**.

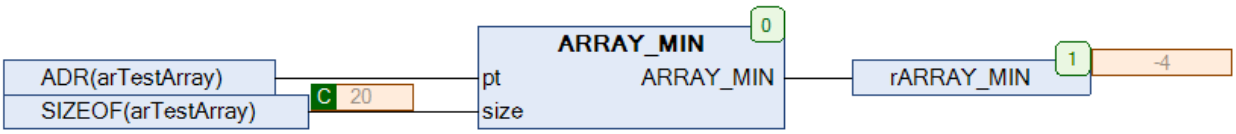


Рис. 6.23. Пример работы с функцией **ARRAY_MIN** на языке CFC

6.12. ARRAY_MAX

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	pt	POINTER TO ARRAY […] OF REAL	Указатель на массив.
	size	UINT	Размер массива.
Выходы	ARRAY_MAX	REAL	Значение наибольшего элемента.

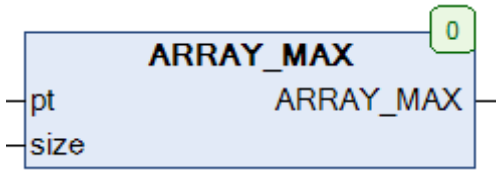


Рис. 6.24. Внешний вид функции **ARRAY_MAX** на языке CFC

Функция **ARRAY_MAX** возвращает значение наибольшего элемента входного массива. Функция не изменяет содержимое массива.

Пример: для массива [-2, 12, -4, 8, 6] функция вернет значение **12**.

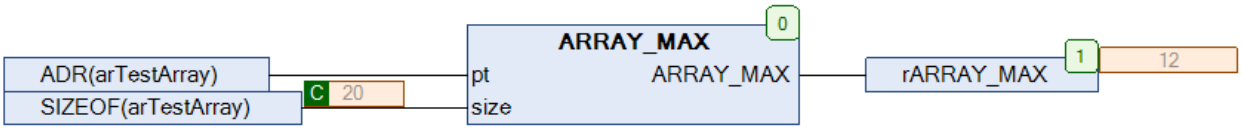


Рис. 6.25. Пример работы с функцией **ARRAY_MAX** на языке CFC

6.13. ARRAY_SDV

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	pt	POINTER TO ARRAY […] OF REAL	Указатель на массив.
	size	UINT	Размер массива.
Выходы	ARRAY_SDV	REAL	Среднеквадратичное отклонение.
Используемые модули	ARRAY_VAR		

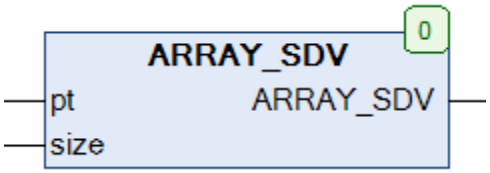


Рис. 6.26. Внешний вид функции **ARRAY_SDV** на языке CFC

Функция **ARRAY_SDV** возвращает среднеквадратичное отклонение для входного массива. Функция не изменяет содержимое массива.

Пример: для массива [-2, 12, -4, 8, 6] функция вернет значение **6.78233**.

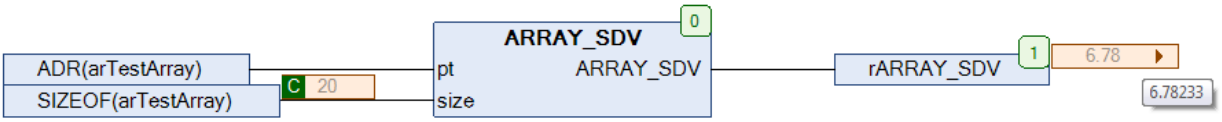


Рис. 6.27. Пример работы с функцией **ARRAY_SDV** на языке CFC

6.14. ARRAY_SPR

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	pt	POINTER TO ARRAY […] OF REAL	Указатель на массив.
	size	UINT	Размер массива.
Выходы	ARRAY_SPR	REAL	Абсолютное отклонение.
Используемые модули	ARRAY_MIN, ARRAY_MAX		

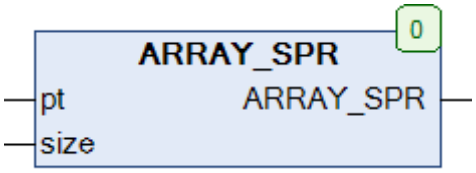


Рис. 6.28. Внешний вид функции **ARRAY_SPR** на языке CFC

Функция **ARRAY_SPR** возвращает абсолютное отклонение для входного массива. Функция не изменяет содержимое массива.

Пример: для массива [-2, 12, -4, 8, 6] функция вернет значение 16 (*полученное следующим образом:* 12 - (-4)).

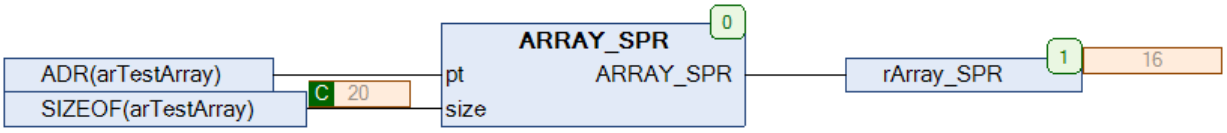


Рис. 6.29. Пример работы с функцией **ARRAY_SPR** на языке CFC

6.15. ARRAY_SUM

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	pt	POINTER TO ARRAY […] OF REAL	Указатель на массив.
	size	UINT	Размер массива.
Выходы	ARRAY_SUM	REAL	Сумма элементов массива.

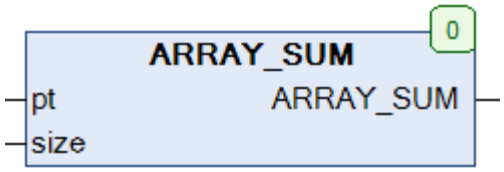


Рис. 6.30. Внешний вид функции **ARRAY_SUM** на языке CFC

Функция **ARRAY_SUM** возвращает сумму элементов входного массива. Функция не изменяет содержимое массива.

Пример: для массива [-2, 12, -4, 8, 6] функция вернет значение **20**.

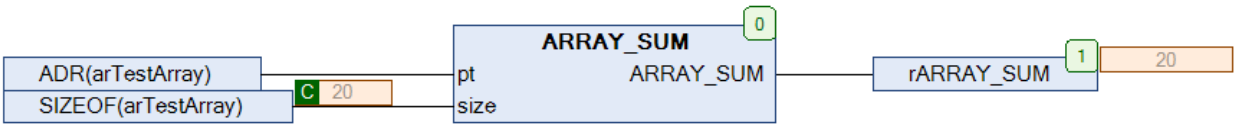


Рис. 6.31. Пример работы с функцией **ARRAY_SUM** на языке CFC

6.16. ARRAY_TREND

Тип модуля: функция	Переменная	Тип	Описание
Входы	pt	POINTER TO ARRAY […] OF REAL	Указатель на массив.
	size	UINT	Размер массива.
Выходы	ARRAY_TREND	REAL	Абсолютное отклонение.
Используемые модули	EVEN		

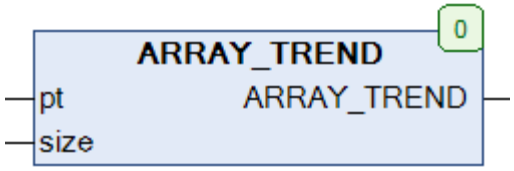


Рис. 6.32. Внешний вид функции **ARRAY_TREND** на языке CFC

Функция **ARRAY_TREND** возвращает значение тренда входного массива. Тренд представляет собой разность между средними значениями второй и первой половин массива. Если количество элементов в массиве является нечетным, то «центральный» элемент включается в обе половины. Функция не изменяет содержимое массива.

Пример: для массива [-2, 12, -4, 8, 6] функция вернет значение **1.33333337.**

$$\frac{-4+8+6}{3}-\frac{-2+12-4}{3}=\frac{4}{3}=1.333(3)$$

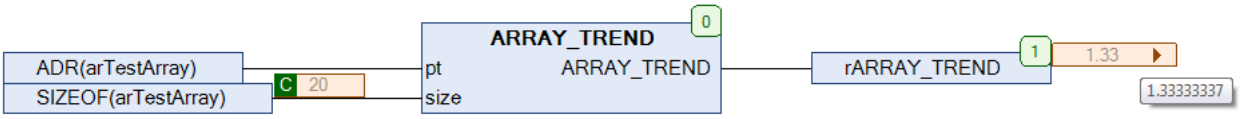


Рис. 6.33. Пример работы с функцией **ARRAY_TREND** на языке CFC

6.17. ARRAY_VAR

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	pt	POINTER TO ARRAY […] OF REAL	Указатель на массив.
	size	UINT	Размер массива.
Выходы	ARRAY_VAR	REAL	Дисперсия.

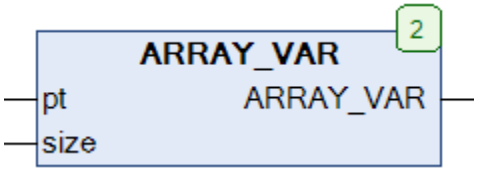


Рис. 6.34. Внешний вид функции **ARRAY_VAR на** языке CFC

Функция **ARRAY_VAR** возвращает значение [дисперсии](#) входного массива. Функция не изменяет содержимое массива.

Пример: для массива [-2, 12, -4, 8, 6] функция вернет значение **46**.

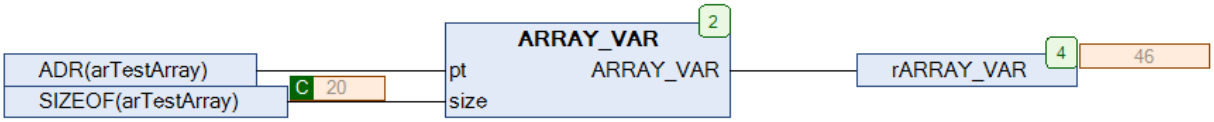


Рис. 6.35. Пример работы с функцией **ARRAY_VAR** на языке CFC

6.18. IS_SORTED

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	pt	POINTER TO ARRAY […] OF REAL	Указатель на массив.
	size	UINT	Размер массива.
Выходы	IS_SORTED	BOOL	Флаг сортировки массива.

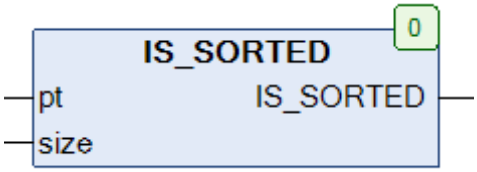


Рис. 6.36. Внешний вид функции **IS_SORTED** на языке CFC

Функция **IS_SORTED** проверяет, отсортированы ли элементы входного массива по возрастанию. Функция возвращает **TRUE**, если массив отсортирован; во всех остальных случаях функция возвращает **FALSE**.

Пример: для массива [-2, 12, -4, 8, 6] функция вернет **FALSE**.

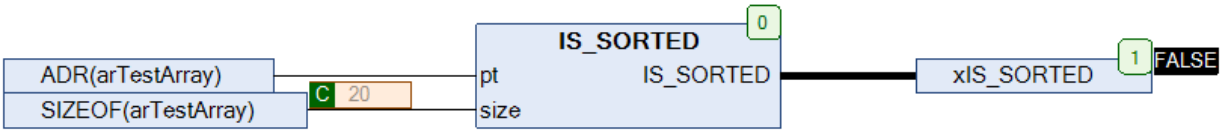


Рис. 6.37. Пример работы с функцией **IS_SORTED** на языке CFC

7. Комплексная арифметика

7.1. Вступление

Комплексным числом называется выражение вида **x + i·y,** где **x** и **y** – любые действительные числа, а **i** – специальное число, которое называется мнимой единицей. Величины **x** и **y** называются **действительной** и **мнимой** часть комплексного числа. Комплексные числа нашли применение в ряде областей математики и физики (в частности, в расчетах цепей переменного тока). В библиотеке **OSCAT** комплексные числа представляют собой экземпляры структуры COMPLEX, которая состоит из двух переменных: **re** (целая часть комплексного числа) и **im** (мнимая часть действительного числа). В данной главе описаны функции, используемые для работы с комплексными числами.

7.2. CABS

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	X	COMPLEX	Комплексное число.
Выходы	CABS	REAL	Модуль комплексного числа.
Используемые модули	HYPOT		

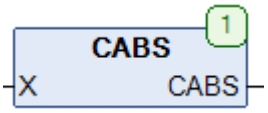


Рис. 7.1. Внешний вид функции **CABS** на языке CFC

Функция **CABS** возвращает модуль комплексного числа X, вычисленный по формуле

$$\mathrm{CABS} = \sqrt{(\mathrm{X.re})^2 + (\mathrm{X.im})^2}$$

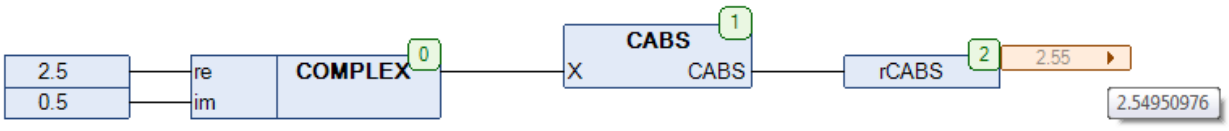


Рис. 7.2. Пример работы с функцией **CABS** на языке CFC

7.3. CACOS

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	X	COMPLEX	Комплексное число.
Выходы	CACOS	COMPLEX	Арккосинус комплексного числа.
Используемые модули	CACOSH		

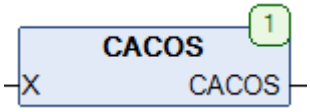


Рис. 7.3. Внешний вид функции **CACOS** на языке CFC

Функция **CACOS** возвращает арккосинус комплексного числа **X**. Действительная часть полученного результата принадлежит интервалу [0, π], мнимая принадлежит интервалу [-∞,∞].

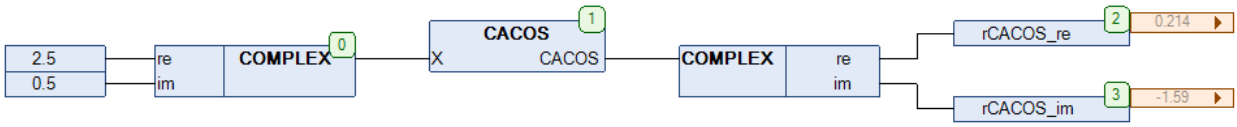


Рис. 7.4. Пример работы с функцией **CACOS** на языке CFC

7.4. CACOSH

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	X	COMPLEX	Комплексное число.
Выходы	CACOSH	COMPLEX	Гиперболический арккосинус комплексного числа.
Используемые модули	CSQRT, CLOG		

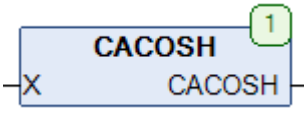


Рис. 7.5. Внешний вид функции **CACOSH** на языке CFC

Функция **CACOSH** возвращает гиперболический арккосинус комплексного числа **X**. Мнимая часть полученного результата принадлежит интервалу $[-\pi, \pi]$.

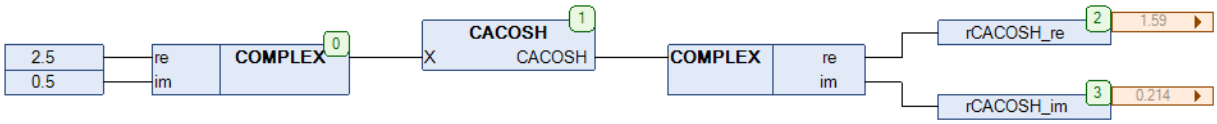


Рис. 7.6. Пример работы с функцией **CACOSH** на языке CFC

7.5. CADD

Тип модуля: функция	Переменная	Тип	Описание
Входы	X	COMPLEX	Комплексное число.
	Y	COMPLEX	Комплексное число.
Выходы	CADD	COMPLEX	Сумма комплексных чисел.

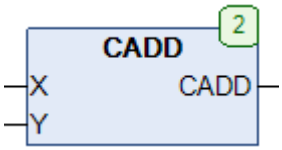


Рис. 7.7. Внешний вид функции **CADD** на языке CFC

Функция **CADD** возвращает сумму комплексных чисел **X** и **Y**.

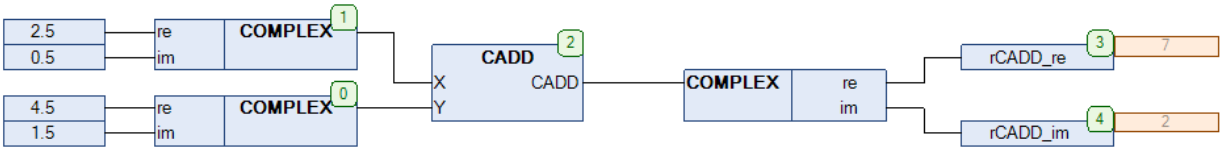


Рис. 7.8. Пример работы с функцией **CADD** на языке CFC

7.6. CARG

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	X	COMPLEX	Комплексное число.
Выходы	CARG	REAL	Угол между осью действительных чисел и радиус-вектором геометрического представления комплексного числа X в радианах.
Используемые модули	ATAN2		

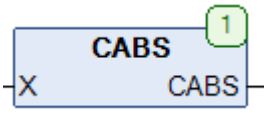


Рис. 7.9. Внешний вид функции **CARG** на языке CFC

Функция **CARG** возвращает значение угла (в радианах) между осью действительных чисел и радиус-вектором геометрического представления комплексного числа **X**.

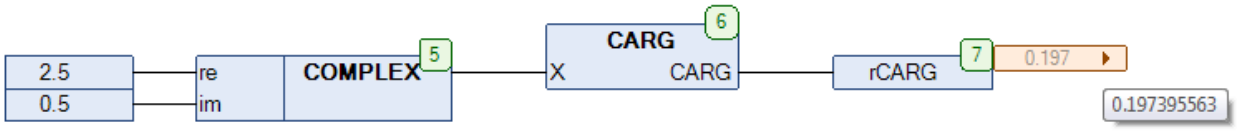


Рис. 7.10. Пример работы с функцией **CARG** на языке CFC

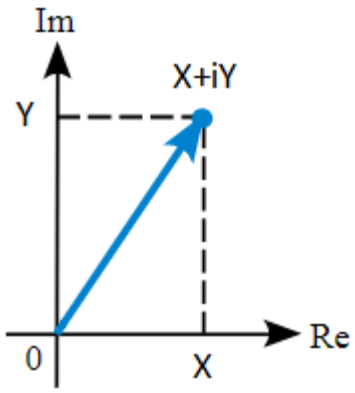


Рис. 7.11. Геометрическое представление комплексного числа

7.7. CASIN

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	X	COMPLEX	Комплексное число.
Выходы	CASIN	COMPLEX	Арксинус комплексного числа.
Используемые модули	CASINH		

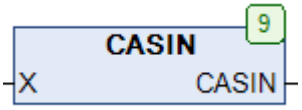


Рис. 7.12. Внешний вид функции **CASIN** на языке CFC

Функция **CASIN** возвращает арксинус комплексного числа **X**. Мнимая часть полученного результата принадлежит интервалу $[-\frac{\pi}{2},\frac{\pi}{2}]$.

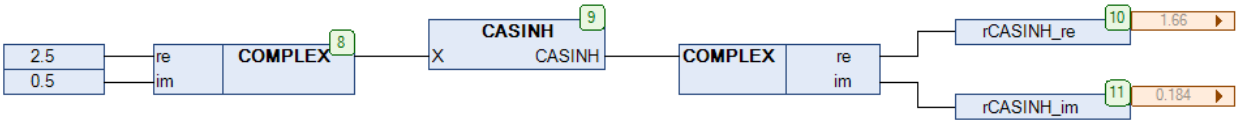


Рис. 7.13. Пример работы с функцией **CASIN** на языке CFC

7.8. CASINH

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	X	COMPLEX	Комплексное число.
Выходы	CASINH	COMPLEX	Гиперболический арксинус комплексного числа.

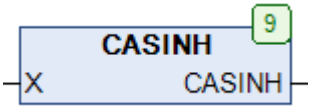


Рис. 7.14. Внешний вид функции CASINH на языке CFC

Функция **CASINH** возвращает гиперболический арксинус комплексного числа **X**. Мнимая часть полученного результата принадлежит интервалу $[-\pi, \pi]$.

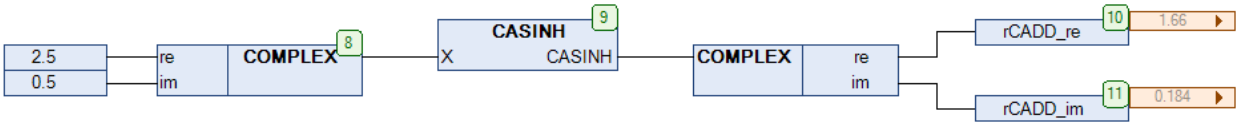


Рис. 7.15. Пример работы с функцией CASINH на языке CFC

7.9. CATAN

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	X	COMPLEX	Комплексное число.
Выходы	CATAN	COMPLEX	Арктангенс комплексного числа.

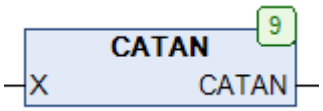


Рис. 7.16. Внешний вид функции **CATAN** на языке CFC

Функция **CATAN** возвращает арктангенс комплексного числа **X**. Мнимая часть полученного результата принадлежит интервалу $[-\frac{\pi}{2}, \frac{\pi}{2}]$.

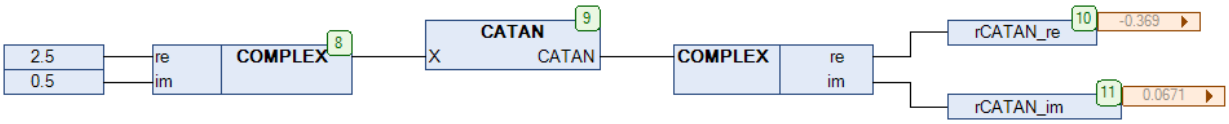


Рис. 7.17. Пример работы с функцией **CATAN** на языке CFC

7.10. CATANH

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	X	COMPLEX	Комплексное число.
Выходы	CATANH	COMPLEX	Гиперболический арктангенс комплексного числа.

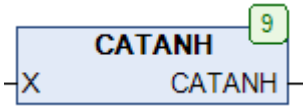


Рис. 7.18. Внешний вид функции **CATANH** на языке CFC

Функция **CATANH** возвращает гиперболический арктангенс комплексного числа **X**. Мнимая часть полученного результата принадлежит интервалу $[-\frac{\pi}{2}, \frac{\pi}{2}]$.

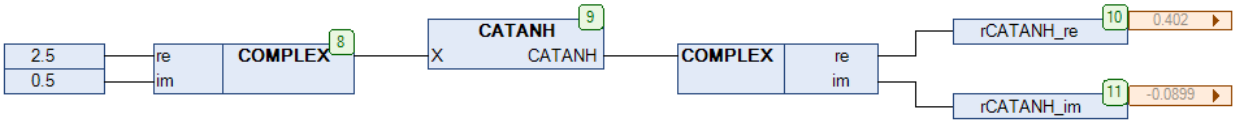


Рис. 7.19. Пример работы с функцией **CATANH** на языке CFC

7.11. CCON

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	X	COMPLEX	Комплексное число.
Выходы	CCON	COMPLEX	Сопряженное число.

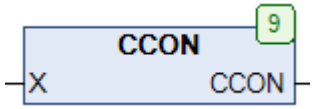


Рис. 7.20. Внешний вид функции **CCON** на языке CFC

Функция **CCON** возвращает сопряженное число для комплексного числа **X**.

$$\begin{cases} \mathrm{CCON.RE} = \mathrm{X.RE} \\ \mathrm{CCON.IM} = -\mathrm{X.IM} \end{cases}$$

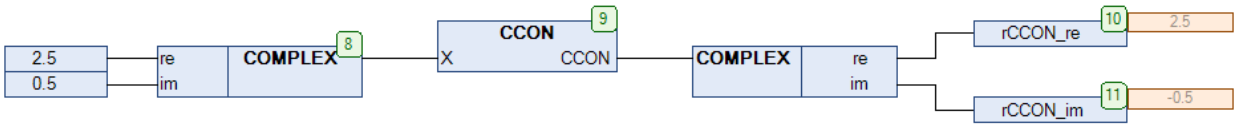


Рис. 7.21. Пример работы с функцией **CCON** на языке CFC

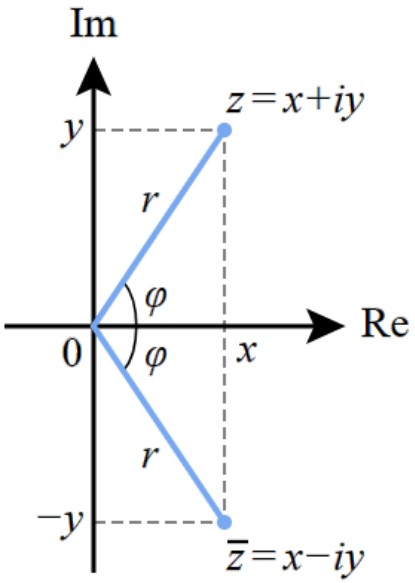


Рис. 7.22. Геометрическое представление сопряженных чисел

7.12. CCOS

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	X	COMPLEX	Комплексное число.
Выходы	CCOS	COMPLEX	Косинус комплексного числа.
Используемые модули	CCOSH, CSET		

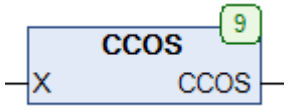


Рис. 7.23. Внешний вид функции **CCOS** на языке CFC

Функция **CCOS** возвращает косинус комплексного числа **X**.

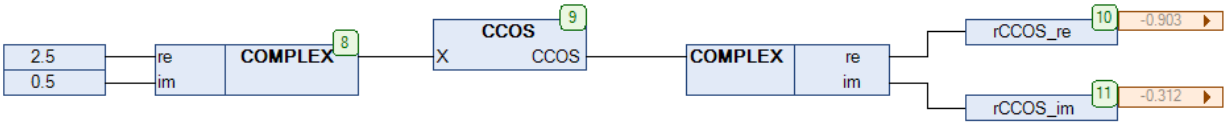


Рис. 7.24. Пример работы с функцией **CCOS** на языке CFC

7.13. CCOSH

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	X	COMPLEX	Комплексное число.
Выходы	CCOSH	COMPLEX	Гиперболический косинус комплексного числа.
Используемые модули	COSH, SINH		

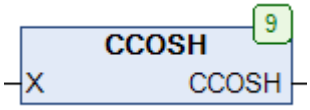


Рис. 7.25. Внешний вид функции **CCOSH** на языке CFC

Функция **CCOSH** возвращает гиперболический косинус комплексного числа **X**.

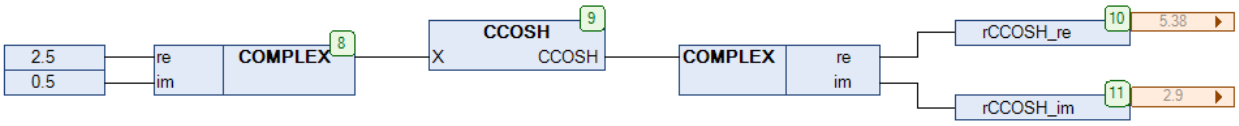


Рис. 7.26. Пример работы с функцией **CCOSH** на языке CFC

7.14. CDIV

Тип модуля: функция	Переменная	Тип	Описание
Входы	X	COMPLEX	Комплексное число.
	Y	COMPLEX	Комплексное число.
Выходы	CDIV	COMPLEX	Частное комплексных чисел.

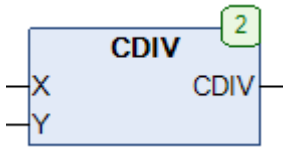


Рис. 7.27. Внешний вид функции **CDIV** на языке CFC

Функция **CDIV** возвращает частное комплексных чисел **X** и **Y**.

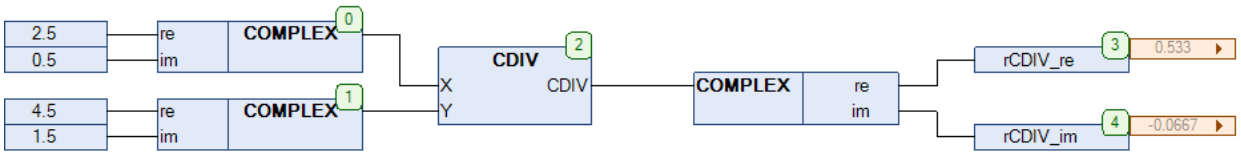


Рис. 7.28. Пример работы с функцией **CDIV** на языке CFC

7.15. CEXP

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	X	COMPLEX	Комплексное число.
Выходы	CEXP	COMPLEX	Экспонента комплексного числа.

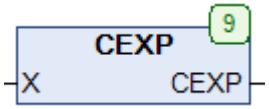


Рис. 7.29. Внешний вид функции **CEXP** на языке CFC

Функция **CEXP** возвращает значение экспоненты комплексного числа **X**.

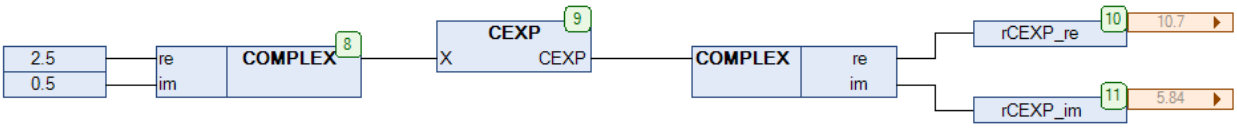


Рис. 7.30. Пример работы с функцией **CEXP** на языке CFC

7.16. CINV

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	X	COMPLEX	Комплексное число.
Выходы	CINV	COMPLEX	Обратное комплексное число.

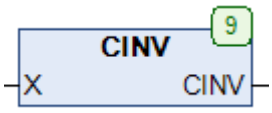


Рис. 7.31. Внешний вид функции **CINV** на языке CFC

Функция CINV возвращает обратное число для комплексного числа **X**.

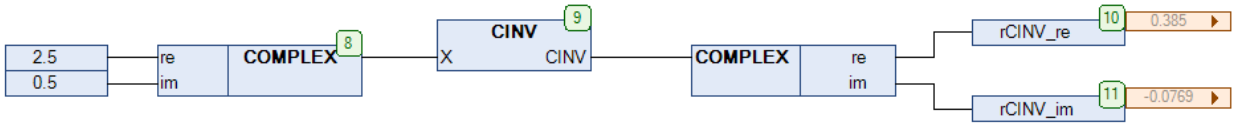


Рис. 7.32. Пример работы с функцией **CINV** на языке CFC

7.17. CLOG

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	X	COMPLEX	Комплексное число.
Выходы	CLOG	COMPLEX	Натуральный логарифм комплексного числа.
Используемые модули	HYPOT, ATAN2		

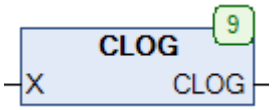


Рис. 7.33. Внешний вид функции **CLOG** на языке CFC

Функция **CLOG** возвращает значение натурального логарифма для комплексного числа **X**.

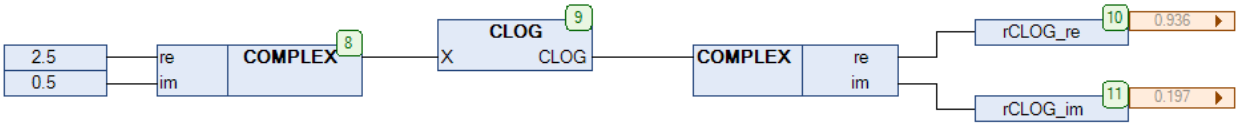


Рис. 7.34. Пример работы с функцией **CLOG** на языке CFC

7.18. CMUL

Тип модуля: функция	Переменная	Тип	Описание
Входы	X	COMPLEX	Комплексное число.
	Y	COMPLEX	Комплексное число.
Выходы	CMUL	COMPLEX	Произведение комплексных чисел.

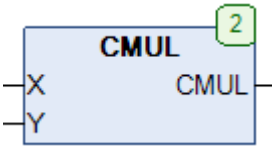


Рис. 7.35. Внешний вид функции **CMUL** на языке CFC

Функция **CMUL** возвращает произведение комплексных чисел **X** и **Y**.

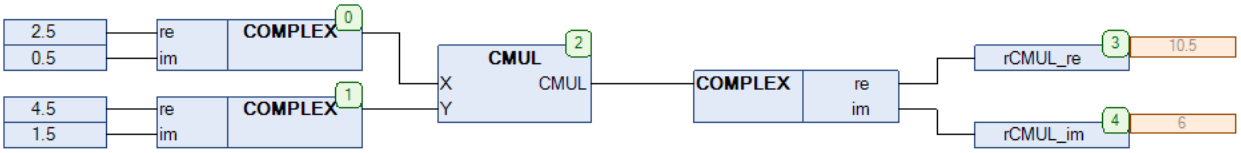


Рис. 7.36. Пример работы с функцией **CMUL** на языке CFC

7.19. CPOL

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	L	REAL	Радиус-вектор.
	A	REAL	Угол между осью действительных чисел и радиус-вектором в радианах.
Выходы	CPOL	COMPLEX	Комплексное число.

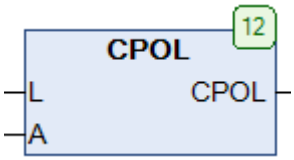


Рис. 7.37. Внешний вид функции **CPOL** на языке CFC

Функция **CPOL** возвращает комплексное число, представленное в геометрическом виде радиус-вектором **L** и углом (в радианах) между осью действительных чисел и радиус-вектором **A**.

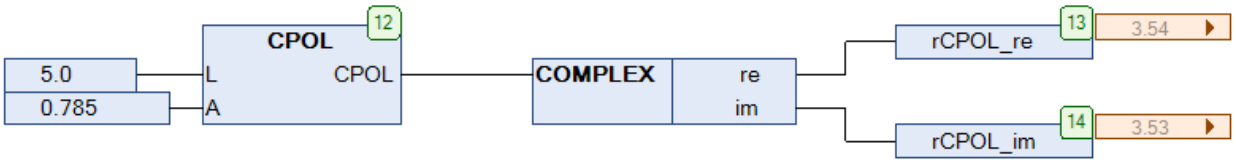


Рис. 7.38. Пример работы с функцией **CPOL** на языке CFC

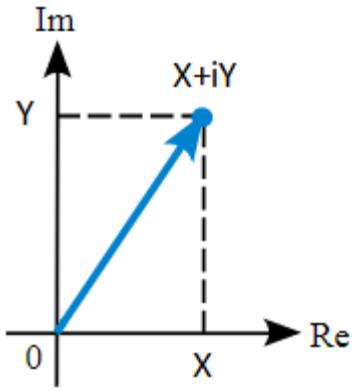


Рис. 7.39. Геометрическое представление комплексного числа

7.20. CPOW

Тип модуля: функция	Переменная	Тип	Описание
Входы	X	COMPLEX	Комплексное число.
	Y	COMPLEX	Показатель степени комплексного числа.
Выходы	CPOW	COMPLEX	Комплексное число, возведенное в степень.
Используемые модули	CEXP, CMUL, CLOG		

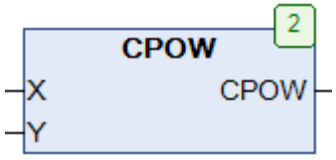


Рис. 7.40. Внешний вид функции **CPOW** на языке CFC

Функция **CPOW** возвращает комплексное число **X**, возведенное в степень **Y**. Показатель степени **Y** также является комплексным числом.

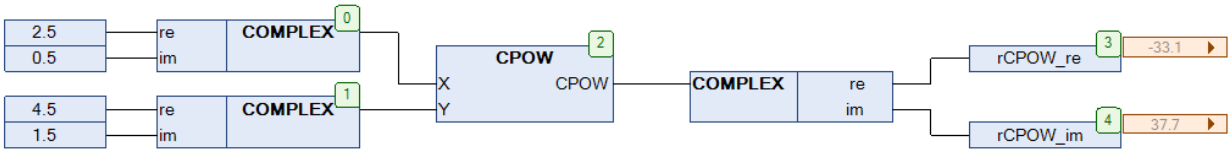


Рис. 7.41. Пример работы с функцией **CPOW** на языке CFC

7.21. CSET

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	RE	REAL	Действительная часть комплексного числа.
	IM	REAL	Мнимая часть комплексного числа.
Выходы	CSET	COMPLEX	Комплексное число.

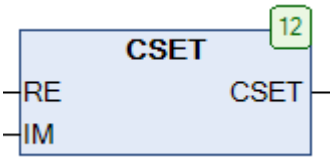


Рис. 7.42. Внешний вид функции **CSET** на языке CFC

Функция **CSET** возвращает комплексное число с действительной частью **X** и мнимой частью **Y**.

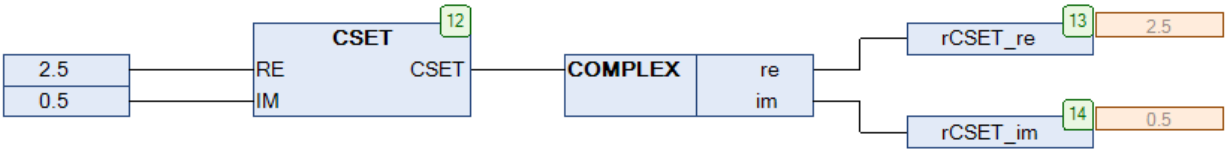


Рис. 7.43. Пример работы с функцией **CSET** на языке CFC

7.22. CSIN

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	X	COMPLEX	Комплексное число.
Выходы	CSIN	COMPLEX	Синус комплексного числа.
Используемые модули	CCOSH, CSINH		

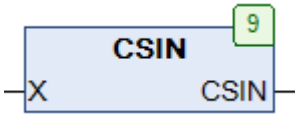


Рис. 7.44. Внешний вид функции **CSIN** на языке CFC

Функция **CSIN** возвращает синус комплексного числа **X**.

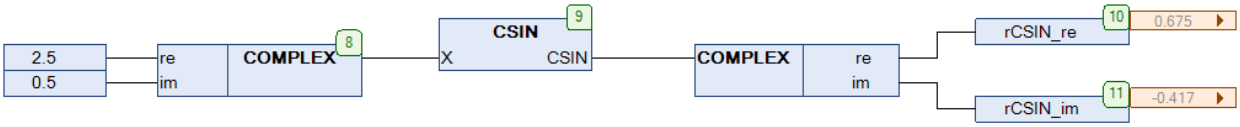


Рис. 7.45. Пример работы с функцией **CSIN** на языке CFC

7.23. CSINH

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	X	COMPLEX	Комплексное число.
Выходы	CSINH	COMPLEX	Гиперболический синус комплексного числа.
Используемые модули	COSH, SINH		

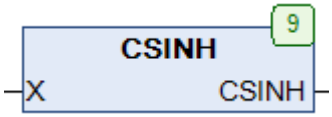


Рис. 7.46. Внешний вид функции **CSINH** на языке CFC

Функция **CSINH** возвращает гиперболический синус комплексного числа X.

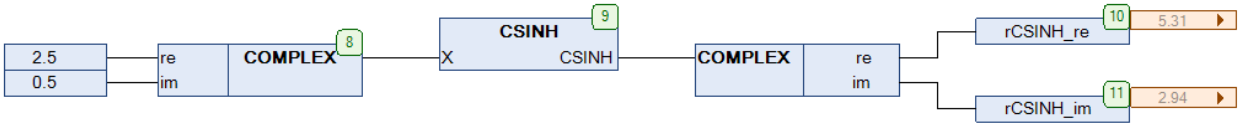


Рис. 7.47. Пример работы с функцией **CSINH** на языке CFC

7.24. CSQRT

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	X	COMPLEX	Комплексное число.
Выходы	CSQRT	COMPLEX	Квадратный корень комплексного числа.
Используемые модули	HYPOT, SGN		

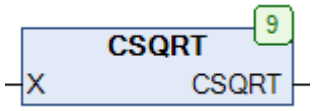


Рис. 7.48. Внешний вид функции **CSQRT** на языке CFC

Функция **CSQRT** возвращает квадратный корень комплексного числа **X**.

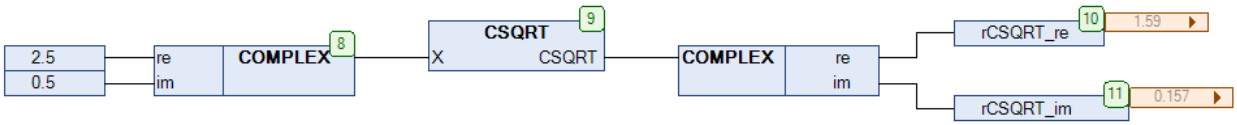


Рис. 7.49. Пример работы с функцией **CSQRT** на языке CFC

7.25. CSUB

Тип модуля: функция	Переменная	Тип	Описание
Входы	X	COMPLEX	Комплексное число.
	Y	COMPLEX	Комплексное число.
Выходы	CSUB	COMPLEX	Разность комплексных чисел.

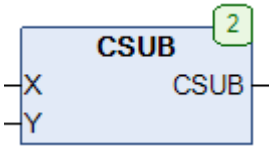


Рис. 7.50. Внешний вид функции **CSUB** на языке CFC

Функция **CSUB** возвращает разность комплексных чисел **X** и **Y**.

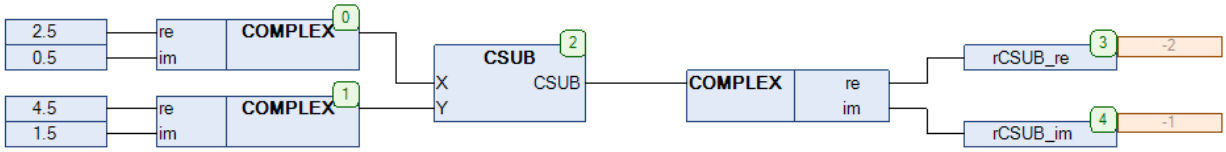


Рис. 7.51. Пример работы с функцией **CSUB** на языке CFC

7.26. CTAN

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	X	COMPLEX	Комплексное число.
Выходы	CTAN	COMPLEX	Тангенс комплексного числа.
Используемые модули	COSH, SINH		

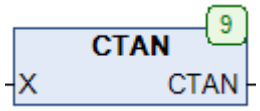


Рис. 7.52. Внешний вид функции **CTAN** на языке CFC

Функция **CTAN** возвращает тангенс комплексного числа **X**.

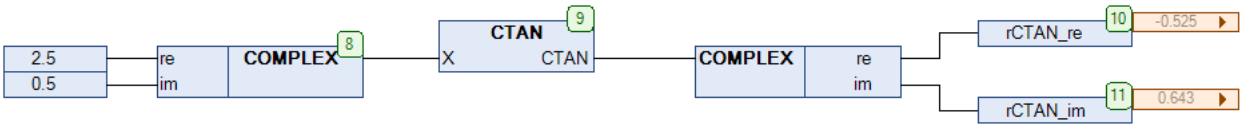


Рис. 7.53. Пример работы с функцией **CTAN** на языке CFC

7.27. CTANH

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	X	COMPLEX	Комплексное число.
Выходы	CTANH	COMPLEX	Гиперболический тангенс комплексного числа.
Используемые модули	COSH, SINH		

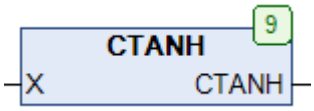


Рис. 7.54. Внешний вид функции **CTANH** на языке CFC

Функция **CTANH** возвращает гиперболический тангенс комплексного числа **X**.

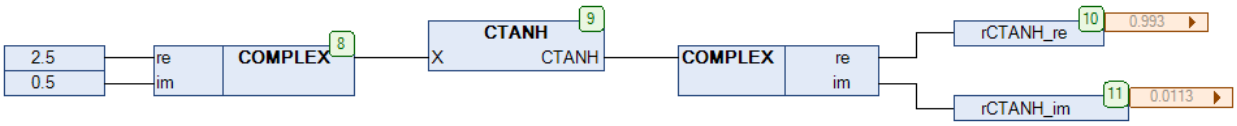


Рис. 7.55. Пример работы с функцией **CTANH** на языке CFC

8. Арифметика чисел двойной точности

8.1. Вступление

В соответствии со стандартом МЭК 61131-3 числа с плавающей точкой хранятся в ПЛК в виде переменных типа **REAL**. Переменные типа **REAL** соответствует числам одинарной точности стандарта IEEE 754 и занимает **32 бита**. Они обеспечивают точность в 7-8 десятичных цифр в диапазоне от 10^{-39} до 10^{38}. Для большинства задач этого оказывается достаточно, но в отдельных ситуациях может потребоваться повышенная точность вычислений. **МЭК 61131-3** предусматривает тип **LREAL**, переменные которого занимают **64 бита** и соответствуют числам двойной точности стандарта **IEEE 754** – но поддержка данного типа реализована не на всех ПЛК. Для подобных случаев библиотека **OSCAT** предоставляет тип данных REAL2, эмулирующий поддержку чисел двойной точности. **REAL2** представляет собой структуру из двух переменных типа **REAL**, одна из которых – **RX** – содержит «грубую» часть значения (в нее входит целая часть и 7-8 знаков после запятой), а вторая – **R1** – «точную» (содержит оставшиеся знаки после запятой). При этом каждая операция над подобной структурой может привести к потере точности. В данной главе описаны функции, которые применяются для работы с этой структурой.

8.2. R2_ABS

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	X	REAL2	Входное значение.
Выходы	R2_ABS	REAL2	Значение по модулю.

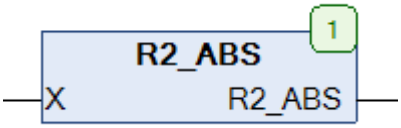


Рис. 8.1. Внешний вид функции **R2_ABS** на языке CFC

Функция **R2_ABS** возвращает модуль числа двойной точности **X** типа REAL2.

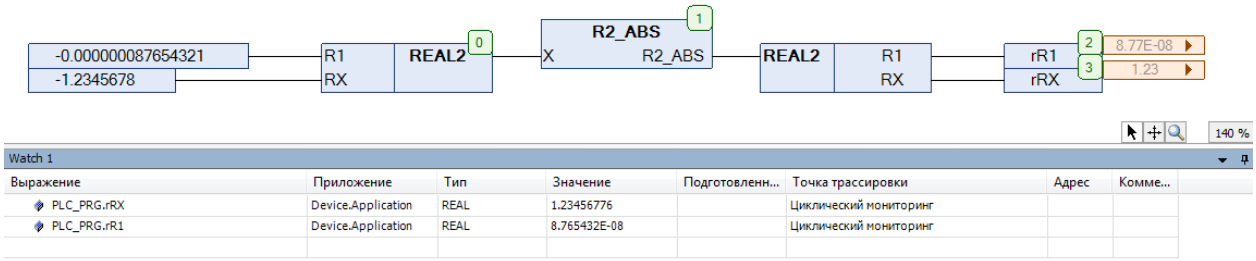


Рис. 8.2. Пример работы с функцией **R2_ABS** на языке CFC

8.3. R2_ADD

Тип модуля: функция	Переменная	Тип	Описание
Входы	X	REAL2	Входное значение.
	Y	REAL	Входное значение.
Выходы	R2_ABS	REAL2	Сумма X и Y.

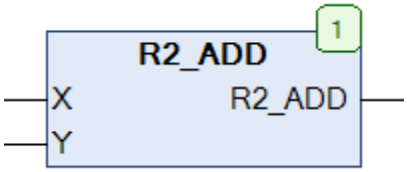


Рис. 8.3. Внешний вид функции **R2_ADD** на языке CFC

Функция **R2_ADD** возвращает сумму числа двойной точности **X** типа REAL2 и значения **Y** типа **REAL**.

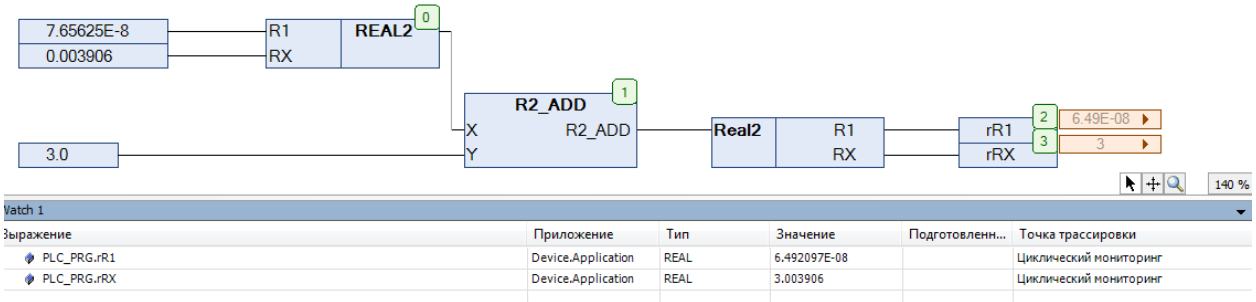


Рис. 8.4. Пример работы с функцией **R2_ADD** на языке CFC

8.4. R2_ADD2

Тип модуля: функция	Переменная	Тип	Описание
Входы	X	REAL2	Входное значение.
	Y	REAL2	Входное значение.
Выходы	R2_ADD2	REAL2	Сумма X и Y.

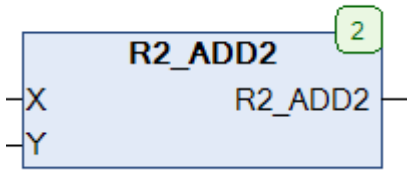


Рис. 8.5. Внешний вид функции **R2_ADD2** на языке CFC

Функция **R2_ADD2** возвращает сумму чисел двойной точности **X** и **Y** типа REAL2.

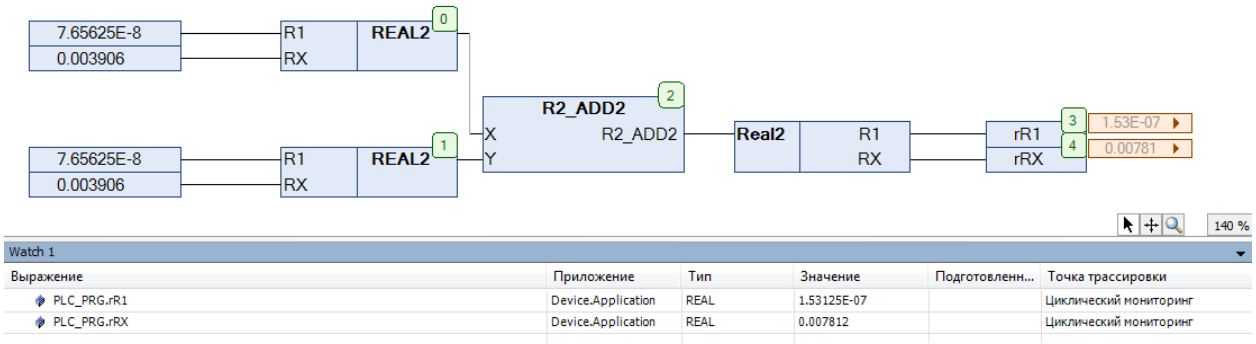


Рис. 8.6. Пример работы с функцией **R2_ADD2** на языке CFC

8.5. R2_MUL

Тип модуля: функция	Переменная	Тип	Описание
Входы	X	REAL2	Входное значение.
	Y	REAL	Входное значение.
Выходы	R2_MUL	REAL2	Произведение X и Y.

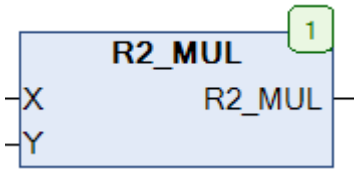


Рис. 8.7. Внешний вид функции **R2_MUL** на языке CFC

Функция **R2_MUL** возвращает произведение числа двойной точности **X** типа REAL2 и значения **Y** типа **REAL**.

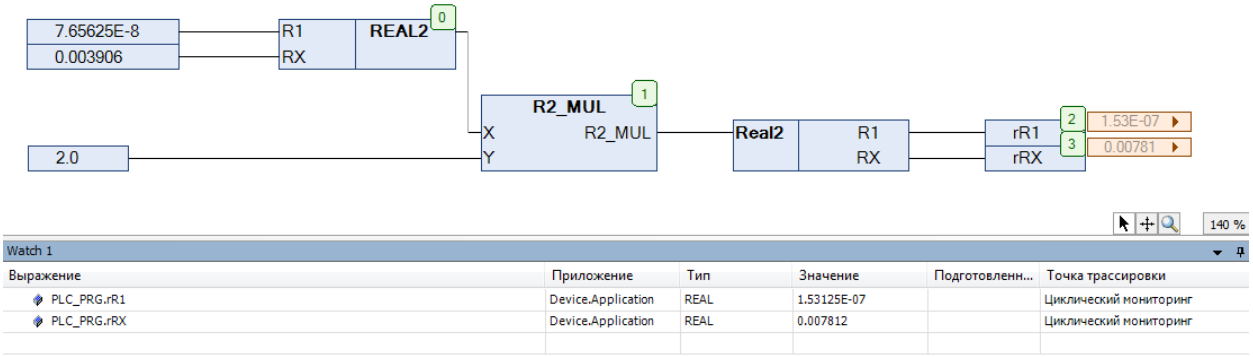


Рис. 8.8. Пример работы с функцией **R2_MUL** на языке CFC

8.6. R2_SET

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	X	REAL	Входное значение.
Выходы	R2_SET	REAL2	Выходное значение.

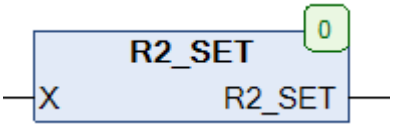


Рис. 8.9. Внешний вид функции **R2_SET** на языке CFC

Функция **R2_SET** присваивает значение входной переменной **X** типа **REAL** выходной переменной двойной точности **R2_SET** типа REAL2.

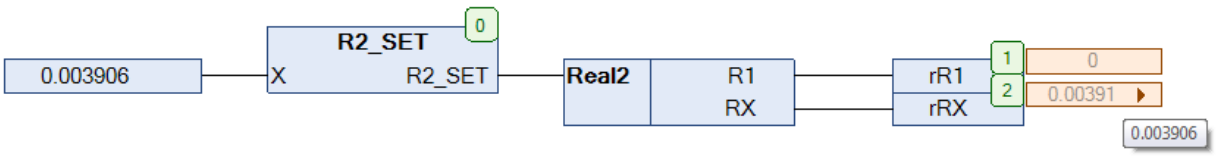


Рис. 8.10. Пример работы с функцией **R2_SET** на языке CFC

9. Арифметические функции

9.1. F_LIN

Тип модуля: функция	Переменная	Тип	Описание
Входы	X	REAL	Аргумент.
	A	REAL	Угловой коэффициент.
	B	REAL	Коэффициент сдвига.
Выходы	F_LIN	REAL	Значение линейной функции.

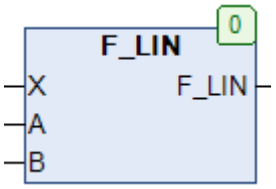


Рис. 9.1. Внешний вид функции **F_LIN** на языке CFC

Функция **F_LIN** возвращает для аргумента **X** значение линейной функции, вычисленное по формуле

$$\mathrm{F\_LIN(X)} = \mathrm{A} \cdot \mathrm{X} + \mathrm{B}$$

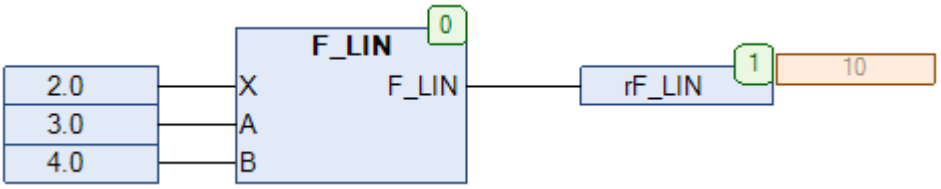


Рис. 9.2. Пример работы с функцией **F_LIN** на языке CFC

9.2. F_LIN2

Тип модуля: функция	Переменная	Тип	Описание
Входы	X	REAL	Аргумент.
	X1	REAL	Координата X для точки 1.
	Y1	REAL	Координата Y для точки 1.
	X2	REAL	Координата X для точки 2.
	Y2	REAL	Координата Y для точки 2.
Выходы	F_LIN2	REAL	Линейно интерполированное значение Y для X.

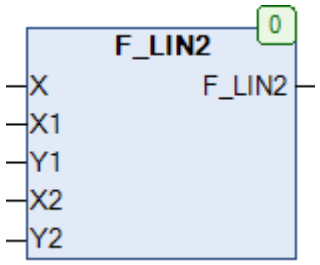


Рис. 9.3. Внешний вид функции **F_LIN2** на языке CFC

Функция **F_LIN2** возвращает для точки с координатой X результат линейной интерполяции координаты **Y** между узловыми точками **(X1, Y1)** и **(X2, Y2).** Если значение **X** не входит в диапазон **(X1…X2),** то значение Y (**F_LIN2**) экстраполируется.

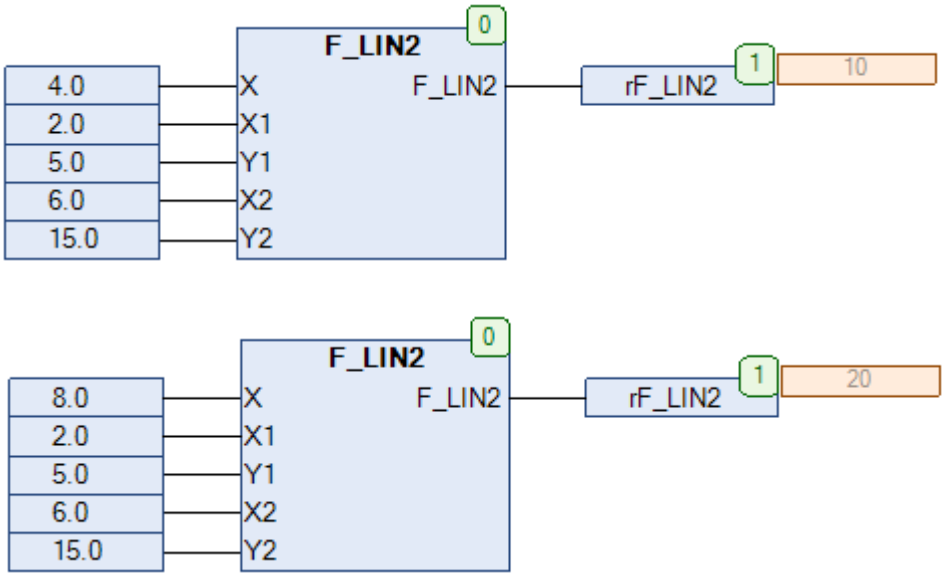


Рис. 9.4. Пример работы с функцией **F_LIN2** на языке CFC

9.3. F_POLY

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	X	REAL	Аргумент.
	C	ARRAY [0..7] OF REAL	Коэффициенты полинома.
Выходы	F_POLY	REAL	Значение полинома 7-ой степени.

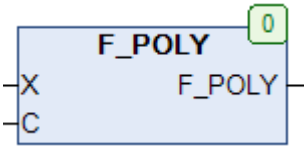


Рис. 9.5. Внешний вид функции **F_POLY** на языке CFC

Функция **F_POLY** возвращает значение полинома 7-ой степени с коэффициентами **C** (где C – массив из 8-ми значений) для аргумента **X**, вычисленное по формуле

$$\mathrm{F\_POLY(X)} = \sum_{i=0}^{7} \mathrm{C[i]} \cdot \mathrm{X^i}$$

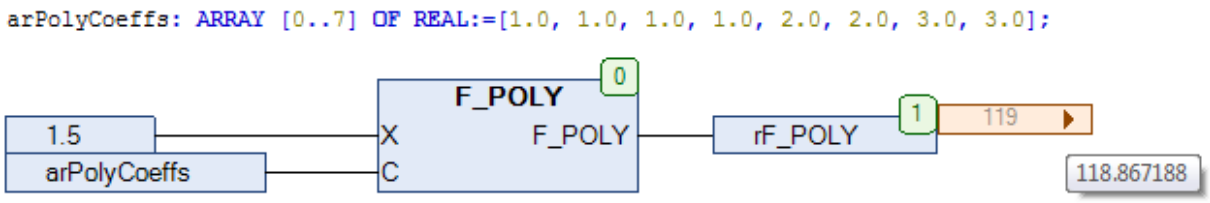


Рис. 9.6. Пример работы с функцией **F_POLY** на языке CFC

9.4. F_POWER

Тип модуля: функция	Переменная	Тип	Описание
Входы	a	REAL	Коэффициент.
	X	REAL	Аргумент.
	n	REAL	Показатель степени.
Выходы	F_POWER	REAL	Значение степенной функции.

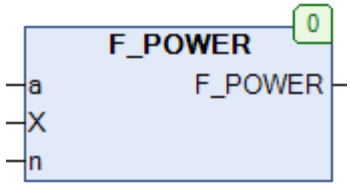


Рис. 9.7. Внешний вид функции **F_POWER** на языке CFC

Функция **F_POWER** возвращает для аргумента X значение степенной функции, вычисленное по формуле

$$\mathrm{F\_POWER(X)} = \mathrm{a} \cdot \mathrm{X^n}$$

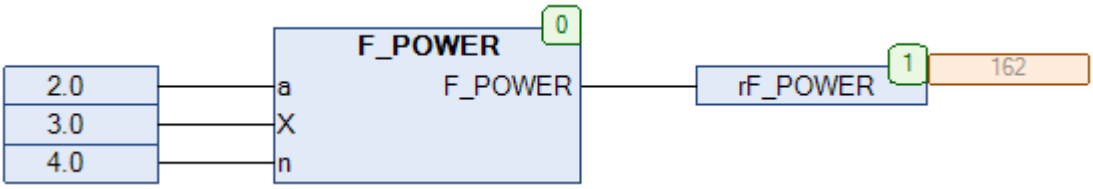


Рис. 9.8. Пример работы с функцией **F_POWER** на языке CFC

9.5. F_QUAD

Тип модуля: функция	Переменная	Тип	Описание
Входы	X	REAL	Аргумент.
	A	REAL	Коэффициент.
	B	REAL	Коэффициент.
	C	REAL	Коэффициент.
Выходы	F_QUAD	REAL	Значение квадратичной функции.

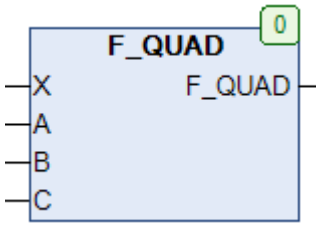


Рис. 9.9. Внешний вид функции **F_QUAD** на языке CFC

Функция **F_QUAD** возвращает для аргумента X значение квадратичной функции, вычисленное по формуле

$$\mathrm{F\_QUAD(X)} = \mathrm{A} \cdot \mathrm{X}^2 + \mathrm{B} \cdot \mathrm{X} + \mathrm{C}$$

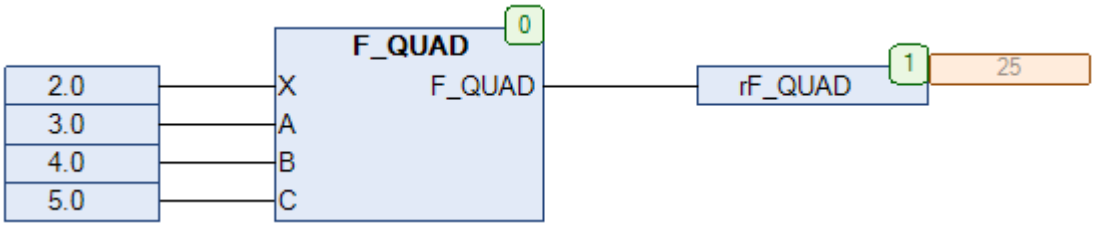


Рис. 9.10. Пример работы с функцией **F_QUAD** на языке CFC

9.6. FRMP_B

Тип модуля: функция	Переменная	Тип	Описание
Входы	START	BYTE	Начальное значение.
	DIR	BOOL	Направление изменения выхода.
	TD	TIME	Прошедшее время.
	TR	TIME	Полное время генерации (от 0 до 255).
Выходы	FRMP_B	BYTE	Выход блока.

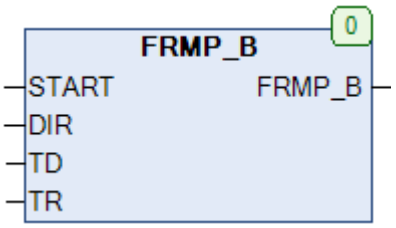


Рис. 9.11. Внешний вид функции **FRMP_B** на языке CFC

Функция **FRMP_B** используется для реализации генератора линейной функции. Пока функция находится в работе, значение ее выхода линейно изменяется от текущего (**START**) до максимального (при **DIR=TRUE**) или минимального (при **DIR=FALSE**). Минимальное и максимальное значения определяются диапазоном типа **BYTE** и соответственно составляют **0** и **255**. Вход **TR** определяет полное время генерации (от 0 до 255 секунд), а вход **TD** – время, прошедшее с начала работы функции. Таким образом, значение входа **TD** должно увеличиваться во время работы функции для изменения ее выхода.

См. также генератор _RMP_B, построенный на базе данной функции.

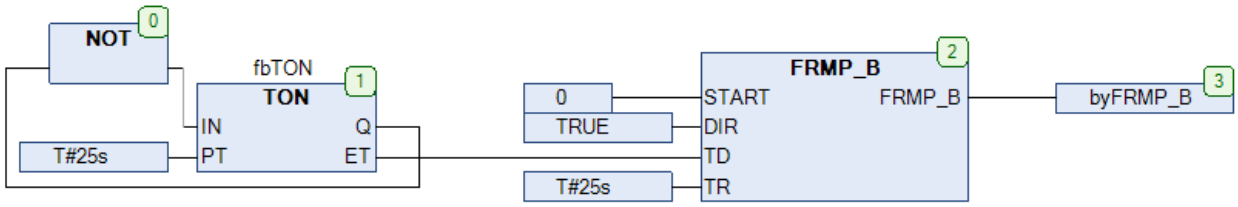


Рис. 9.12.Пример работы с функцией **FRMP_B** на языке CFC

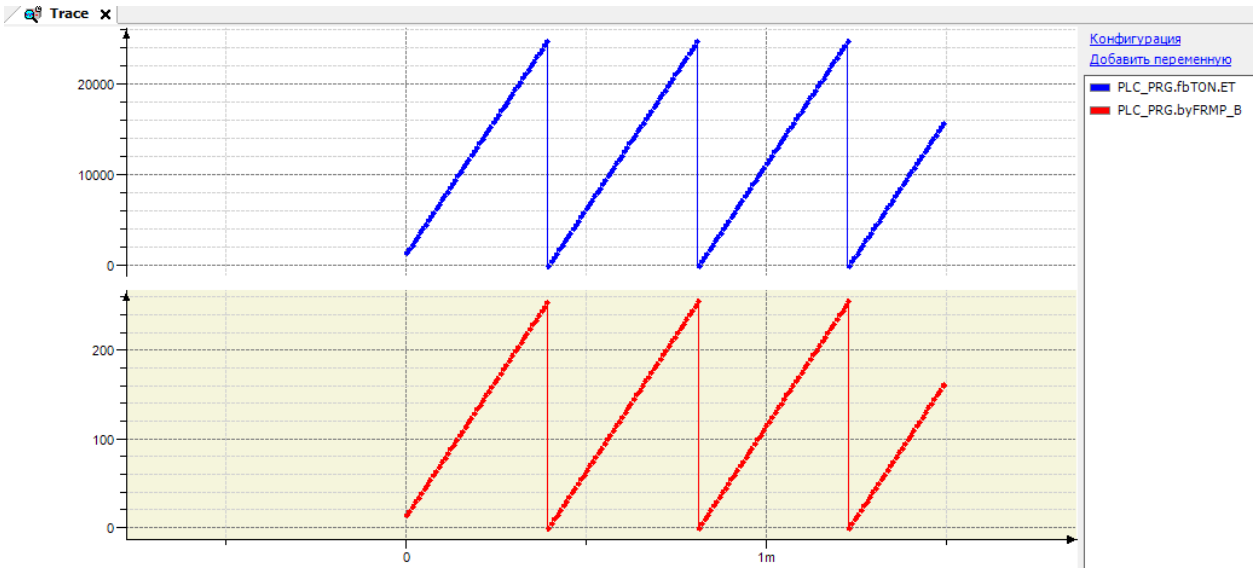


Рис. 9.13. Трассировка работы функции **FRMP_B** для рис. 9.12
(для **ET** значения по оси Y – в мс)

9.7. FT_AVG

Тип модуля: ФБ	Переменная	Тип	Описание
Входы	IN	REAL	Контролируемое значение.
	E	BOOL	Сигнал управления блоком.
	N	N	Число точек усреднения сигнала (0…32).
	RST	BOOL	Сигнал сброса блока.
Выходы	FT_AVG	REAL	Значение скользящей средней.

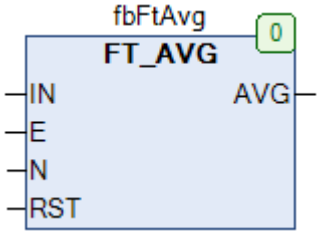


Рис. 9.14. Внешний вид ФБ **FT_AVG** на языке CFC

Функциональный блок **FT_AVG** представляет собой фильтр скользящей средней. Если вход **E** имеет значение **TRUE**, то блок находится в работе, и на выход **AVG** подается значение входа **IN**, усредненное за последние **N** циклов ПЛК (**K** – номер текущего цикла ПЛК):

$$Y_K = \frac{1}{\mathrm{N}} \sum_{i=K-N}^{K} X_i$$

Если N=0, то AVG=IN. По переднему фронту на входе **RST** данные блока обнуляются, и усреднение начинается заново.

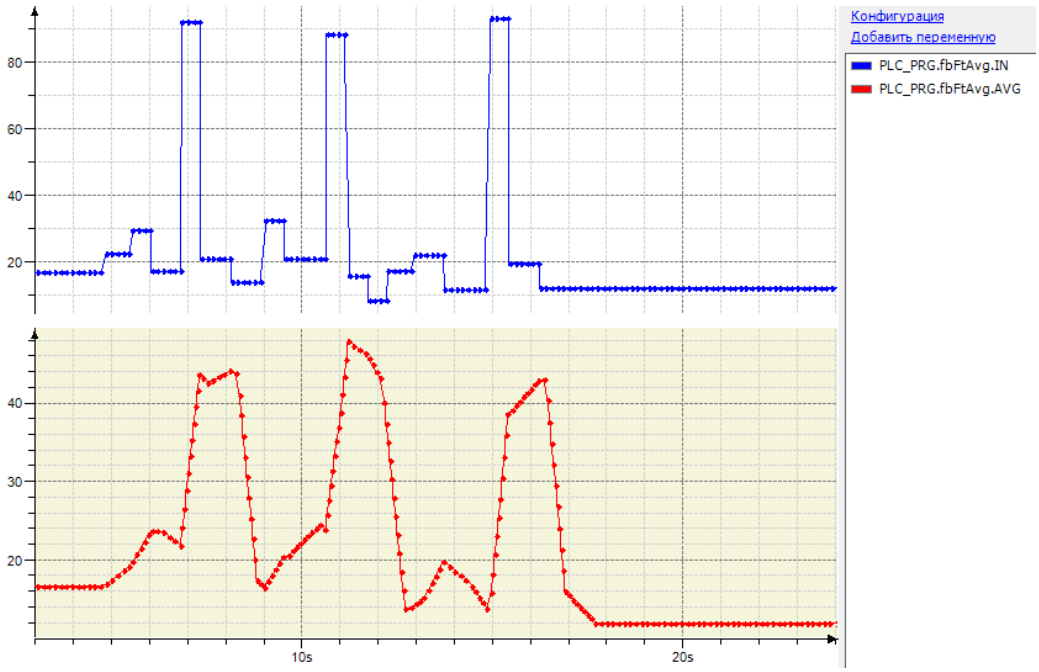


Рис. 9.15. Трассировка работы ФБ **FT_AVG** (N=30, время цикла ПЛК=50 мс)

9.8. FT_MIN_MAX

Тип модуля: ФБ	**Переменная**	**Тип**	**Описание**
Входы	IN	REAL	Контролируемое значение.
	RST	BOOL	Сигнал сброса блока.
Выходы	MX	REAL	Максимальное значение.
	MN	REAL	Минимальное значение.

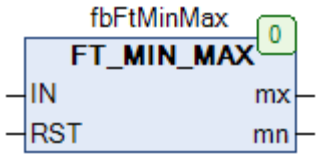


Рис. 9.16. Внешний вид ФБ **FT_MIN_MAX** на языке CFC

Функциональный блок **FT_MIN_MAX** собирает статистику о входе **IN**, сохраняя его максимальное и минимальное значение на выходах **MX** и **MN**. По переднему фронту на входе **RST** значения выходов обнуляются, после чего сбор статистики начинается заново.

9.9. FT_RMP

Тип модуля: ФБ	Переменная	Тип	Описание
Входы	Rmp	BOOL	Сигнал управления блоком.
	IN	REAL	Контролируемый сигнал.
	KR	REAL	Допустимая скорость нарастания, ед/с.
	KF	REAL	Допустимая скорость спада, ед/с.
Выходы	out	REAL	Сглаженное значение сигнала.
	busy	BOOL	Флаг «блок в работе».
	UD	BOOL	Направление изменения сигнала.

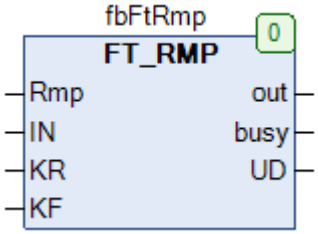


Рис. 9.17. Внешний вид ФБ **FT_RMP** на языке CFC

Функциональный блок **FT_RMP** используется для сглаживания сигналов. Пока вход **Rmp** имеет значение **TRUE** блок находится в работе и значение выхода **out** следует за значением входа **IN**, при этом допустимая максимальная скорость увеличения значения на выходе (в ед./с) определяется входом **KR**, а допустимая максимальная скорость уменьшения – входом **KF**. Выход **busy** имеет значение **TRUE**, пока происходит изменение выхода **out**. Выход **UD** имеет значение **TRUE**, если значение выхода **out** увеличивается, и **FALSE** – если уменьшается. Величина изменения выхода **out** зависит от времени цикла ПЛК; см. также блоки RMP_B и RMP_W с фиксированной величиной изменения выходов.

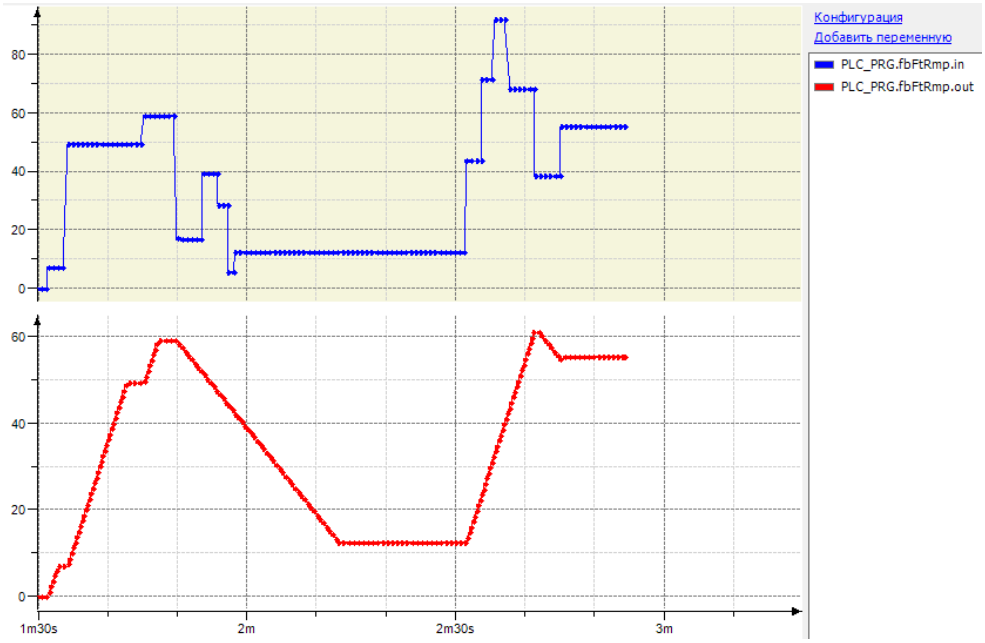


Рис. 9.18. Трассировка работы ФБ **FT_RMP** (KR=5.0, KF=2.0, время цикла ПЛК=50 мс)

9.10. LINEAR_INT

Тип модуля: функция	Переменная	Тип	Описание
Входы	X	REAL	Аргумент.
	XY	ARRAY [1..20, 0..1] OF REAL	Таблица узловых точек.
	Pts	INT	Число используемых узловых точек (3..20).
Выходы	LINEAR_INT	REAL	Линейно интерполированное значение Y для X.

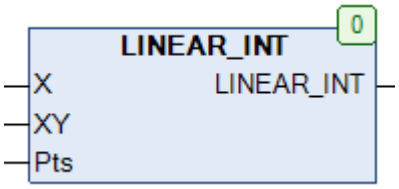


Рис. 9.19. Внешний вид функции **LINEAR_INT** на языке CFC

Функция **LINEAR_INT** выполняет для значения **X** кусочно-линейную интерполяцию по зависимости, определяемой таблицей узловых точек **XY**. Значения **X** в таблице должны быть отсортированы по возрастанию. Вход **Pts** определяет число используемых для интерполяции узловых точек и может принимать значение в диапазоне **[3…20]**. Если значение **X** не входит в диапазон таблицы, то значение Y (**LINEAR_INT**) экстраполируется.

```
arXY: ARRAY[1..20,0..1] OF REAL:=[-10.0, -0.53, 10.0, 0.53, 100.0, 88.3, 200.0, 122.2];
```

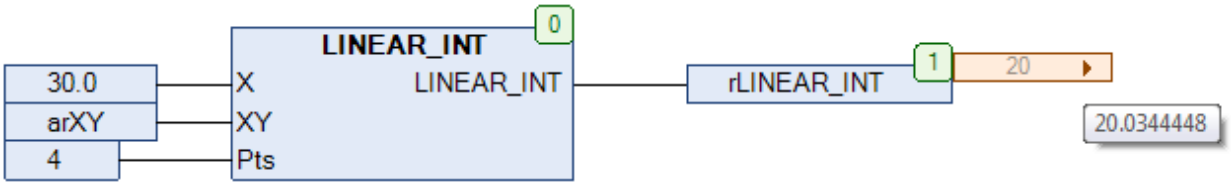


Рис. 9.20. Пример работы с функцией **LINEAR_INT** на языке CFC

См. также рис. 9.23.

9.11. POLYNOM_INT

Тип модуля: функция	Переменная	Тип	Описание
Входы	X	REAL	Аргумент.
	XY	ARRAY [1..5, 0..1] OF REAL	Таблица узловых точек.
	Pts	INT	Число используемых узловых точек (3..5).
Выходы	POLYNOM_INT	REAL	Полиномиально интерполированное значение Y для X.

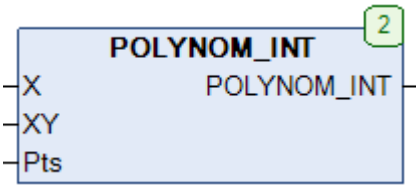


Рис. 9.21. Внешний вид функции **POLYNOM_INT** на языке CFC

Функция **POLYNOM_INT** выполняет для значения **X** полиномиальную интерполяцию по зависимости, определяемой таблицей узловых точек **XY**. Значения **X** в таблице должны быть отсортированы по возрастанию. Вход **Pts** определяет число используемых для интерполяции узловых точек и может принимать значение в диапазоне **[3…5].** Если значение **X** не входит в диапазон таблицы, то значение Y (**POLYNOM_INT**) экстраполируется.

```
arXY: ARRAY[1..20,0..1] OF REAL:=[-10.0, -0.53, 10.0, 0.53, 100.0, 88.3, 200.0, 122.2];
```

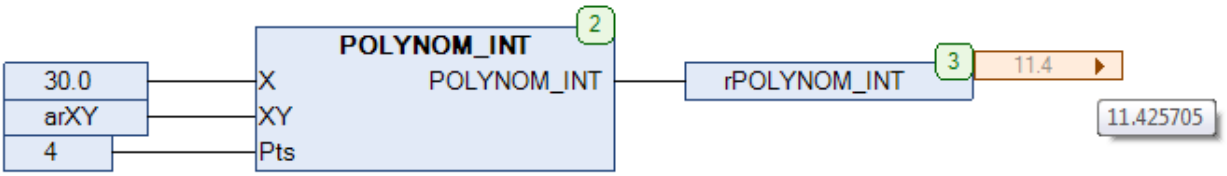


Рис. 9.22. Пример работы с функцией **POLYNOM_INT** на языке CFC

На рис. 9.23 приведен пример работы функций **LINEAR_INT** и **POLYNOM_INT**. Черным цветом построена исходная табличная зависимость **arXY**, синим – зависимость с дополнительным линейно интерполированными точками, красным – зависимость с дополнительными полиномиально интерполированными точками. Координаты **X** дополнительных точек: (-5, 5, 75, 150).

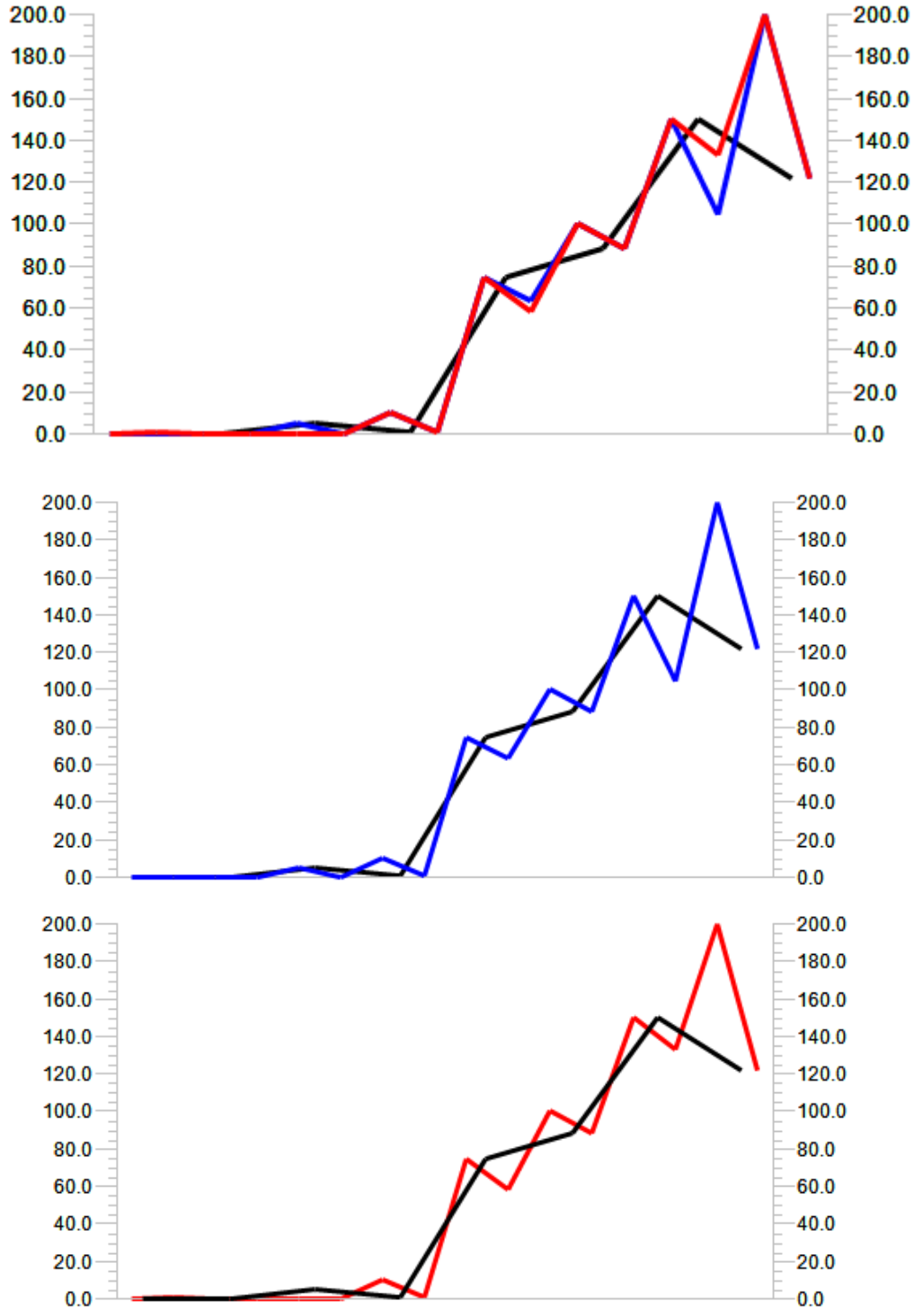


Рис. 9.23. Пример работы функций **LINEAR_INT** и **POLYNOM_INT** на языке CFC

10. Геометрические функции

10.1. CIRCLE_A

Тип модуля: функция	Переменная	Тип	Описание
Входы	rx	REAL	Радиус окружности.
	ax	REAL	Угол сектора в градусах.
Выходы	CIRCLE_A	REAL	Площадь сектора окружности.

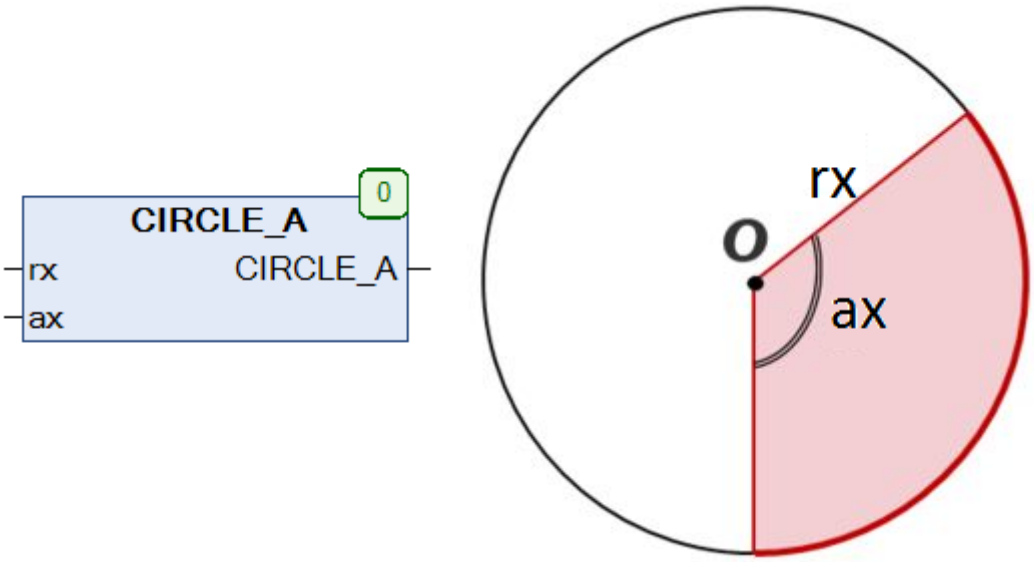


Рис. 10.1. Внешний вид функции **CIRCLE_A** на языке CFC

Функция **CIRCLE_A** возвращает для окружности с радиусом **rx** площадь сектора с углом **ax**, вычисленную по формуле:

$$\mathrm{CIRCLE\_A} = \frac{\pi \cdot \mathrm{rx}^2 \cdot \mathrm{ax}}{360}$$

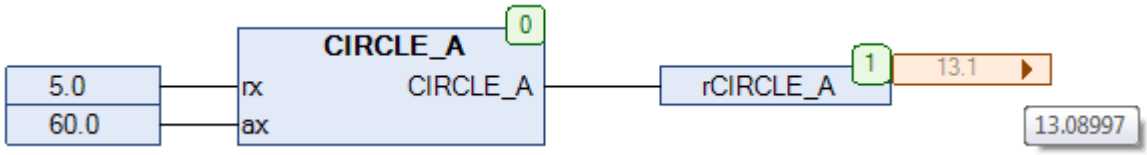


Рис. 10.2. Пример работы с функцией **CIRCLE_A** на языке CFC

10.2. CIRCLE_C

Тип модуля: функция	Переменная	Тип	Описание
Входы	rx	REAL	Радиус окружности.
	ax	REAL	Угол сектора в градусах.
Выходы	CIRCLE_C	REAL	Длина дуги сектора окружности.

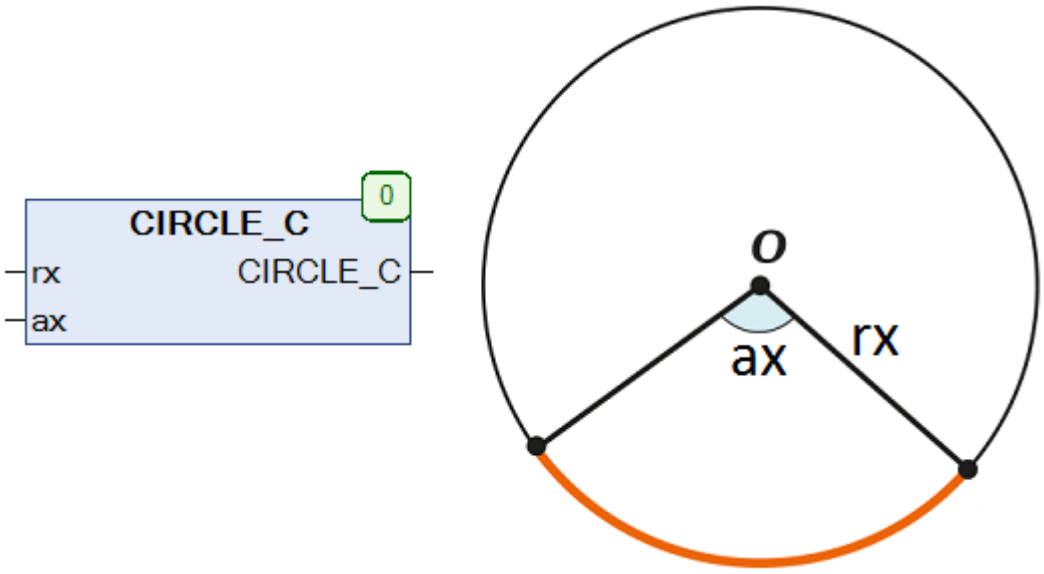


Рис. 10.3. Внешний вид функции **CIRCLE_C** на языке CFC

Функция **CIRCLE_C** возвращает для окружности с радиусом **rx** длину дуги сектора с углом **ax**, вычисленную по формуле:

$$\mathrm{CIRCLE\_C} = \frac{2\pi \cdot \mathrm{rx} \cdot \mathrm{ax}}{360}$$

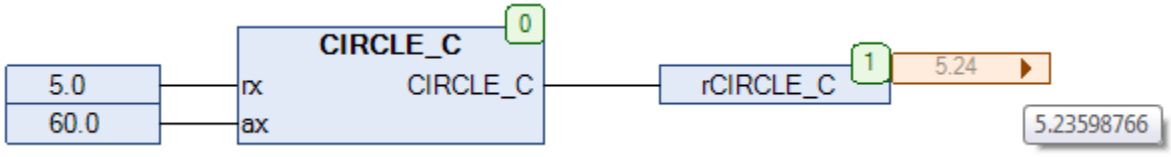


Рис. 10.4. Пример работы с функцией **CIRCLE_C** на языке CFC

10.3. CIRCLE_SEG

Тип модуля: функция	Переменная	Тип	Описание
Входы	rx	REAL	Радиус окружности.
	HX	REAL	Высота сегмента.
Выходы	CIRCLE_SEG	REAL	Площадь сегмента окружности.

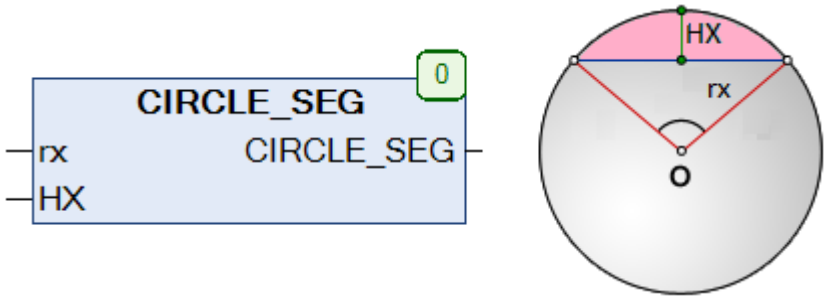


Рис. 10.5. Внешний вид функции **CIRCLE_SEG** на языке CFC

Функция **CIRCLE_SEG** возвращает для окружности с радиусом **RX** площадь сегмента с высотой **HX**.

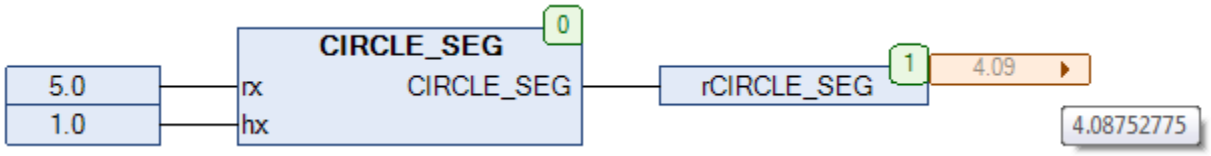


Рис. 10.6. Пример работы с функцией **CIRCLE_SEG** на языке CFC

10.4. CONE_V

Тип модуля: функция	Переменная	Тип	Описание
Входы	rx	REAL	Радиус основания конуса.
	hx	REAL	Высота конуса.
Выходы	CONE_V	REAL	Объем конуса.

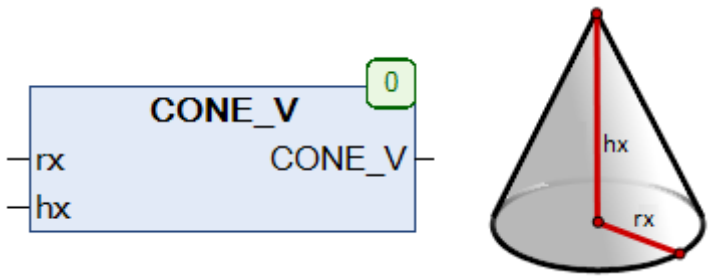


Рис. 10.7. Внешний вид функции **CONE_V** на языке CFC

Функция **CONE_V** возвращает объем конуса с радиусом основания **rx** и высотой **hx**, вычисленный по формуле

$$\mathrm{CONE\_V} = \frac{\pi \cdot \mathrm{rx}^2 \cdot \mathrm{hx}}{3}$$

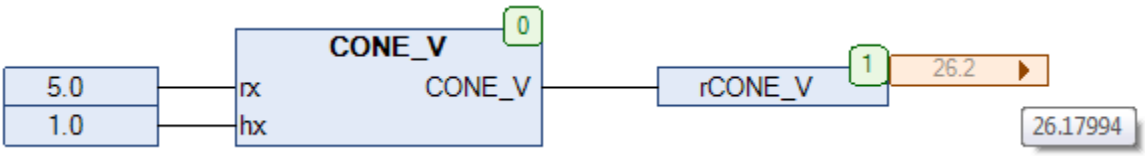


Рис. 10.8. Пример работы с функцией **CONE_V** на языке CFC

10.5. ELLIPSE_A

Тип модуля: функция	Переменная	Тип	Описание
Входы	R1	REAL	Малая полуось эллипса.
	R2	REAL	Большая полуось эллипса.
Выходы	ELLIPSE_A	REAL	Площадь эллипса.

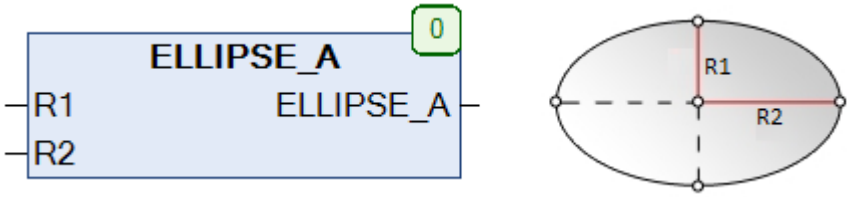


Рис. 10.9. Внешний вид функции **ELLIPSE_A** на языке CFC

Функция **ELLIPSE_A** возвращает площадь эллипса с полуосями **R1** и **R2**, вычисленную по формуле

$$\mathrm{ELLIPSE\_A} = \pi \cdot \mathrm{R1} \cdot \mathrm{R2}$$

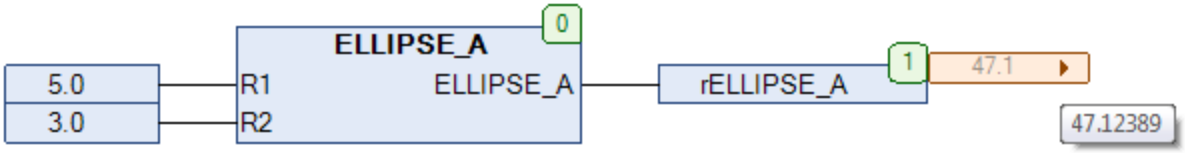


Рис. 10.10. Пример работы с функцией **ELLIPSE_A** на языке CFC

10.6. ELLIPSE_C

Тип модуля: функция	Переменная	Тип	Описание
Входы	R1	REAL	Малая полуось эллипса.
	R2	REAL	Большая полуось эллипса.
Выходы	ELLIPSE_C	REAL	Периметр эллипса.

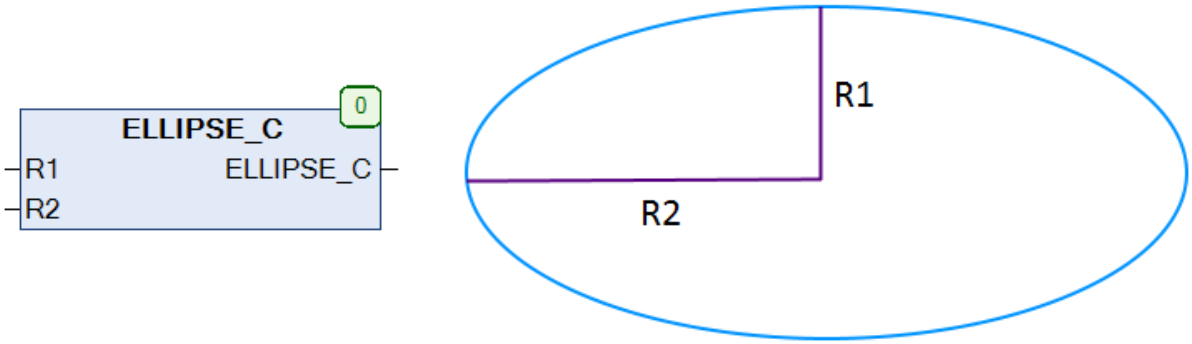


Рис. 10.11. Внешний вид функции **ELLIPSE_C** на языке CFC

Функция **ELLIPSE_C** возвращает периметр эллипса с полуосями **R1** и **R2**, вычисленный по формуле

$$\mathrm{ELLIPSE\_C} = \pi \cdot [3 \cdot (\mathrm{R1} + \mathrm{R2}) - \sqrt{(3\mathrm{R1} + \mathrm{R2}) \cdot (\mathrm{R1} + 3\mathrm{R2})}]$$

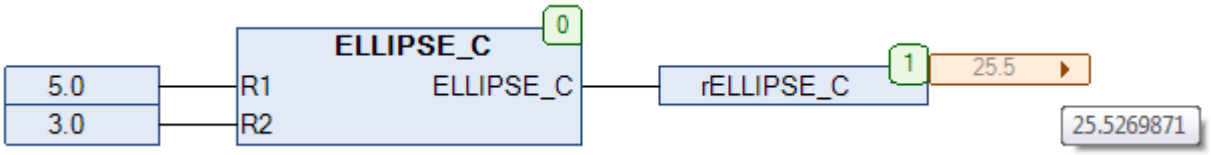


Рис. 10.12. Пример работы с функцией **ELLIPSE_C** на языке CFC

10.7. SPHERE_V

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	rx	REAL	Радиус сферы.
Выходы	SPHERE_V	REAL	Объем сферы

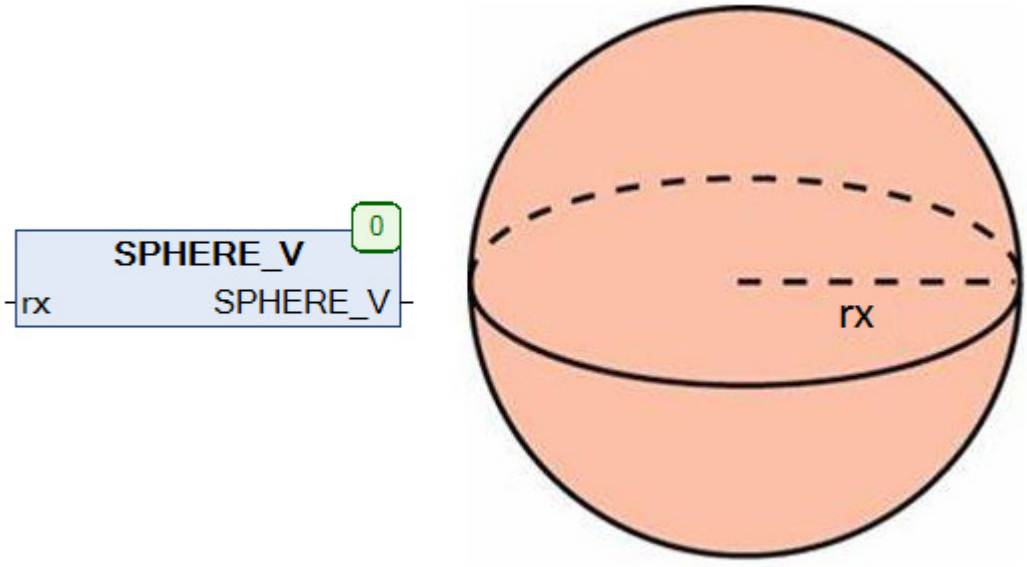


Рис. 10.13. Внешний вид функции **SPHERE_V** на языке CFC

Функция **SPHERE_V** возвращает объем сферы с радиусом **rx**, вычисленный по формуле

$$\text{SPHERE\_V} = \frac{4}{3}\pi \cdot rx^3$$

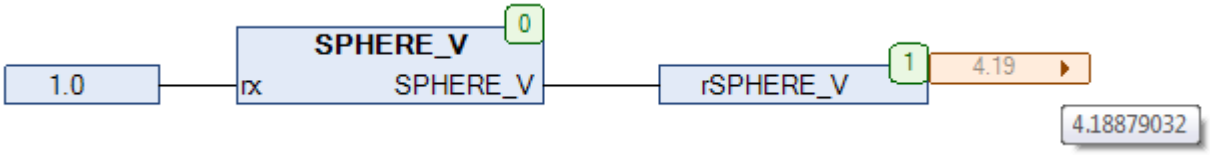


Рис. 10.14. Пример работы с функцией **SPHERE_V** на языке CFC

10.8. TRIANGLE_A

Тип модуля: функция	Переменная	Тип	Описание
Входы	S1	REAL	Длина стороны 1.
	S2	REAL	Длина стороны 2.
	A	REAL	Угол между сторонами 1 и 2, градусы.
	S3	REAL	Длина стороны 3.
Выходы	TRIANGLE_A	REAL	Площадь треугольника.

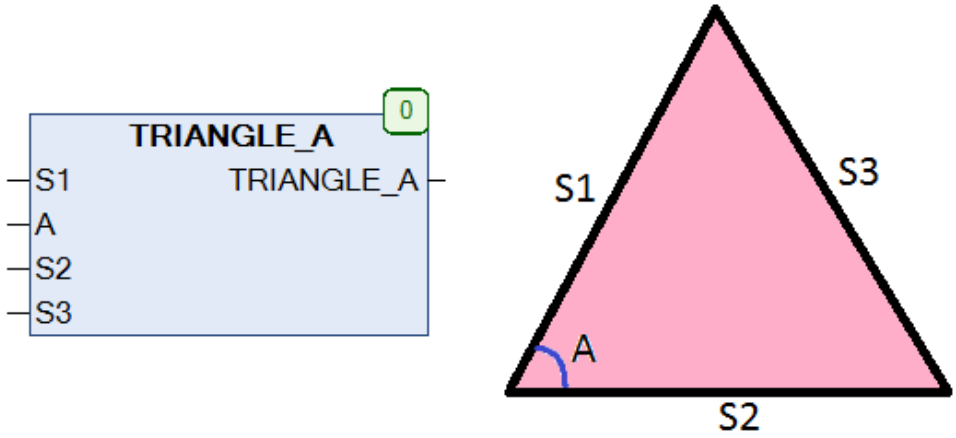


Рис. 10.15. Внешний вид функции **TRIANGLE_A** на языке CFC

Функция **TRIANGLE_A** возвращает площадь треугольника, вычисленную одним из двух способов:

1. по двум сторонам и углу между ними (**S1, S2, A**). В этом случае значение входа должно быть равно **0**.

$$\mathrm{TRIANGLE\_A} = \frac{1}{2} \cdot \mathrm{S1} \cdot \mathrm{S2} \cdot \sin(\mathrm{A})$$

2. по трем сторонам (**S1, S2, S3**). В этом случае значение входа **A** должно быть равно **0**.

$$\mathrm{TRIANGLE\_A} = \sqrt{\mathrm{p} \cdot (\mathrm{p} - \mathrm{S1}) \cdot (\mathrm{p} - \mathrm{S2}) \cdot (\mathrm{p} - \mathrm{S3})}$$

$$\mathrm{p} = \frac{\mathrm{S1} + \mathrm{S2} + \mathrm{S3}}{2}$$

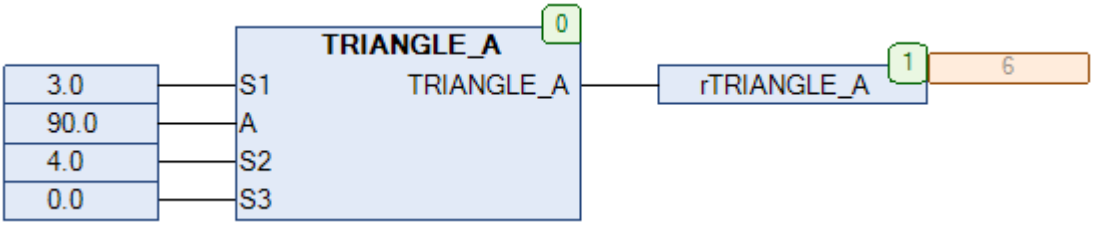


Рис. 10.16. Пример работы с функцией **TRIANGLE_A** на языке CFC

11. Операции над векторами

11.1. Вступление

Функции, описанные в данной главе, используется для работы с радиус-векторами трехмерного пространства. Координаты радиус-вектора описываются структурой VECTOR_3.

11.2. V3_ABS

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	A	VECTOR_3	Радиус-вектор.
Выходы	V3_ABS	REAL	Длина радиус-вектора.

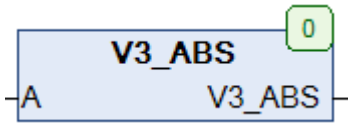


Рис. 11.1. Внешний вид функции **V3_ABS** на языке CFC

Функция **V3_ABS** возвращает модуль радиус-вектора **A** (равный длине соответствующего отрезка), вычисленный по формуле

$$\mathrm{V3\_ABS(A)} = \sqrt{(\mathrm{A.X})^2 + (\mathrm{A.Y})^2 + (\mathrm{A.Z})^2})$$

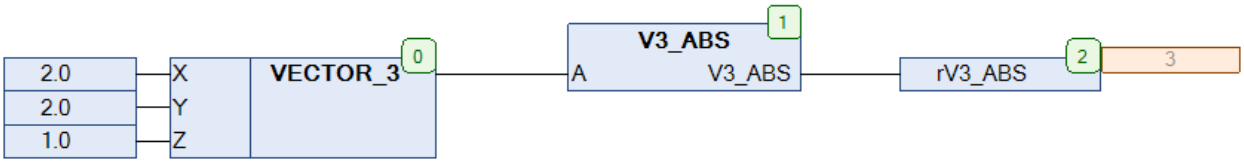


Рис. 11.2. Пример работы с функцией **V3_ABS** на языке CFC

11.3. V3_ADD

Тип модуля: функция	Переменная	Тип	Описание
Входы	A	VECTOR_3	Радиус-вектор A.
	B	VECTOR_3	Радиус-вектор B.
Выходы	V3_ADD	VECTOR_3	Сумма радиус-векторов A и B.

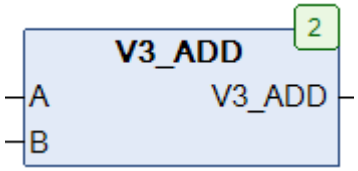


Рис. 11.3. Внешний вид функции **V3_ADD** на языке CFC

Функция **V3_ADD** возвращает сумму радиус-векторов **A** и **B** (суммирование векторов представляет собой суммирование их координат).

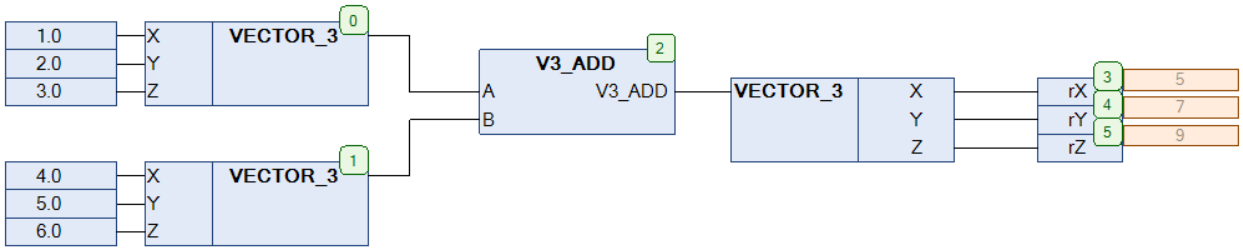


Рис. 11.4. Пример работы с функцией **V3_ADD** на языке CFC

11.4. V3_ANG

Тип модуля: функция	Переменная	Тип	Описание
Входы	A	VECTOR_3	Радиус-вектор A.
	B	VECTOR_3	Радиус-вектор B.
Выходы	V3_ANG	REAL	Угол между радиус-векторами A и B (в радианах).
Используемые модули	V3_DPRO		

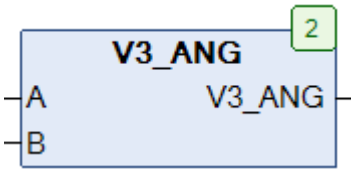


Рис. 11.5. Внешний вид функции **V3_ANG** на языке CFC

Функция **V3_ANG** возвращает значение угла (в радианах) между радиус-векторами **A** и **B**.

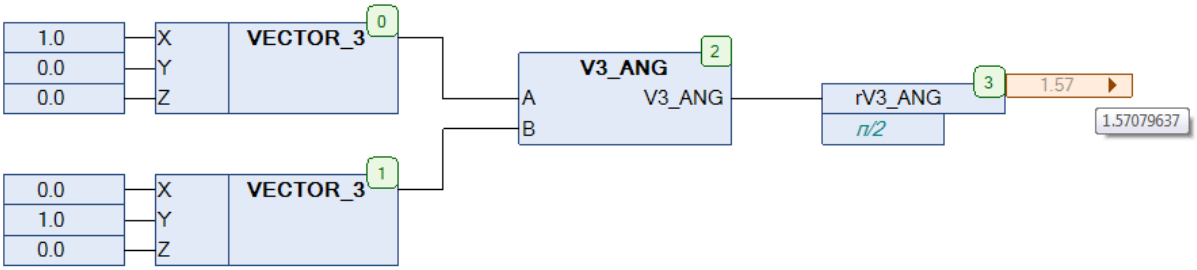


Рис. 11.6. Пример работы с функцией **V3_ANG** на языке CFC

11.5. V3_DPRO

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	A	VECTOR_3	Радиус-вектор A.
	B	VECTOR_3	Радиус-вектор B.
Выходы	V3_DPRO	VECTOR_3	Скалярное произведение радиус-векторов A и B.

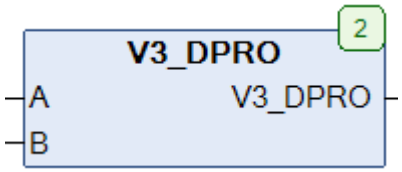


Рис. 11.7. Внешний вид функции **V3_DPRO** на языке CFC

Функция **V3_DPRO** возвращает скалярное произведение радиус-векторов **A** и **B**.

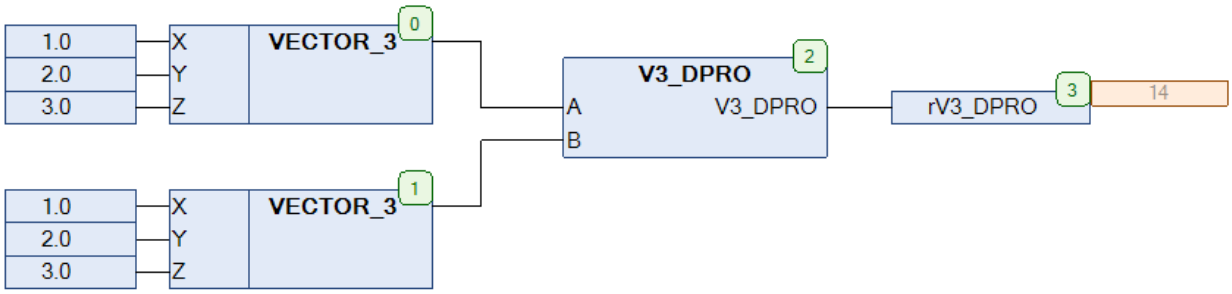


Рис. 11.8. Пример работы с функцией **V3_DPRO** на языке CFC

11.6. V3_NORM

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	A	VECTOR_3	Одномерный радиус-вектор.
Выходы	V3_NORM	VECTOR_3	Единичный радиус-вектор.
Используемые модули	V3_ABS, V3_SMUL		

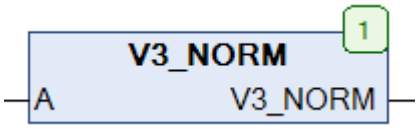


Рис. 11.9. Внешний вид функции **V3_NORM** на языке CFC

Функция **V3_NORM** конвертирует одномерный радиус-вектор **A** в единичный вектор, ориентированный в том же направлении.

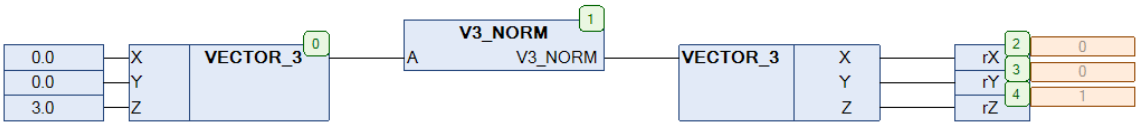


Рис. 11.10. Пример работы с функцией **V3_NORM** на языке CFC

11.7. V3_NUL

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	A	VECTOR_3	Радиус-вектор.
Выходы	V3_NUL	BOOL	Флаг «нулевой вектор».

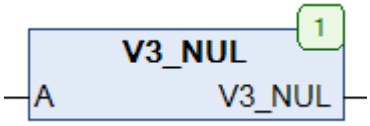


Рис. 11.11. Внешний вид функции **V3_NUL** на языке CFC

Функция **V3_NUL** возвращает **TRUE**, если радиус-вектор **A** является нулевым вектором.

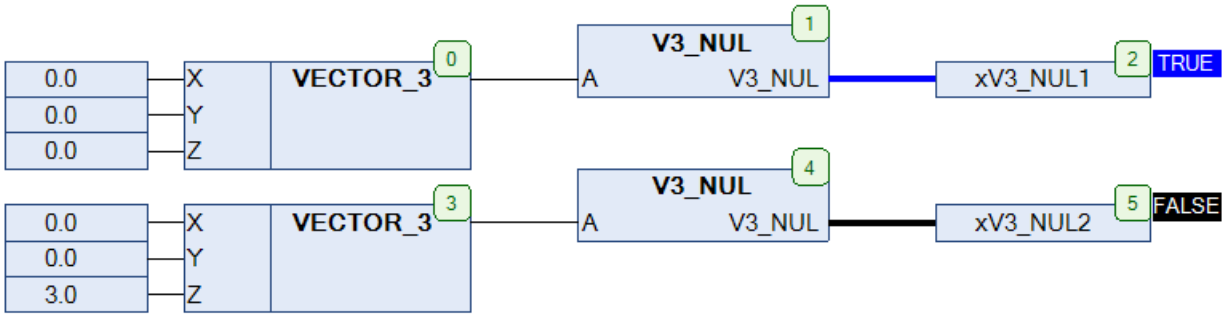


Рис. 11.12. Пример работы с функцией **V3_NUL** на языке CFC

11.8. V3_PAR

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	A	VECTOR_3	Радиус-вектор A.
	B	VECTOR_3	Радиус-вектор B.
Выходы	V3_PAR	BOOL	Флаг «коллинеарные векторы».
Используемые модули	V3_ABS, V3_XPRO		

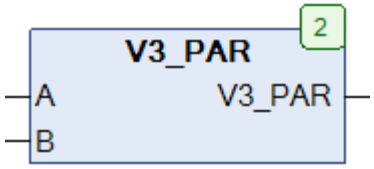


Рис. 11.13. Внешний вид функции **V3_PAR** на языке CFC

Функция **V3_PAR** возвращает **TRUE**, если радиус-векторы **A** и **B** являются коллинеарными. Нулевой вектор коллинеарен любому вектору, т.к. не имеет направления.

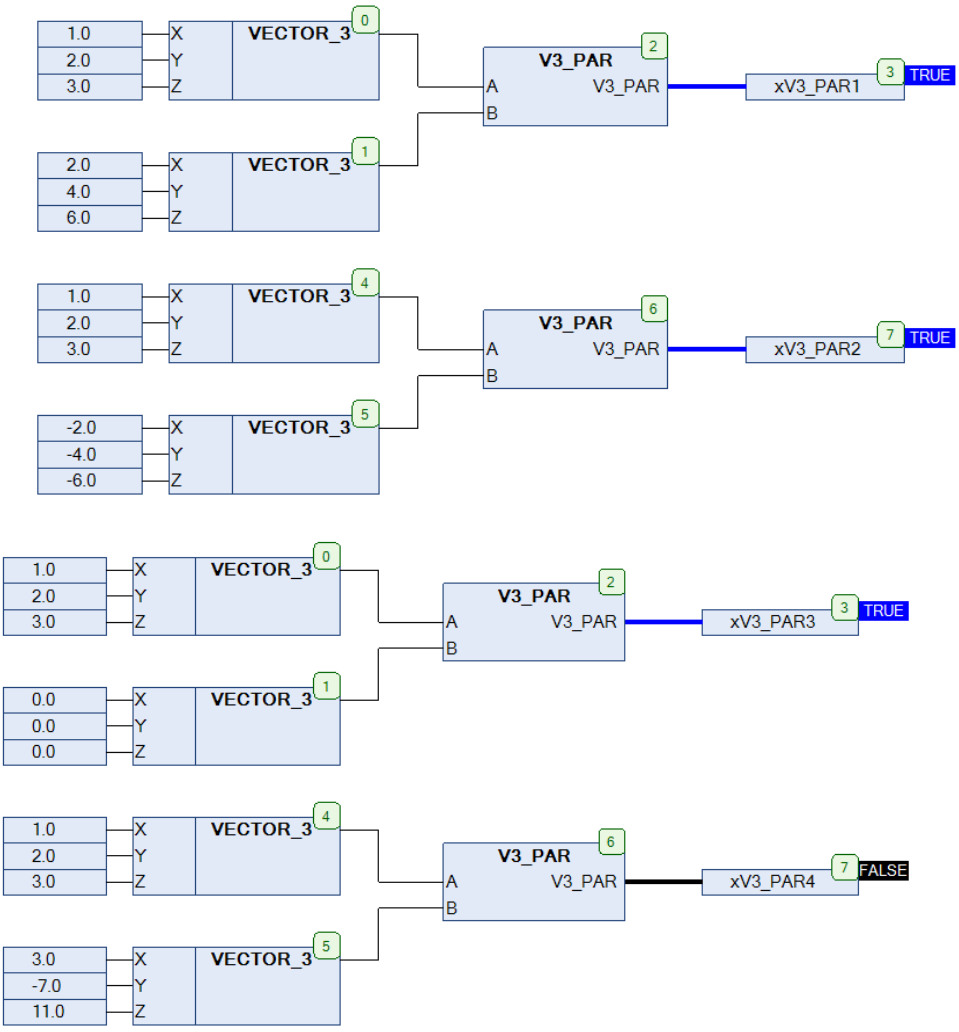


Рис. 11.14. Пример работы с функцией **V3_PAR** на языке CFC

11.9. V3_REV

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	A	VECTOR_3	Радиус-вектор.
Выходы	V3_REV	VECTOR_3	Противонаправленный радиус-вектор.

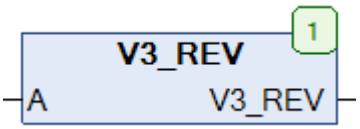


Рис. 11.15. Внешний вид функции **V3_REV** на языке CFC

Функция **V3_REV** конвертирует радиус-вектор **A** в радиус-вектор с равной длиной, но ориентированный в противоположном направлении.

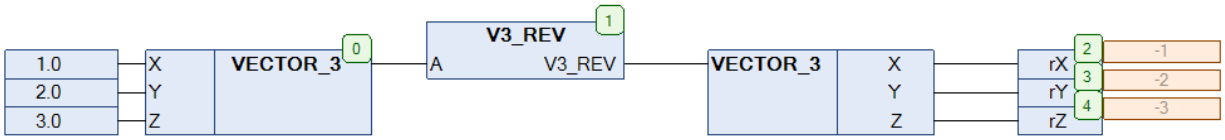


Рис. 11.16. Пример работы с функцией **V3_REV** на языке CFC

11.10. V3_SMUL

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	A	VECTOR_3	Радиус-вектор A.
	M	REAL	Число.
Выходы	V3_SMUL	VECTOR_3	Произведение радиус-векторов A и числа M.

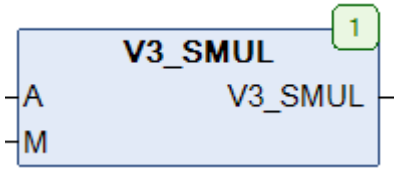


Рис. 11.17. Внешний вид функции **V3_SMUL** на языке CFC

Функция **V3_SMUL** возвращает произведение радиус-вектора **A** и числа **M**.

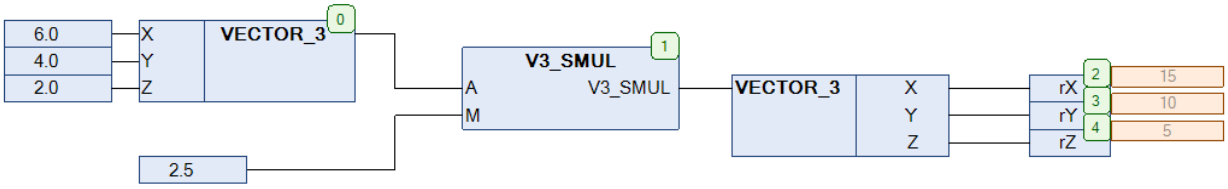


Рис. 11.18. Пример работы с функцией **V3_SMUL** на языке CFC

11.11. V3_SUB

Тип модуля: функция	Переменная	Тип	Описание
Входы	A	VECTOR_3	Радиус-вектор A.
	B	VECTOR_3	Радиус-вектор B.
Выходы	V3_SUB	VECTOR_3	Разность радиус-векторов A и B.

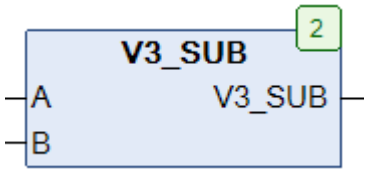


Рис. 11.19. Внешний вид функции **V3_SUB** на языке CFC

Функция **V3_SUB** возвращает разность радиус-векторов **A** и **B**.

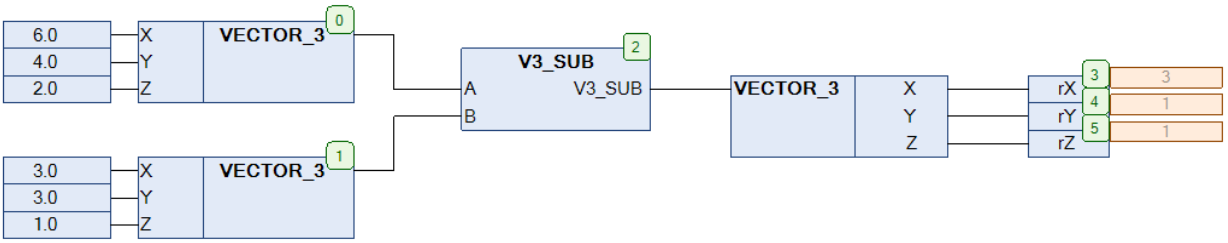


Рис. 11.20. Пример работы с функцией **V3_SUB** на языке CFC

11.12. V3_XANG

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	A	VECTOR_3	Радиус-вектор.
Выходы	V3_XANG	REAL	Угол между радиус-вектором и осью X.
Используемые модули	V3_ABS		

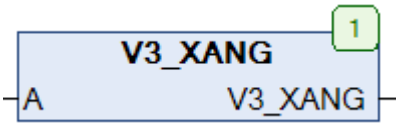


Рис. 11.21. Внешний вид функции **V3_XANG** на языке CFC

Функция **V3_XANG** возвращает угол между радиус-вектором **A** и осью **X** (в радианах).

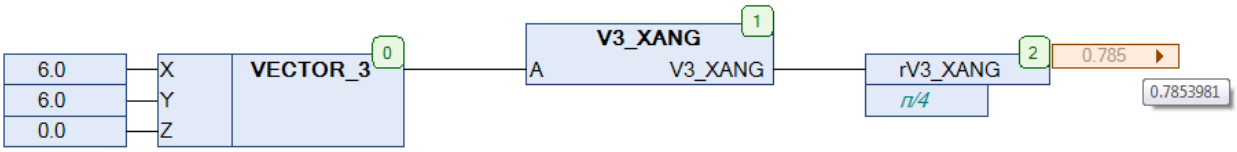


Рис. 11.22. Пример работы с функцией **V3_XANG** на языке CFC

11.13. V3_XPRO

Тип модуля: функция	Переменная	Тип	Описание
Входы	A	VECTOR_3	Радиус-вектор A.
	B	VECTOR_3	Радиус-вектор B.
Выходы	V3_XPRO	VECTOR_3	Векторное произведение радиус-векторов A и B.

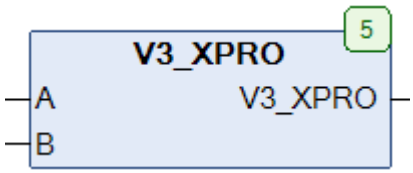


Рис. 11.23. Внешний вид функции **V3_XPRO** на языке CFC

Функция **V3_XPRO** возвращает векторное произведение радиус-векторов **A** и **B**.

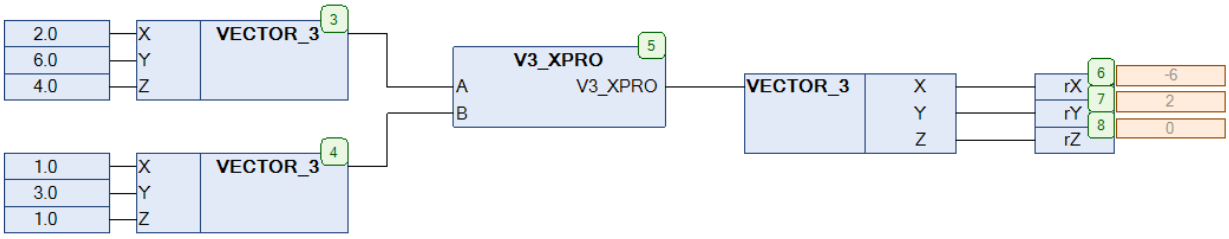


Рис. 11.24. Пример работы с функцией **V3_XPRO** на языке CFC

11.14. V3_YANG

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	A	VECTOR_3	Радиус-вектор.
Выходы	V3_YANG	REAL	Угол между радиус-вектором и осью Y.
Используемые модули	V3_ABS		

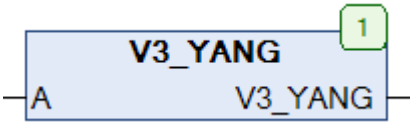


Рис. 11.25. Внешний вид функции **V3_YANG** на языке CFC

Функция **V3_YANG** возвращает угол между радиус-вектором **A** и осью Y (в радианах).

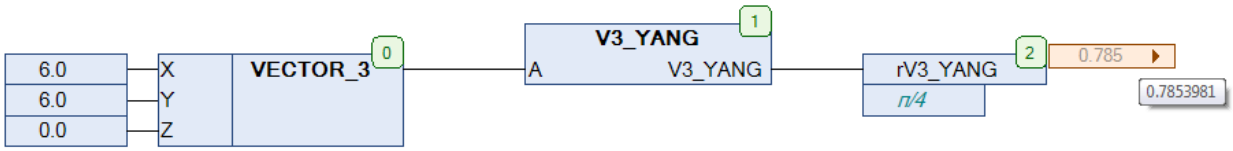


Рис. 11.26. Пример работы с функцией **V3_YANG** на языке CFC

11.15. V3_ZANG

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	A	VECTOR_3	Радиус-вектор.
Выходы	V3_ZANG	REAL	Угол между радиус-вектором и осью Z.
Используемые модули	V3_ABS		

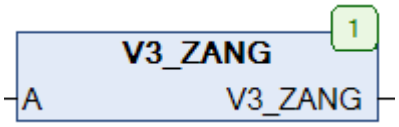


Рис. 11.27. Внешний вид функции **V3_ZANG** на языке CFC

Функция **V3_ZANG** возвращает угол между радиус-вектором **A** и осью Z (в радианах).

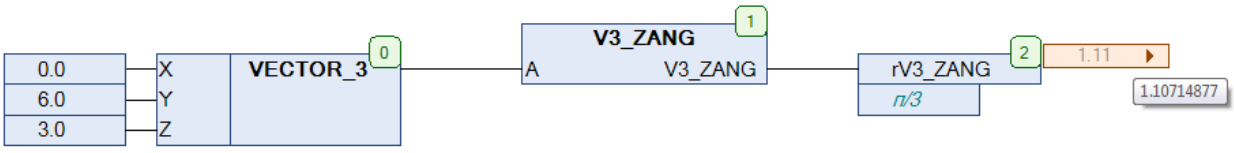


Рис. 11.28. Пример работы с функцией **V3_ZANG** на языке CFC

12. Дата и время

12.1. Вступление

Функции работы с датой и временем, включенные в библиотеку **OSCAT**, отличаются в зависимости от используемой среды разработки. Так, например, в среде **CODESYS** системное время отсчитывается от опорной точки **01.01.1970 00:00:00**, а в среде **STEP7** – от **01.01.1990 00:00:00.** Кроме того, данные функции реализованы в соответствии со стандартом ISO 8601 (например, при нумерации дней понедельник представляется как **1**, воскресенье – как **7**).

12.2. CALENDAR_CALC

Тип модуля: ФБ	**Переменная**	**Тип**	**Описание**
Входы	SPE	BOOL	Сигнал включения расчета азимута и высоты солнца над горизонтом.
	H	REAL	Высота солнца над горизонтом на восходе в градусах.
Входы-выходы	XCAL	CALENDAR	Данные календаря.
	HOLIDAYS	ARRAY [0…29] OF HOLIDAY_DATA	Список праздников.
Используемые модули	UTC_TO_LTIME, DAY_OF_DATE, HOUR, DST, YEAR_OF_DATE, MONTH_OF_DATE, DAY_OF_MONTH, DAY_OF_WEEK, WORK_WEEK, HOLIDAY, SUN_TIME, SUN_POS		

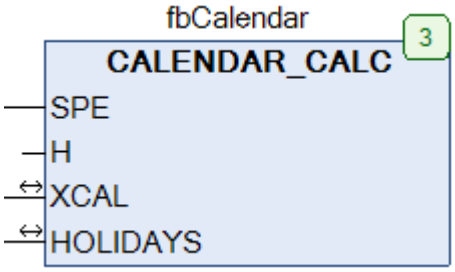


Рис. 12.1. Внешний вид ФБ **CALENDAR_CALC** на языке CFC

Функциональный блок **CALENCAR_CALC** принимает на вход-выход **XCAL** экземпляр структуры типа CALENDAR, для которого пользователь определяет значения переменных **UTC, OFFSET, DST_EN, NAME, LANGUAGE, LONGITUDE, LATITUDE** (достаточно определить только нужные переменные). На основании этих данных блок рассчитывает значения всех остальных переменных структуры. Вход **SPE** включает (при значении **TRUE**) и отключает (при значении **FALSE**) расчет параметров позиции солнца (поскольку эта операция может оказаться ресурсоемкой, особенно для ПЛК, не оснащенных FPU). При включенном расчете параметров обновление переменных происходит каждые 25 секунд, при этом точность составляет ±0.1°. Вход **H** определяет высоту

центра солнечного диска в градусах (значение по умолчанию: **-0. 83333333333**), используемую при расчете параметров позиции солнца. Вход-выход **HOLIDAYS** представляет собой массив типа HOLIDAY_DATA и используется для определения дат праздничных дней.

Ниже приведен пример использования ФБ на виртуальном контроллере **CODESYS Control Win V3**. Системное время (в UTC+0) считывается с помощью функции **SysRtcGetTime** из библиотеки **SysRtc23**. в переменную **OFFSET** записывается значение 180 (180 минут = 3 часа), что позволяет получить данные для местного времени (UTC+3).

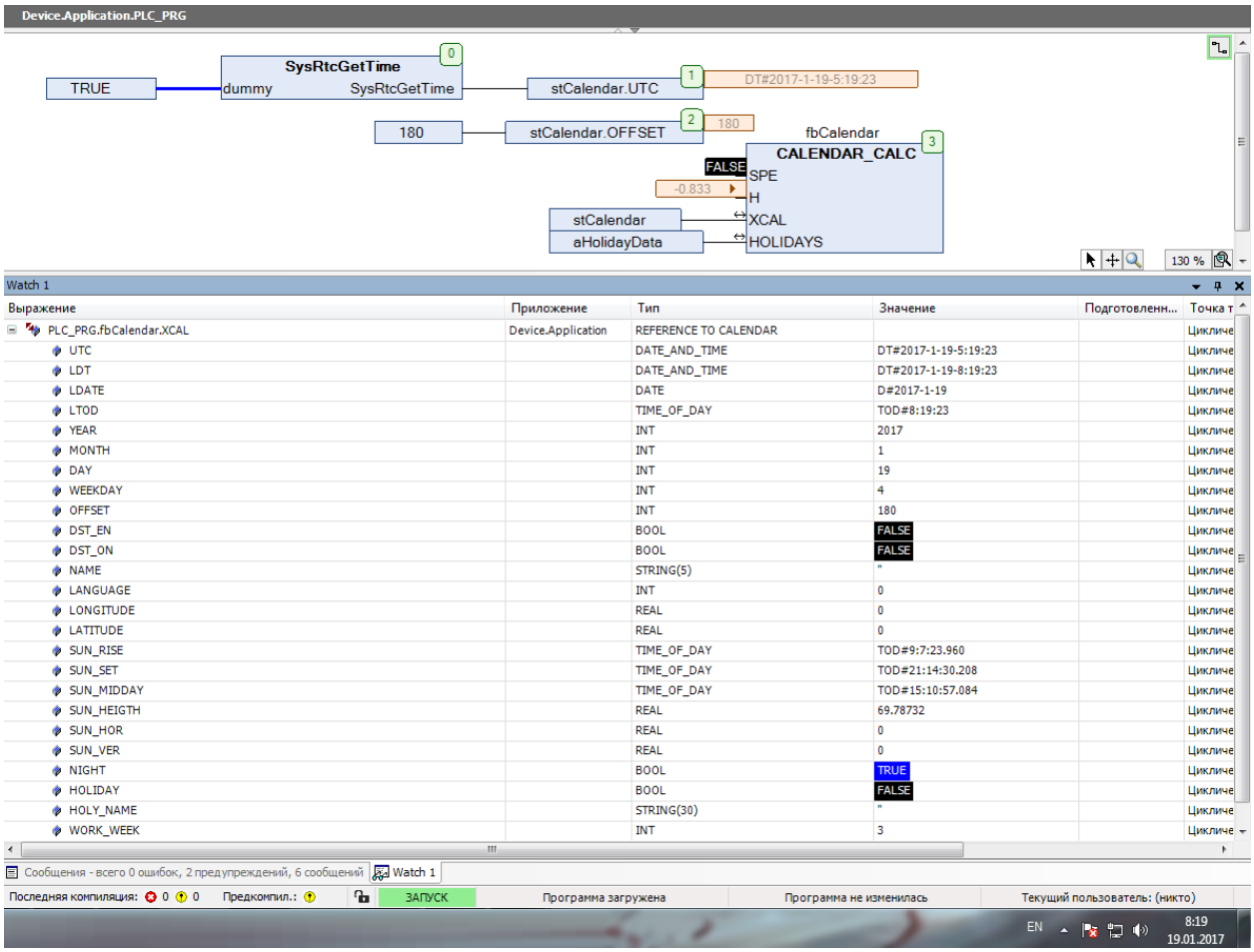


Рис. 12.2. Пример работы с ФБ **CALENDAR_CALC** на языке CFC

12.3. DATE_ADD

Тип модуля: функция	Переменная	Тип	Описание
Входы	IDATE	DATE	Исходная дата.
	D	INT	Число прибавляемых дней.
	W	INT	Число прибавляемых недель.
	M	INT	Число прибавляемых месяцев.
	Y	INT	Число прибавляемых лет.
Выходы	DATE_ADD	DATE	Вычисленная дата.
Используемые модули	YEAR_OF_DATE, MONTH_OF_DATE, DAY_OF_MONTH, SET_DATE		

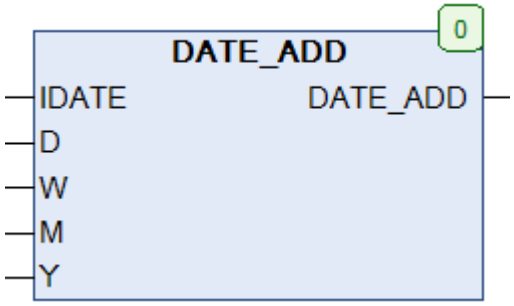


Рис. 12.3. Внешний вид функции **DATE_ADD** на языке CFC

Функция **DATE_ADD** прибавляет к начальной дате **IDATE** заданное число дней **D**, недель **W**, месяцев **M** и лет **Y**. Прибавляемые числа могут быть как положительными, так и отрицательными. При этом следует помнить, что диапазон возможных значений переменной типа **DATE** составляет 01.01.1970 – 31.12.2099.

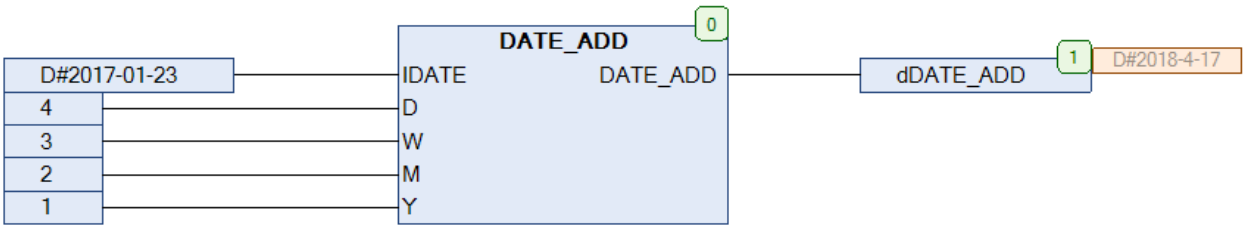


Рис. 12.4. Пример работы с функцией **DATE_ADD** на языке CFC

12.4. DAY_OF_DATE

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	IDATE	DATE	Исходная дата.
Выходы	DAY_OF_DATE	DINT	Число дней, прошедших с 01.01.1970 до исходной даты.

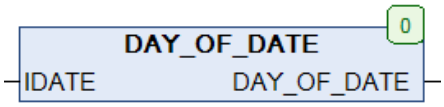


Рис. 12.5. Внешний вид функции **DAY_OF_DATE** на языке CFC

Функция **DAY_OF_DATE** возвращает число дней, прошедших с **01.01.1970** до исходной даты **IDATE.**

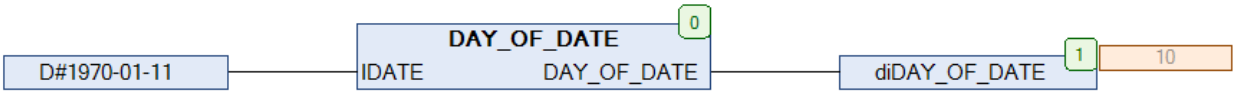


Рис. 12.6. Пример работы с функцией **DAY_OF_DATE** на языке CFC

12.5. DAY_OF_MONTH

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	IDATE	DATE	Исходная дата.
Выходы	DAY_OF_MONTH	INT	Номер дня в месяце исходной даты.
Используемые модули	DAY_OF_YEAR, LEAP_OF_DATE		

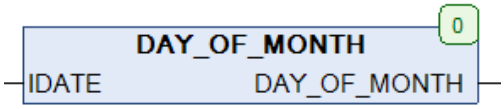


Рис. 12.7. Внешний вид функции **DAY_OF_MONTH** на языке CFC

Функция **DAY_OF_MONTH** возвращает номер дня в месяце исходной даты **IDATE**.

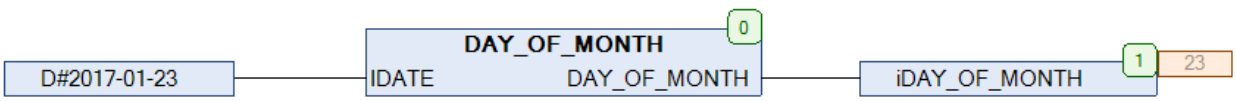


Рис. 12.8. Пример работы с функцией **DAY_OF_MONTH** на языке CFC

12.6. DAY_OF_WEEK

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	IDATE	DATE	Исходная дата.
Выходы	DAY_OF_WEEK	INT	Номер дня недели исходной даты.

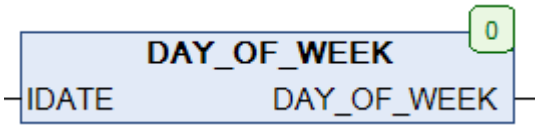


Рис. 12.9. Внешний вид функции **DAY_OF_WEEK** на языке CFC

Функция **DAY_OF_WEEK** возвращает номер дня недели исходной даты **IDATE** (**1** – понедельник, **7** – воскресение).

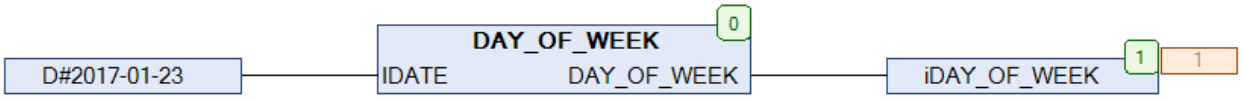


Рис. 12.10. Пример работы с функцией **DAY_OF_WEEK** на языке CFC

12.7. DAY_OF_YEAR

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	IDATE	DATE	Исходная дата.
Выходы	DAY_OF_YEAR	INT	Номер дня в году исходной даты.

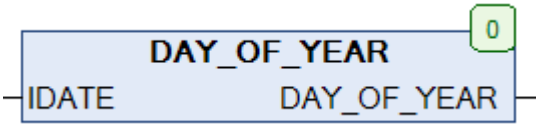


Рис. 12.11. Внешний вид функции **DAY_OF_YEAR** на языке CFC

Функция **DAY_OF_YEAR** возвращает номер дня в году исходной даты **IDATE**.

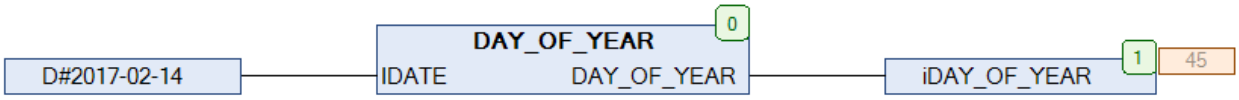


Рис. 12.12. Пример работы с функцией **DAY_OF_YEAR** на языке CFC

12.8. DAY_TO_TIME

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	IN	REAL	Число дней в формате REAL.
Выходы	DAY_TO_TIME	TIME	Число дней в формате TIME.

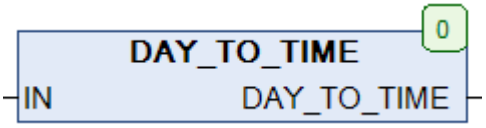


Рис. 12.13. Внешний вид функции **DAY_TO_TIME** на языке CFC

Функция **DAY_TO_TIME** конвертирует число дней **IN** типа **REAL** в значение типа **TIME**.

Рис. 12.14. Пример работы с функцией **DAY_TO_TIME** на языке CFC

12.9. DAYS_DELTA

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	DATE_1	DATE	Дата 1.
	DATE_2	DATE	Дата 2.
Выходы	DAYS_DELTA	DINT	Число дней между двумя датами.

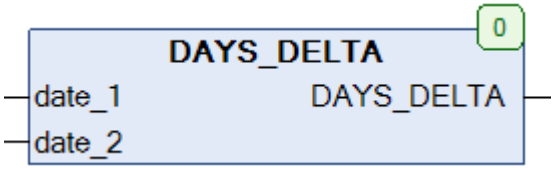


Рис. 12.15. Внешний вид функции **DAYS_DELTA** на языке CFC

Функция **DAYS_DELTA** возвращает разницу в днях между двумя датами типа **DATE**.

Рис. 12.16. Пример работы с функцией **DAYS_DELTA** на языке CFC

12.10. DAYS_IN_MONTH

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	IDATE	DATE	Исходная дата.
Выходы	DAYS_IN_MONTH	INT	Число дней в месяце исходной даты.
Используемые модули	DAY_OF_YEAR, LEAP_OF_DATE		

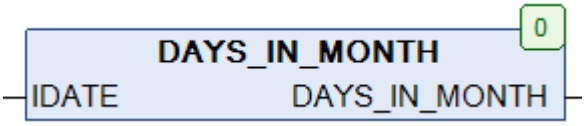


Рис. 12.17. Внешний вид функции **DAYS_IN_MONTH** на языке CFC

Функция **DAYS_IN_MONTH** возвращает число дней в месяце исходной даты **IDATE**.

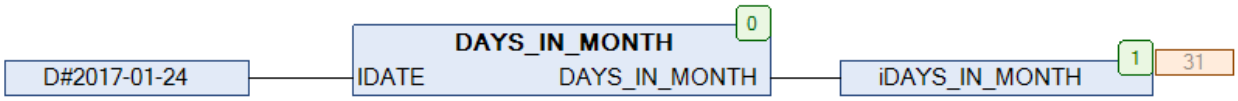


Рис. 12.18. Пример работы с функцией **DAYS_IN_MONTH** на языке CFC

12.11. DAYS_IN_YEAR

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	IDATE	DATE	Исходная дата.
Выходы	DAYS_IN_YEAR	INT	Число дней в году исходной даты.
Используемые модули	LEAP_OF_DATE		

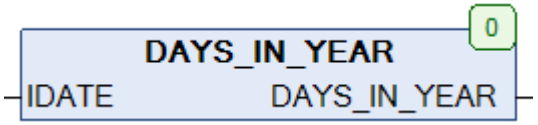


Рис. 12.19. Внешний вид функции **DAYS_IN_YEAR** на языке CFC

Функция **DAYS_IN_YEAR** возвращает число дней в году исходной даты **IDATE.**

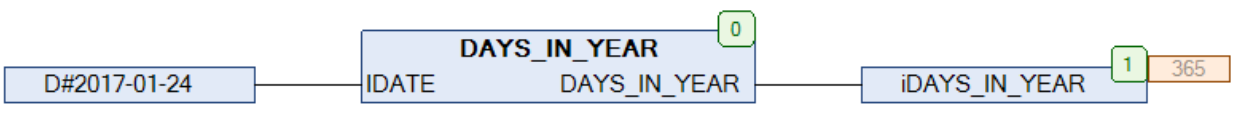


Рис. 12.20. Пример работы с функцией **DAYS_IN_YEAR** на языке CFC

12.12. DCF77

Тип модуля: ФБ	Переменная	Тип	Описание
Входы	REC	BOOL	Сигнал передатчика DCF77.
	SET	BOOL	Сигнал для инициализации часов RTC/RTC1.
	SDT	DT	Значение для инициализации часов RTC/RTC1.
	DSI	BOOL	Сигнал «летнее время» для значений инициализации.
Выходы	TP	BOOL	Флаг получения данных от передатчика.
	DS	BOOL	Флаг «летнее время».
	WDAY	INT	Номер дня недели (1 – понедельник, 7 – воскресение).
	ERROR	BOOL	Флаг «отсутствие сигнала от передатчика».
	RTC	DT	Текущее время.
	RTC1	DT	Текущее время с учетом часового пояса и летнего времени.
	MSEC	INT	Миллисекунды для текущего времени.
	SYNC	BOOL	Флаг «часы синхронизированы с DCF77».
Параметры	SYNC_TIMEOUT	TIME	Таймаут синхронизации.
	TIME_OFFSET	INT	Смещение времени (в часах) для RTC1.
	DST_EN	BOOL	Сигнал учета летнего времени для RTC1.
Используемые модули	T_PLC_MS, EVEN, SET_DT, DAY_OF_WEEK		

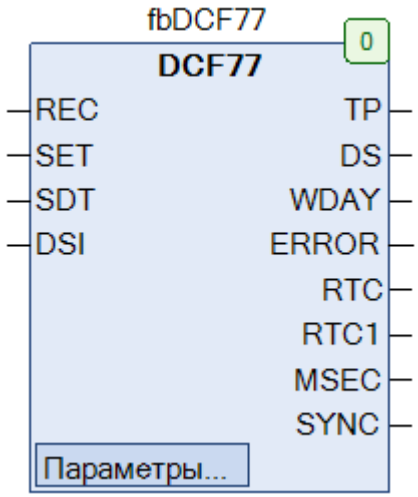


Рис. 12.21. Внешний вид ФБ **DCF77** на языке CFC

Функциональный блок **DCF77** получает на вход REC сигнал передатчика DCF77 (в виде последовательности бит) и декодирует его для извлечения текущей даты и времени. Передача сигнала осуществляется раз в минуту, для подтверждения корректности данных ФБ ожидает двух последовательных корректных передач, что может занять до трех минут.

В случае подтверждения корректности входных данных, полученные дата и время используются для синхронизации часов **RTC/RTC1**, при этом выход **TP** на один цикл принимает значение **TRUE** (для синхронизации внешних устройств). Отсчет секунд для выходов **RTC/RTC1** осуществляется с помощью часов ПЛК. **RTC** содержит дату и время в UTC+1 (исходный сигнал передатчика), а **RTC1** – дату и время со смещением **TIME_OFFSET** (в часах относительно UTC+1) и возможностью учета перехода на летнее время (с помощью параметра **DST_EN**). При переходе на летнее время выход **DS** принимает значение **TRUE**.

Выход **MSEC** содержит текущее значение миллисекунд для **RTC/RTC1,** выход **WDAY** – номер текущего дня недели (**1** – понедельник, **7** – воскресение). Выход **ERROR** принимает значение **TRUE** при отсутствии сигнала от передатчика на входе **REC**. На вход **SDT** подается дата и время, с которыми будут инициализированы часы **RTC/RTC1** в первом цикле ФБ. В дальнейшем можно осуществлять инициализацию часов по переднему фронту входа **SET**. Вход **DSI** определяет, учитывают ли значения инициализации часов переход на летнее время. Если в течение времени **SYNC_TIMEOUT** не происходило синхронизации **RTC/RTC1** с сигналом передатчика, то выход **SYNC** принимает значение **FALSE**.

12.13. DST

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	UTC	DT	Исходные дата и время.
Выходы	DST	BOOL	Флаг «летнее время».
Используемые модули	YEAR_OF_DATE		

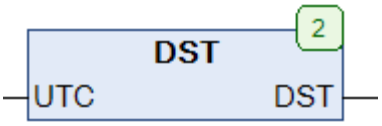


Рис. 12.22. Внешний вид функции **DST** на языке CFC

Функция **DST** возвращает T**RUE**, если дата и время **UTC** относится к летнему времени. Во всех остальных случаях возвращается **FALSE**. Функция может быть полезна при использовании часов, не поддерживающих переход на летнее время. Переход на летнее время осуществляется в 1:00 по UTC последнего воскресенья марта, возвращение на зимнее – в 1:00 по UTC последнего воскресенья октября.

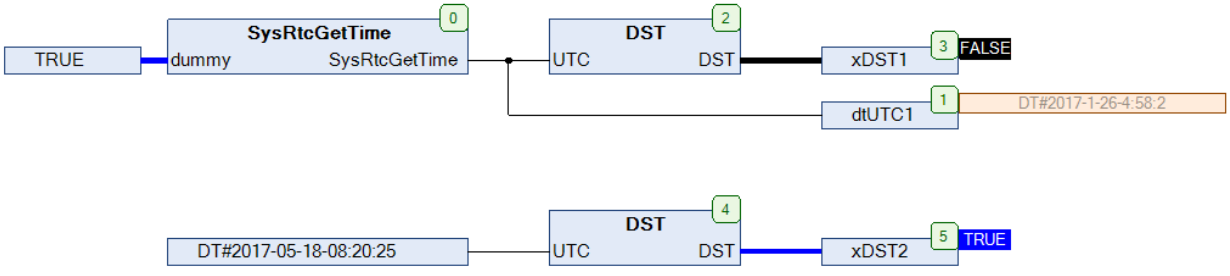


Рис. 12.23. Пример работы с функцией **DST** на языке CFC

12.14. DT2_TO_SDT

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	DI	DATE	Исходная дата.
	TI	TIME	Исходное время.
Выходы	DT2_TO_SDT	SDT	Дата и время в формате SDT.
Используемые модули	YEAR_OF_DATE, MONTH_OF_DATE, DAY_OF_MONTH, DAY_OF_WEEK		

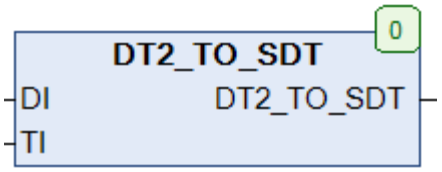


Рис. 12.24. Внешний вид функции **DT2_TO_SDT** на языке CFC

Функция **DT2_TO_SDT** конвертирует исходную дату **DI** и время **TI** в структуру SDT, содержащую разряды времени в виде переменных типа **INT.**

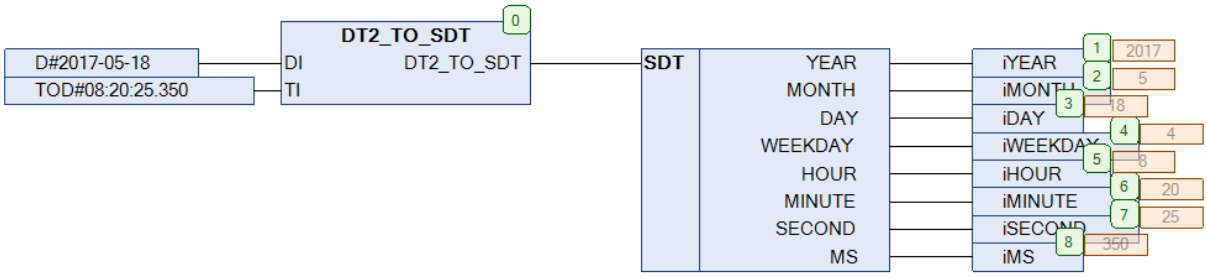


Рис. 12.25. Пример работы с функцией **DT2_TO_SDT** на языке CFC

12.15. DT_TO_SDT

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	DTI	DT	Исходные дата и время.
Выходы	DT_TO_SDT	SDT	Дата и время в формате SDT.
Используемые модули	YEAR_OF_DATE, MONTH_OF_DATE, DAY_OF_MONTH, DAY_OF_WEEK		

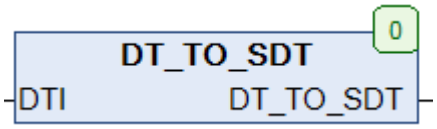


Рис. 12.26. Внешний вид функции **DT_TO_SDT** на языке CFC

Функция **DT_TO_SDT** конвертирует исходную дату и время **DTI** в структуру SDT, содержащую разряды времени в виде переменных типа **INT**.

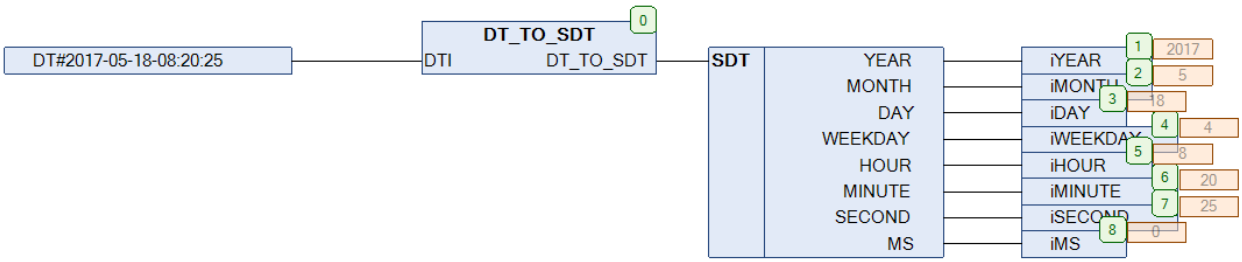


Рис. 12.27. Пример работы с функцией **DT_TO_SDT** на языке CFC

12.16. EASTER

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	year	INT	Год.
Выходы	EASTER	DATE	Дата Пасхи для заданного года.
Используемые модули	SET_DATE		

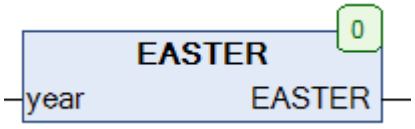


Рис. 12.28. Внешний вид функции **EASTER** на языке CFC

Функция **EASTER** возвращает дату Пасхи для заданного года. Дата Пасхи является опорной для большинства религиозных праздников.

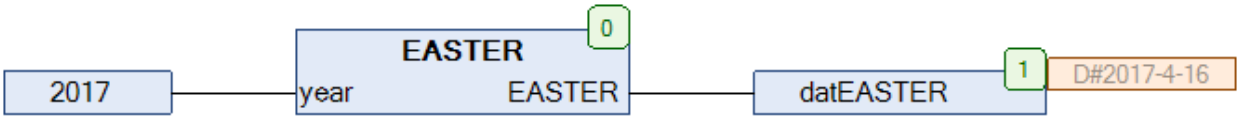


Рис. 12.29. Пример работы с функцией **EASTER** на языке CFC

12.17. EVENTS

Тип модуля: ФБ	**Переменная**	**Тип**	**Описание**
Входы	DATE_IN	DATE	Текущая дата.
	ENA	BOOL	Сигнал включения ФБ.
Выходы	Y	BOOL	Флаг «сегодня мероприятие».
	NAME	STRING(30)	Название мероприятия.
Входы-выходы	ELIST	ARRAY [0..49] OF HOLIDAY_DATA	Список мероприятий.
Используемые модули	DAY_OF_DATE, YEAR_OF_DATE, SET_DATE		

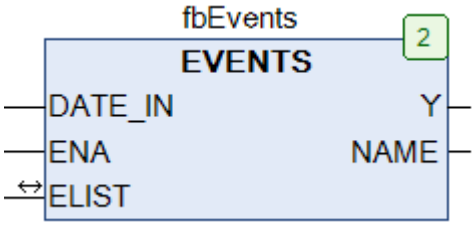


Рис. 12.30. Внешний вид ФБ **EVENTS** на языке CFC

Функциональный блок **EVENTS** используется для создания календаря мероприятий и контроля их наступления. Информация о мероприятиях содержится в массиве **ELIST**, каждый элемент которого представляет собой запись типа HOLIDAY_DATA. На вход **DATE_IN** поступает текущая дата. Вход **ENA** управляет работой блока (**TRUE** – блок включен, **FALSE** – отключен). Если текущей дате соответствует какое-либо мероприятие, то выход **Y** принимает значение **TRUE**, а на выходе **NAME** отображается название этого мероприятия. Каждой дате может соответствовать только одно мероприятие.

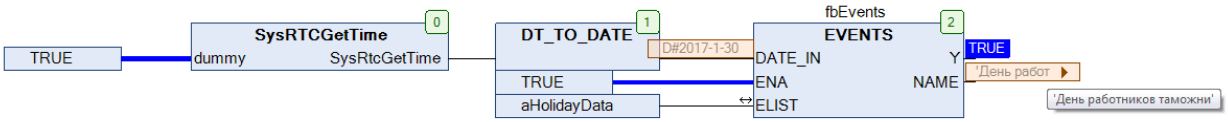


Рис. 12.31. Пример работы с ФБ **EVENTS** на языке CFC

12.18. HOLIDAY

Тип модуля: ФБ	**Переменная**	**Тип**	**Описание**
Входы	DATE_IN	DATE	Текущая дата.
	LANGU	INT	Язык приложения.
	FRIDAY	BOOL	Сигнал «пятница – праздник».
	SATURDAY	BOOL	Сигнал «суббота – праздник».
	SUNDAY	BOOL	Сигнал «воскресенье – праздник».
Выходы	Y	BOOL	Флаг «сегодня праздник».
	NAME	STRING(30)	Название праздника.
Входы-выходы	HOLIDAYS	ARRAY [0..29] OF HOLIDAY_DATA	Список праздников.
Используемые модули	YEAR_OF_DATE, EASTER, DAY_OF_WEEK, SET_DATE, DATE_ADD		

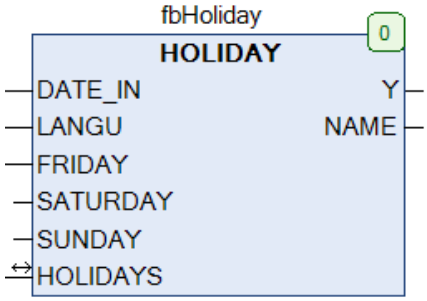


Рис. 12.32. Внешний вид ФБ **HOLIDAY** на языке CFC

Функциональный блок **HOLIDAY** используется для создания календаря праздников и контроля их наступления. Информация о мероприятиях содержится в массиве **HOLIDAYS**, каждый элемент которого представляет собой запись типа HOLIDAY_DATA. На вход **DATE_IN** поступает текущая дата. Если текущей дате соответствует какой-либо праздник, то выход **Y** принимает значение **TRUE**, а на выходе **NAME** отображается название этого праздника. Если входы **FRIDAY**, **SATURDAY** и **SUNDAY** имеют значение **TRUE**, то соответствующие дни недели также считаются праздниками. Для таких случаев вход **LANGU** определяет язык, на котором будет отображаться название дня недели на выходе **NAME**.

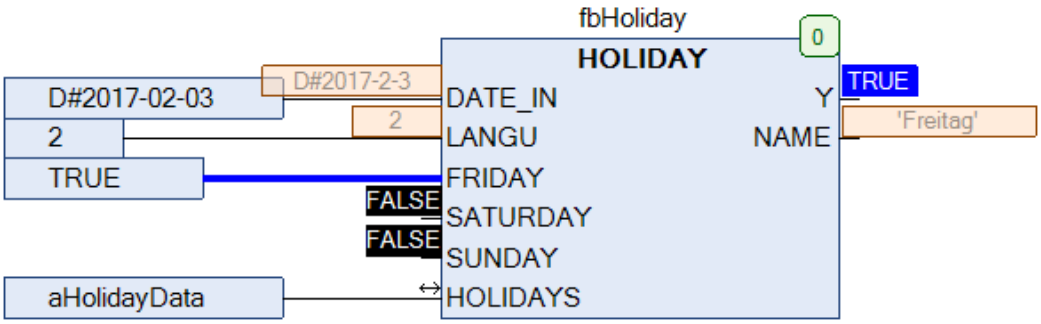


Рис. 12.33. Пример работы с ФБ **HOLIDAY** на языке CFC

12.19. HOUR

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	itod	TOD	Исходное время суток.
Выходы	HOUR	INT	Число часов.

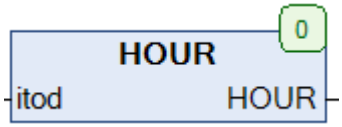


Рис. 12.34. Внешний вид функции **HOUR** на языке CFC

Функция **HOUR** возвращает число часов, вырезанное из исходного значения времени суток **itod** типа **TOD**.

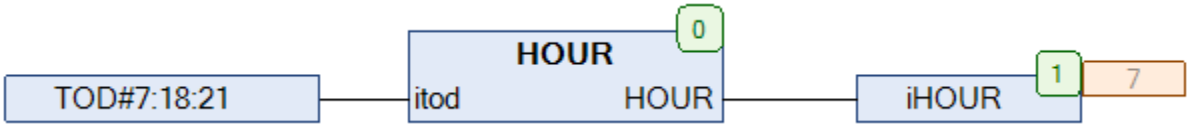


Рис. 12.35. Пример работы с функцией **HOUR** на языке CFC

12.20. HOUR_OF_DT

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	XDT	DT	Исходные дата и время.
Выходы	HOUR_OF_DT	INT	Число часов.

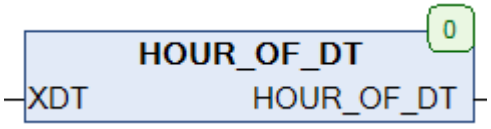


Рис. 12.36. Внешний вид функции **HOUR_OF_DT** на языке CFC

Функция **HOUR_OF_DT** возвращает число часов, вырезанное из исходного значения даты и времени **XDT** типа **DT**.

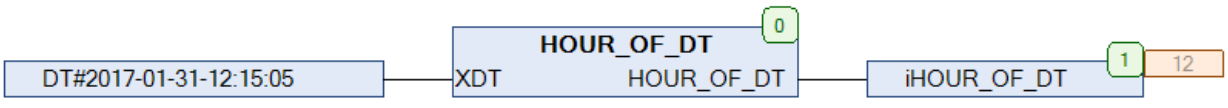


Рис. 12.37. Пример работы с функцией **HOUR_OF_DT** на языке CFC

12.21. HOUR_TO_TIME

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	IN	REAL	Количество часов.
Выходы	HOUR_TO_TIME	TIME	Время.

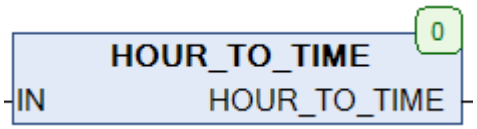


Рис. 12.38. Внешний вид функции **HOUR_TO_TIME** на языке CFC

Функция **HOUR_TO_TIME** конвертирует заданное количество часов **IN** типа **REAL** (в виде числа с плавающей точкой) в значение переменной типа **TIME**.

Рис. 12.39. Пример работы с функцией **HOUR_TO_TIME** на языке CFC

12.22. HOUR_TO_TOD

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	IN	REAL	Количество часов.
Выходы	HOUR_TO_TOD	TOD	Время суток.

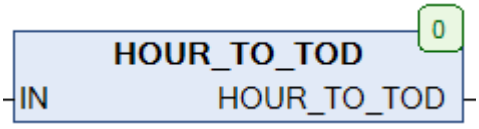


Рис. 12.40. Внешний вид функции **HOUR_TO_TOD** на языке CFC

Функция **HOUR_TO_TOD** конвертирует заданное количество часов **IN** типа **REAL** (в виде числа с плавающей точкой) в значение переменной типа **TOD**.

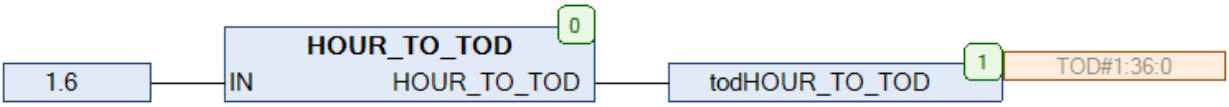


Рис. 12.41. Пример работы с функцией **HOUR_TO_TOD** на языке CFC

12.23. JD2000

Тип модуля: функция	Переменная	Тип	Описание
Входы	DTI	DT	Заданные дата и время по грегорианскому календарю.
Выходы	JD2000	REAL	Заданные дата и время по юлианскому календарю.

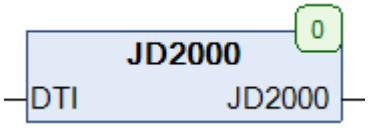


Рис. 12.42. Внешний вид функции **JD2000** на языке CFC

Функция **JD2000** конвертирует значение заданной даты и времени по грегорианскому календарю **DTI** в значение по юлианскому календарю – в виде числа дней (с плавающей точкой), прошедших с полудня 1 января 2000 года.

Точкой отсчета для юлианского календаря является полдень 1 января 4713 г. до нашей эры. Поскольку в настоящий момент количество прошедших с того момента дней превышает диапазон типа **REAL**, то данная функция использует в качестве опорной точки полдень 1 января 2000 года нашей эры, которому соответствует юлианская дата JD 2451544.5. При суммировании данного значения с результатом функции можно получить абсолютную дату по юлианскому календарю.

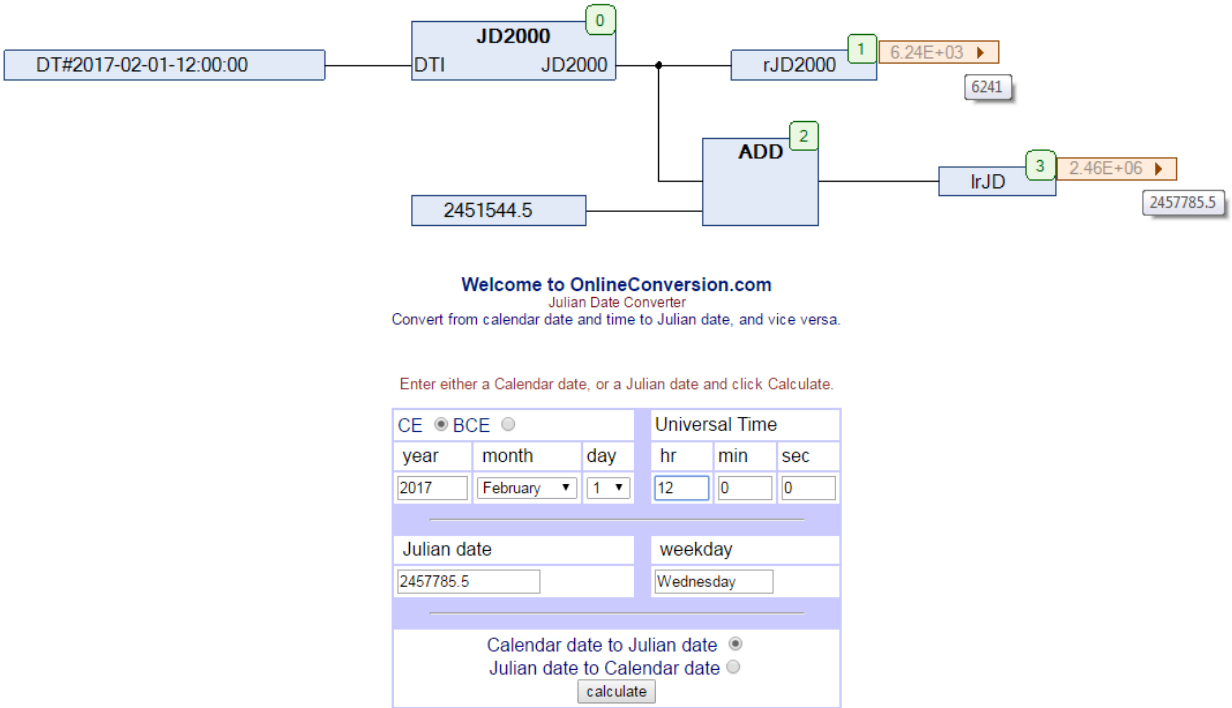


Рис. 12.43. Пример работы с функцией **JD2000** на языке CFC

12.24. LEAP_DAY

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	IDATE	DATE	Заданная дата.
Выходы	LEAP_DAY	BOOL	Флаг «високосный день».

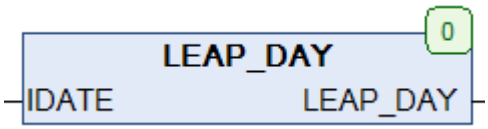


Рис. 12.44. Внешний вид функции **LEAP_DAY** на языке CFC

Функция **LEAP_DAY** возвращает **TRUE**, если заданная дата **IDATE** является високосным днем (29-м февраля). Во всех остальных случаях функция возвращает **FALSE**.

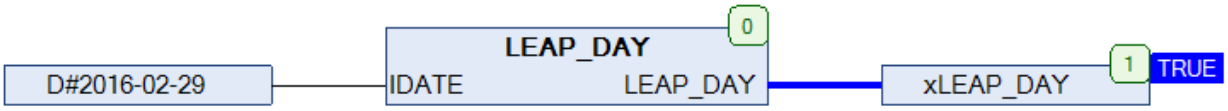


Рис. 12.45. Пример работы с функцией **LEAP_DAY** на языке CFC

12.25. LEAP_OF_DATE

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	IDATE	DATE	Заданная дата.
Выходы	LEAP_OF_DATE	BOOL	Флаг «високосный год».

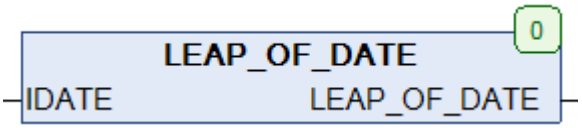


Рис. 12.46. Внешний вид функции **LEAP_OF_DATE** на языке CFC

Функция **LEAP_OF_DATE** возвращает значение **TRUE**, если год заданной даты **IDATE** является високосным. Во всех остальных случаях функция возвращает **FALSE**.

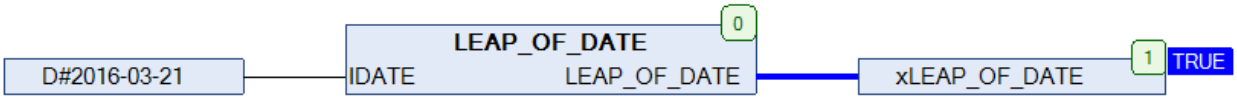


Рис. 12.47. Пример работы с функцией **LEAP_OF_DATE** на языке CFC

12.26. LEAP_YEAR

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	yr	INT	Заданный год.
Выходы	LEAP_YEAR	BOOL	Флаг «високосный год».

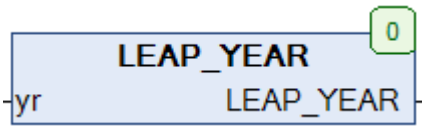


Рис. 12.48. Внешний вид функции **LEAP_YEAR** на языке CFC

Функция **LEAP_YEAR** возвращает **TRUE**, если заданный год **yr** является високосным. Во всех остальных случаях функция возвращает **FALSE**.

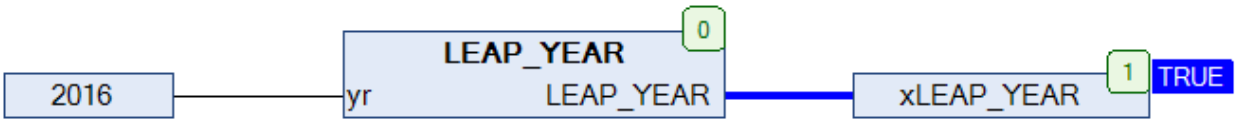


Рис. 12.49. Пример работы с функцией **LEAP_YEAR** на языке CFC

12.27. LTIME_TO_UTC

Тип модуля: функция	Переменная	Тип	Описание
Входы	LTIME	DT	Местное дата и время.
	DST	BOOL	Сигнал учета летнего времени.
	TIME_ZONE_OFFSET	INT	Смещение местного времени относительно UTC в минутах.
Выходы	LEAP_YEAR	DT	Дата и время в UTC.

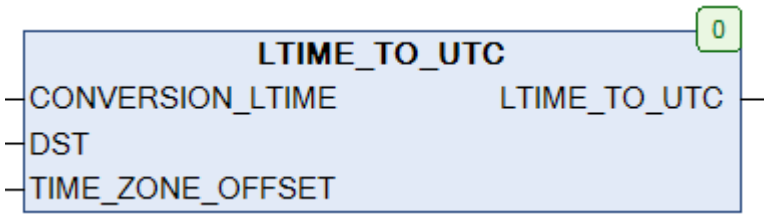


Рис. 12.50. Внешний вид функции **LTIME_TO_UTC** на языке CFC

Функция **LTIME_TO_UTC** конвертирует значение местного даты и времени, смещенного относительно UTC на число минут, определяемым значением переменной **TIME_ZONE_OFFSET.** Если вход **DST** имеет значение **TRUE**, то при конвертации учитывается переход на летнее время. Переход на летнее время регламентируется по-разному в различных странах; функция предполагает, что при переходе на летнее время смещение местного времени относительно **UTC** увеличивается на один час.

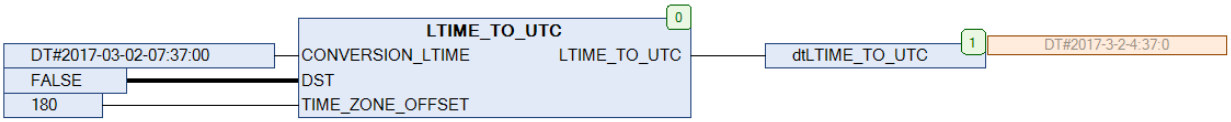


Рис. 12.51. Пример работы с функцией **LTIME_TO_UTC** на языке CFC

12.28. MINUTE

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	itod	TOD	Заданное время.
Выходы	MINUTE	INT	Число минут.

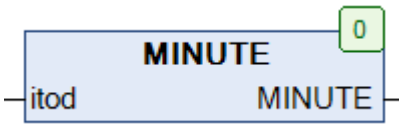


Рис. 12.52. Внешний вид функции **MINUTE** на языке CFC

Функция **MINUTE** возвращает число минут, вырезанное из исходного значения времени суток **itod** типа **TOD**.

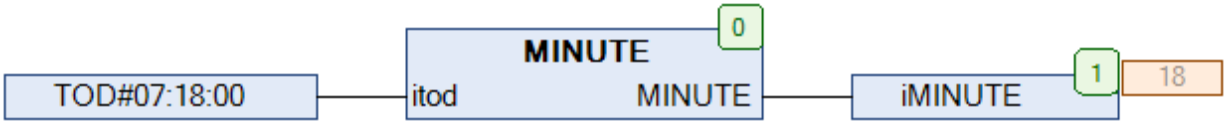


Рис. 12.53. Пример работы с функцией **MINUTE** на языке CFC

12.29. MINUTE_OF_DT

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	XDT	DT	Заданные дата и время.
Выходы	MINUTE_OF_DT	INT	Число минут.

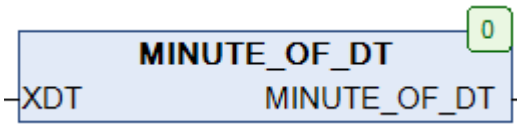


Рис. 12.54. Внешний вид функции **MINUTE_OF_DT** на языке CFC

Функция **MINUTE_OF_DT** возвращает число минут, вырезанное из исходного значения даты и времени **XDT** типа **DT**.

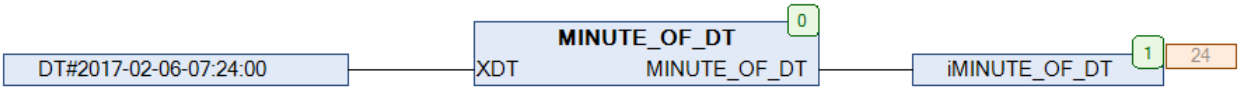


Рис. 12.55. Пример работы с функцией **MINUTE_OF_DT** на языке CFC

12.30. MINUTE_TO_TIME

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	IN	REAL	Количество минут.
Выходы	MINITE_TO_TIME	TIME	Время.

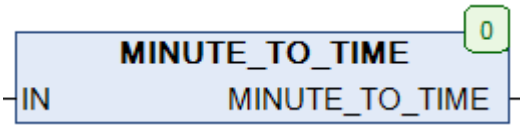


Рис. 12.56. Внешний вид функции **MINUTE_TO_TIME** на языке CFC

Функция **MINUTE_TO_TIME** конвертирует заданное количество минут **IN** типа **REAL** (в виде числа с плавающей точкой) в значение переменной типа **TIME**.

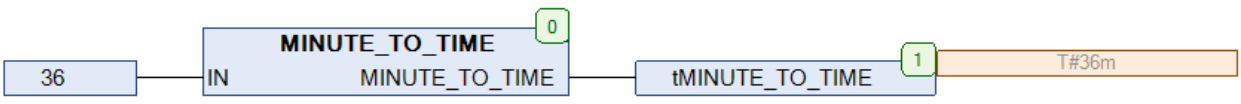


Рис. 12.57. Пример работы с функцией **MINUTE _TO_TIME** на языке CFC

12.31. MONTH_BEGIN

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	idate	DATE	Заданная дата.
Выходы	MONTH_BEGIN	DATE	Дата первого дня заданного месяца.
Используемые модули	DAY_OF_MONTH		

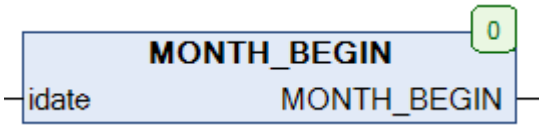


Рис. 12.58. Внешний вид функции **MONTH_BEGIN** на языке CFC

Функция **MONTH_BEGIN** возвращает дату первого дня месяца, заданного переменной **idate** типа **DATE**.

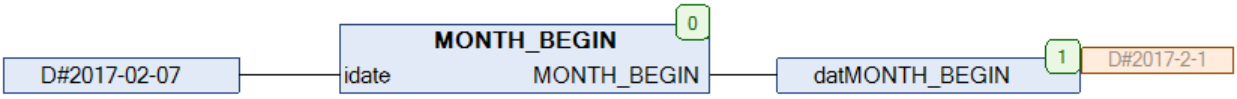


Рис. 12.59. Пример работы с функцией **MONTH_BEGIN** на языке CFC

12.32. MONTH_END

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	idate	DATE	Заданная дата.
Выходы	MONTH_END	DATE	Дата последнего дня заданного месяца.
Используемые модули	SET_DATE, YEAR_OF_DATE, MONTH_OF_DATE		

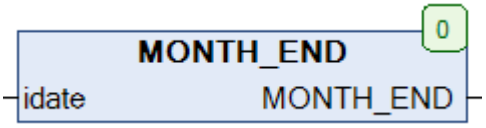


Рис. 12.60. Внешний вид функции **MONTH_END** на языке CFC

Функция **MONTH_END** возвращает дату последнего дня месяца, заданного переменной **idate** типа **DATE**.

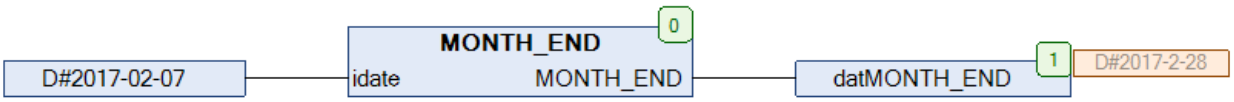


Рис. 12.61. Пример работы с функцией **MONTH_END** на языке CFC

12.33. MONTH_OF_DATE

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	idate	DATE	Заданная дата.
Выходы	MONTH_OF_DATE	INT	Число месяцев.
Используемые модули	DAY_OF_YEAR, LEAP_OF_DATE		

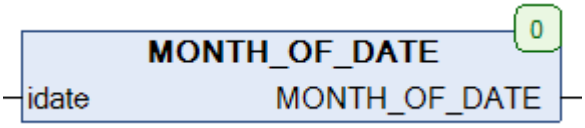


Рис. 12.62. Внешний вид функции **MONTH_OF_DATE** на языке CFC

Функция **MONTH_OF_DATE** возвращает число месяцев, вырезанное из исходного значения даты **idate** типа **DATE**.

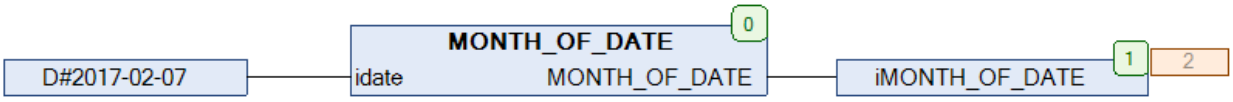


Рис. 12.63. Пример работы с функцией **MONTH_OF_DATE** на языке CFC

12.34. MULTIME

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	t	TIME	Заданное время.
	M	REAL	Коэффициент.
Выходы	MULTIME	TIME	Отмасштабированное значение времени.

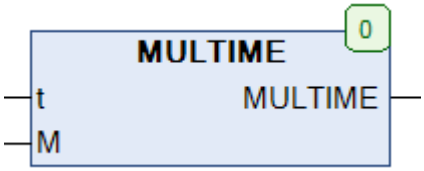


Рис. 12.64. Внешний вид функции **MULTIME** на языке CFC

Функция **MULTIME** возвращает произведение заданного времени **T** типа **TIME** на коэффициент **M** типа **REAL**.

Рис. 12.65. Пример работы с функцией **MULTIME** на языке CFC

12.35. PERIOD

Тип модуля: функция	Переменная	Тип	Описание
Входы	dx1	DATE	Нижняя граница интервала.
	dx	DATE	Проверяемая дата.
	dx2	DATE	Верхняя граница интервала.
Выходы	PERIOD	BOOL	Флаг принадлежности даты интервалу.
Используемые модули	DAY_OF_YEAR, LEAP_OF_DATE		

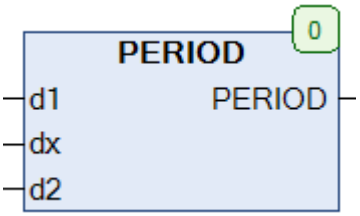


Рис. 12.66. Внешний вид функции **PERIOD** на языке CFC

Функция **PERIOD** возвращает **TRUE**, если дата **dx** принадлежит интервалу **[d1, d2]** (включая границы интервала и при условии, что d1<d2). Функция не учитывает значение года, т.е. может использоваться только для дат, относящихся к одному году.

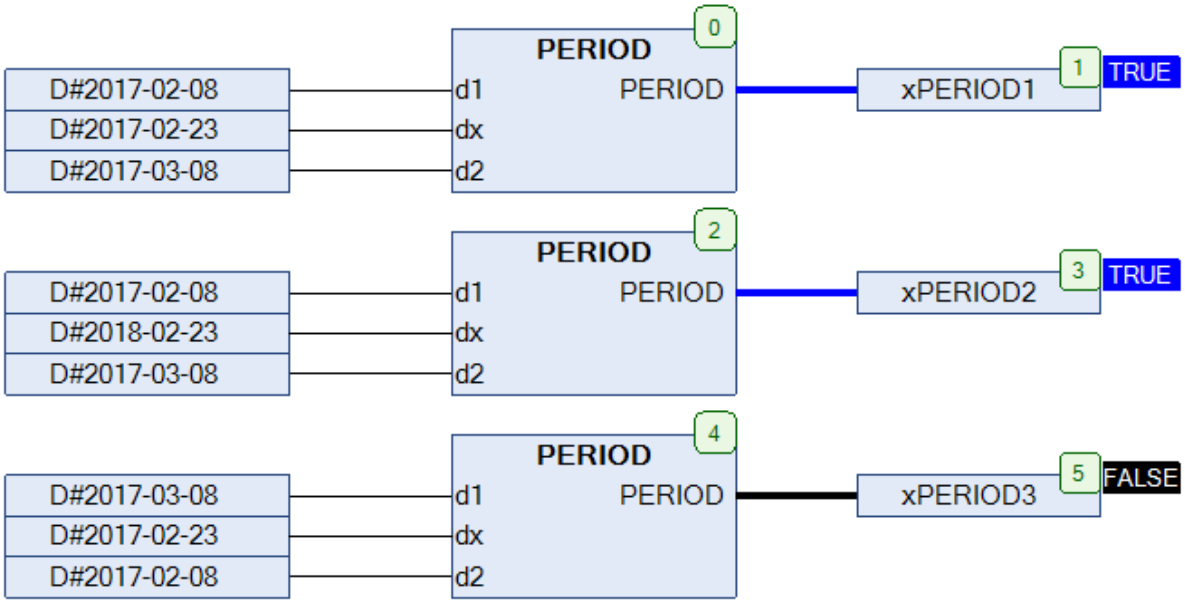


Рис. 12.67. Пример работы с функцией **PERIOD** на языке CFC

12.36. PERIOD2

Тип модуля: функция	Переменная	Тип	Описание
Входы	DP	ARRAY [0..3, 0..1] OF DATE	Интервалы проверки.
	dx	DATE	Проверяемая дата.
Выходы	PERIOD2	BOOL	Флаг принадлежности даты одному из интервалов.

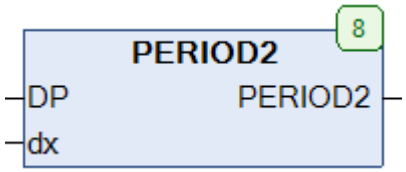


Рис. 12.68. Внешний вид функции **PERIOD2** на языке CFC

Функция **PERIOD2** возвращает **TRUE**, если дата **dx** принадлежит одному из 4-х интервалов, определенных массивом дат **DP**. В отличие от функции **PERIOD**, функция **PERIOD2** учитывает год даты при проверке принадлежности интервалу.

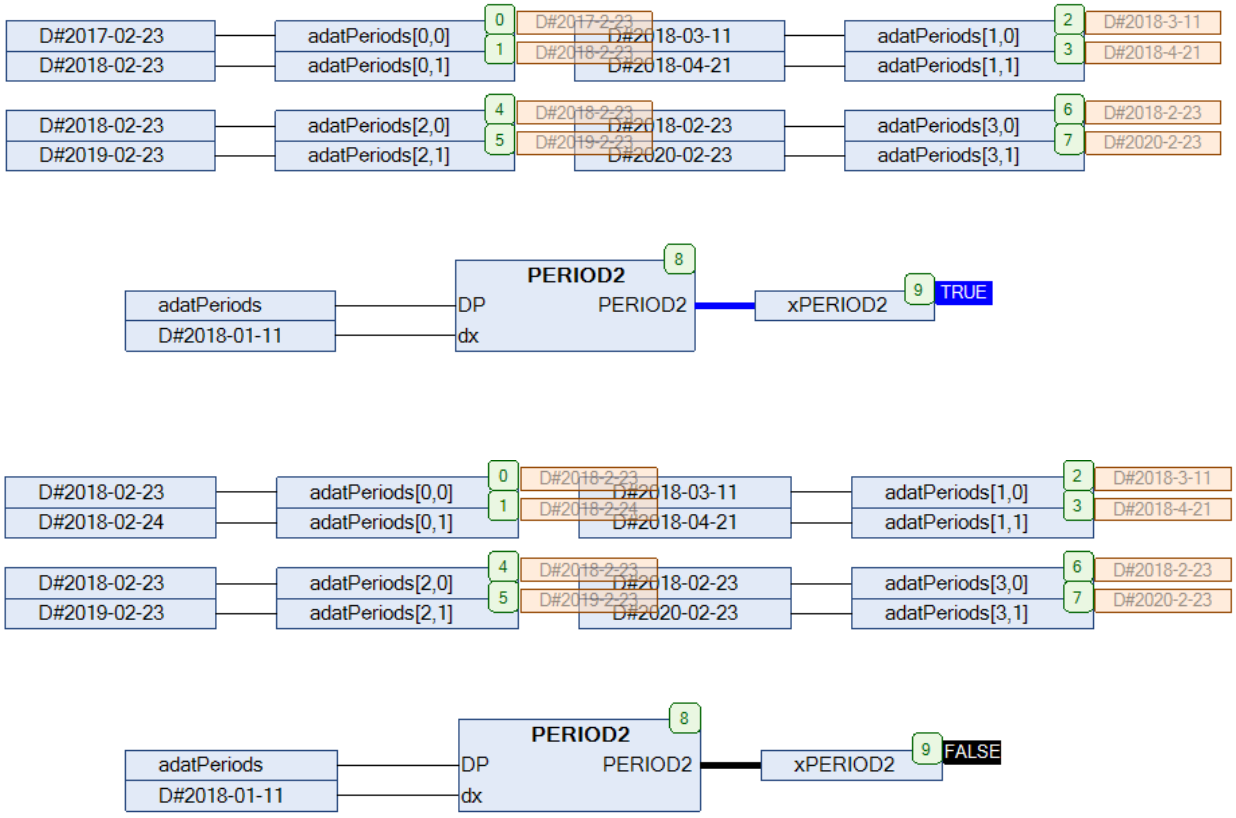


Рис. 12.69. Пример работы с функцией **PERIOD2** на языке CFC

12.37. REFRACTION

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	ELEV	REAL	Истинная высота светила в градусах.
Выходы	REFRACTION	REAL	Значение рефракции в градусах.

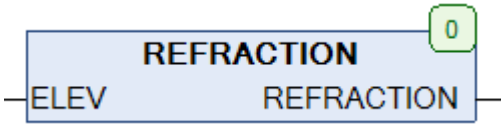
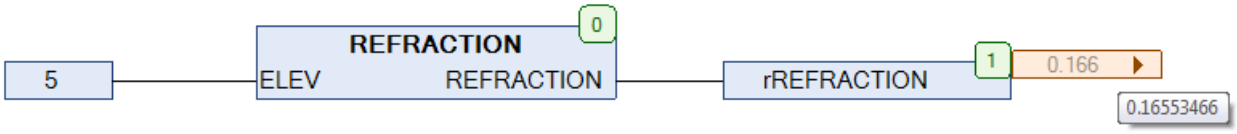


Рис. 12.70. Внешний вид функции **REFRACTION** на языке CFC

Функция **REFRACTION** возвращает значение [атмосферной рефракции](#) для светила с истинной высотой **ELEV** по [формуле Смардсона](#). Обе величины измеряются в градусах. Диапазон допустимых значений для функции: $-1.9 < \mathrm{ELEV} < 80$.



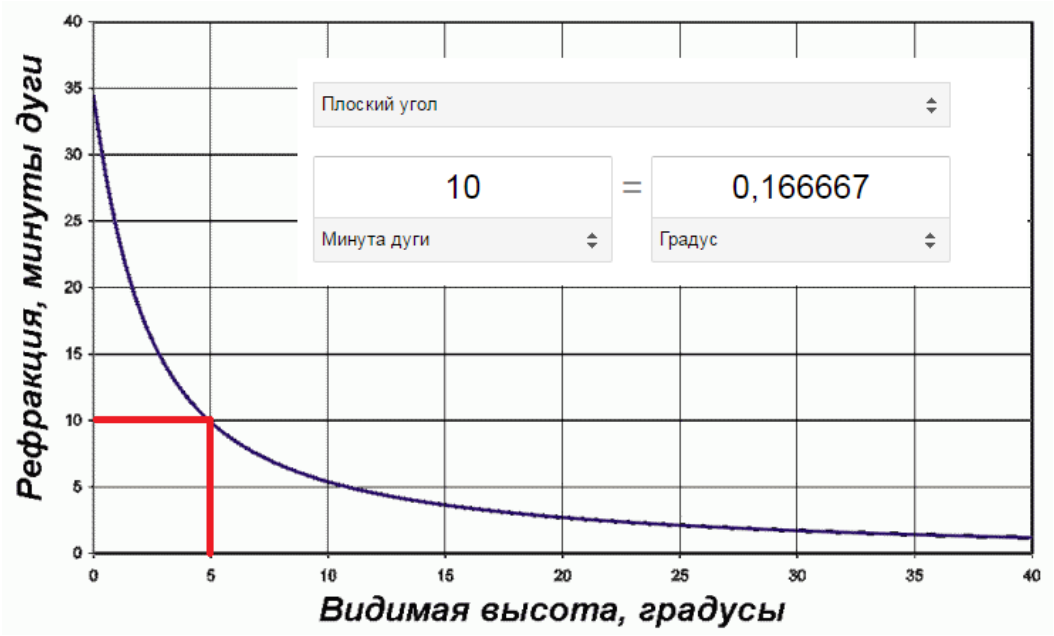


Рис. 12.71. Пример работы с функцией **REFRACTION** на языке CFC

12.38. RTC_2

Тип модуля: ФБ	Переменная	Тип	Описание
Входы	SET	BOOL	Сигнал установки опорной даты и времени.
	SDT	DT	Дата и время начала отсчета.
	SMS	INT	Значение миллисекунд.
	DEN	BOOL	Сигнал включения учета летнего времени.
	OFS	INT	Смещение местного времени относительно SDT в минутах.
Выходы	UDT	DT	Текущие дата и время в UTC.
	LDT	DT	Местные дата и время.
	DSO	BOOL	Флаг «учет летнего времени».
	XMS	INT	Текущее значение миллисекунд.
Используемые модули	RTC_MS		

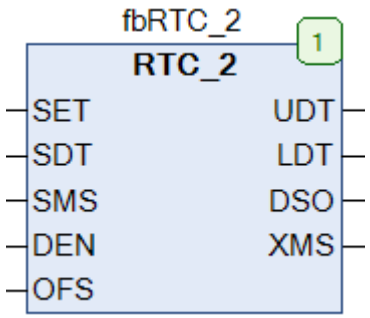


Рис. 12.72. Внешний вид ФБ **RTC_2** на языке CFC

Функциональный блок **RTC_2** представляет собой программные часы реального времени. При вызове блока начинается отсчет времени с момента, определяемого значениями входов **SDT** (дата и время в UTC) и **SMS** (миллисекунды). Текущее значение даты и времени в UTC подается на выход **UDT**, текущее значение миллисекунд – на выход **XMS**. По переднему фронту переменной **SET** в часы записываются новые опорные дата и время (**SDT** и **SMS**), относительно которых начинается отсчет. Если вход **DEN** имеет значение **TRUE**, то блок учитывает переход на летнее время, при этом выход **DSO** имеет значение **TRUE**. Вход **OFS** позволяет задать смещение местного времени относительно UTC в минутах. Текущие значения местного даты и времени подаются на выход **LDT**.

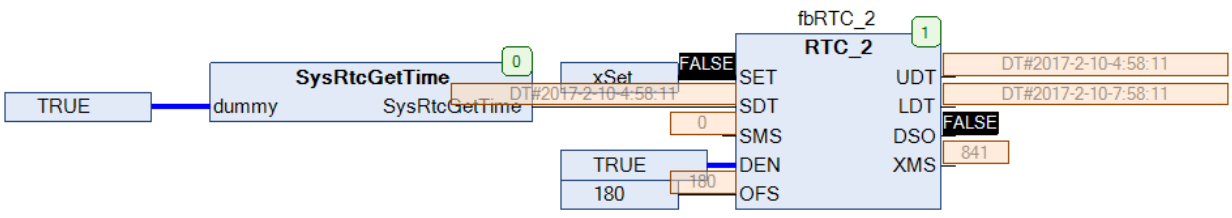


Рис. 12.73. Пример работы с ФБ **RTC_2** на языке CFC

12.39. RTC_MS

Тип модуля: ФБ	Переменная	Тип	Описание
Входы	SET	BOOL	Сигнал установки опорной даты и времени.
	SDT	DT	Дата и время начала отсчета.
	SMS	INT	Значение миллисекунд.
Выходы	XDT	DT	Текущие дата и время в UTC.
	XMS	INT	Текущее значение миллисекунд.
Используемые модули	T_PLC_MS		

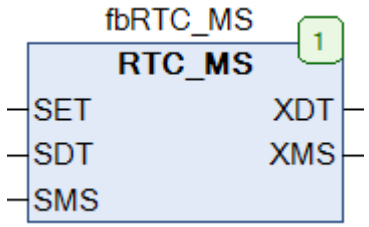


Рис. 12.74. Внешний вид ФБ **RTC_MS** на языке CFC

Функциональный блок **RTC_MS** представляет собой программные часы реального времени. При вызове блока начинается отсчет времени с момента, определяемого значениями входов **SDT** (дата и время в UTC) и **SMS** (миллисекунды). Текущее значение даты и времени в UTC подается на выход **UDT**, текущее значение миллисекунд – на выход **XMS**. По переднему фронту переменной **SET** в часы записываются новые опорные дата и время (**SDT** и **SMS**), относительно которых начинается отсчет.

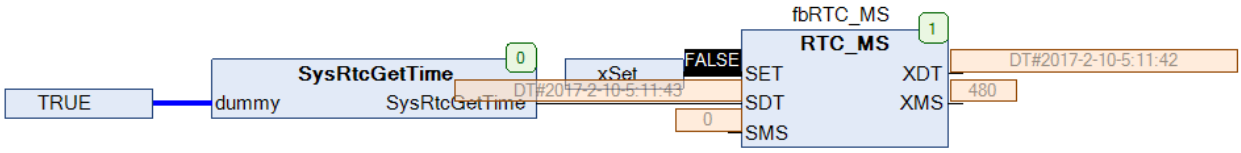


Рис. 12.75. Пример работы с ФБ **RTC_MS** на языке CFC

12.40. SDT_TO_DATE

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	DTI	SDT	Дата в формате SDT.
Выходы	SDT_TO_DATE	DATE	Дата в формате DATE.
Используемые модули	SET_DATE		

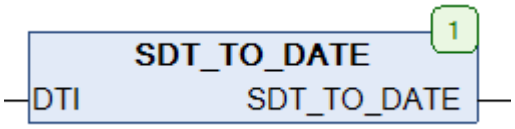


Рис. 12.76. Внешний вид функции **SDT_TO_DATE** на языке CFC

Функция **SDT_TO_DATE** конвертирует исходную дату **DTI** в формате SDT в переменную типа **DATE**.

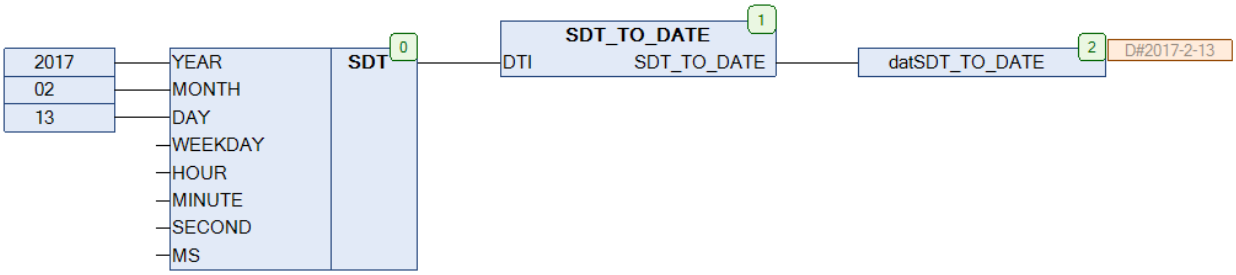


Рис. 12.77. Пример работы с функцией **SDT_TO_DATE** на языке CFC

12.41. SDT_TO_DT

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	DTI	SDT	Дата и время в формате SDT.
Выходы	SDT_TO_DT	DT	Дата и время в формате DT.
Используемые модули	SET_DT		

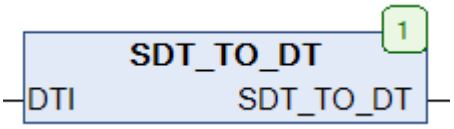


Рис. 12.78. Внешний вид функции **SDT_TO_DT** на языке CFC

Функция **SDT_TO_DT** конвертирует исходные дату и время **DTI** в формате SDT в переменную типа **DT**.

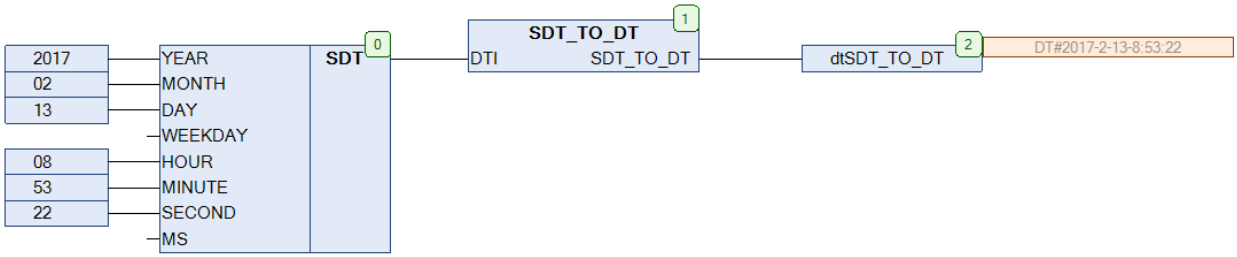


Рис. 12.79. Пример работы с функцией **SDT_TO_DT** на языке CFC

12.42. SDT_TO_TOD

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	DTI	SDT	Время в формате SDT.
Выходы	SDT_TO_TOD	TOD	Время в формате TOD.

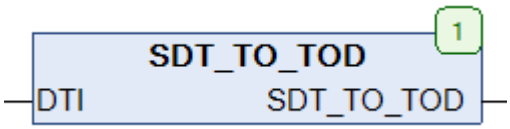


Рис. 12.80. Внешний вид функции **SDT_TO_TOD** на языке CFC

Функция **SDT_TO_TOD** конвертирует исходное время суток **DTI** в формате SDT в переменную типа **TOD**.

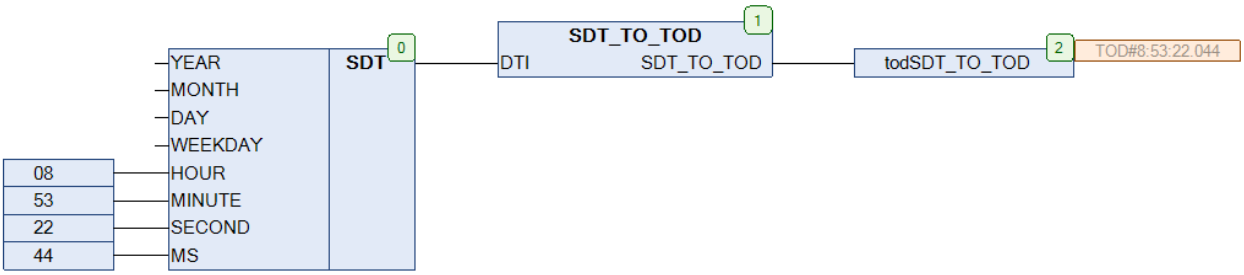


Рис. 12.81. Пример работы с функцией **SDT_TO_TOD** на языке CFC

12.43. SECOND

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	itod	TOD	Заданное время.
Выходы	SECOND	REAL	Число секунд и миллисекунд.

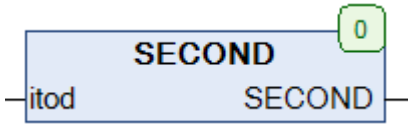


Рис. 12.82. Внешний вид функции **SECOND** на языке CFC

Функция **SECOND** возвращает число секунд и миллисекунд, вырезанное из исходного значения времени суток **itod** типа **TOD**.

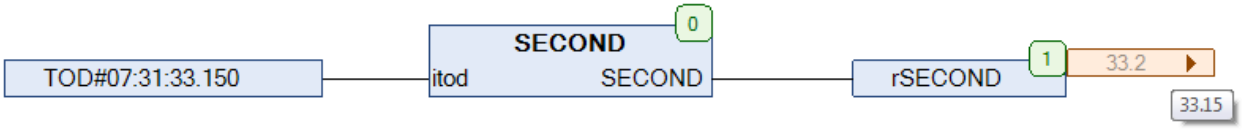


Рис. 12.83. Пример работы с функцией **SECOND** на языке CFC

12.44. SECOND_OF_DT

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	XDT	DT	Заданные дата и время.
Выходы	SECOND_OF_DT	INT	Число секунд.

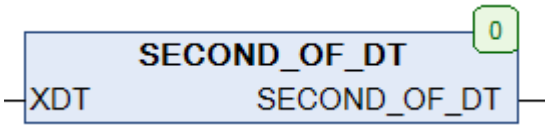


Рис. 12.84. Внешний вид функции **SECOND_OF_DT** на языке CFC

Функция **SECOND_OF_DT** возвращает число секунд, вырезанное из исходного значения даты и времени **XDT** типа **DT**.

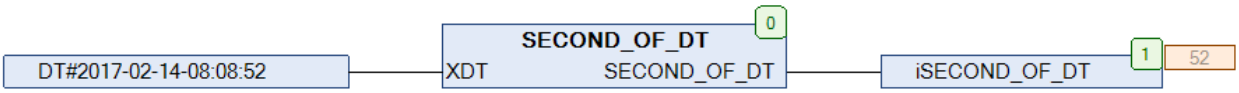


Рис. 12.85. Пример работы с функцией **SECOND_OF_DT** на языке CFC

12.45. SECOND_TO_TIME

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	IN	REAL	Количество секунд и миллисекунд.
Выходы	SECOND_TO_TIME	TIME	Время.

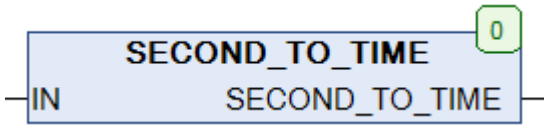


Рис. 12.86. Внешний вид функции **SECOND _TO_TIME** на языке CFC

Функция **SECOND_TO_TIME** конвертирует заданное количество секунд **IN** типа **REAL** (в виде числа с плавающей точкой) в значение переменной типа **TIME**.

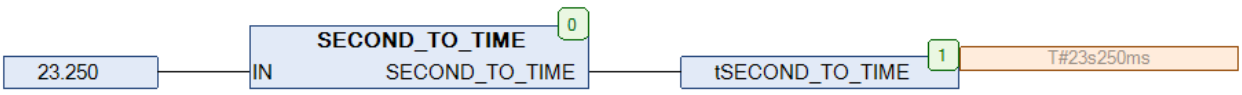


Рис. 12.87. Пример работы с функцией **SECOND _TO_TIME** на языке CFC

12.46. SET_DATE

Тип модуля: функция	Переменная	Тип	Описание
Входы	YEAR	INT	Год.
	MONTH	INT	Месяц.
	DAY	INT	День.
Выходы	SET_DATE	DATE	Дата в формате DATE.

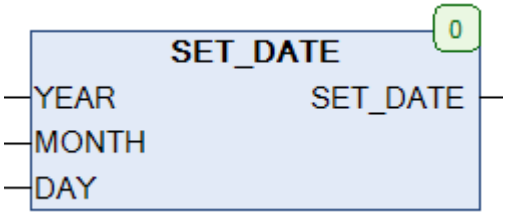


Рис. 12.88. Внешний вид функции **SET_DATE** на языке CFC

Функция **SET_DATE** возвращает дату типа **DATE,** собранную из значений года **YEAR**, месяца **MONTH** и дня **DAY**. Функция не проверяет корректность входных данных, так что пользователь должен ограничить диапазон их возможных значений (например, 1-12 для **MONTH** и т.д.).

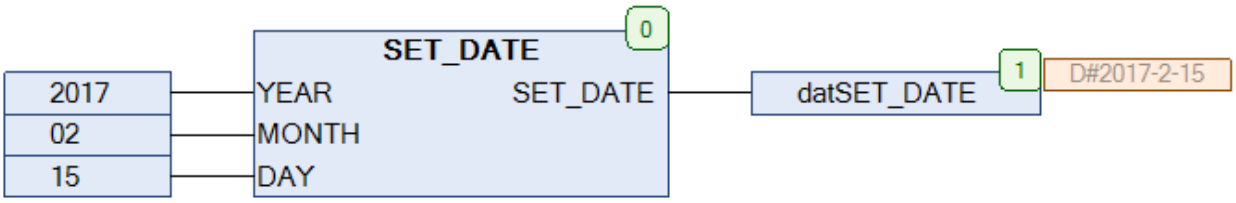


Рис. 12.89. Пример работы с функцией **SET_DATE** на языке CFC

12.47. SET_DT

Тип модуля: функция	Переменная	Тип	Описание
Входы	YEAR	INT	Год.
	MONTH	INT	Месяц.
	DAY	INT	День.
	hour	INT	Часы.
	minute	INT	Минуты.
	second	INT	Секунды.
Выходы	SET_DT	DT	Дата и время в формате DT.
Используемые модули	SET_DATE		

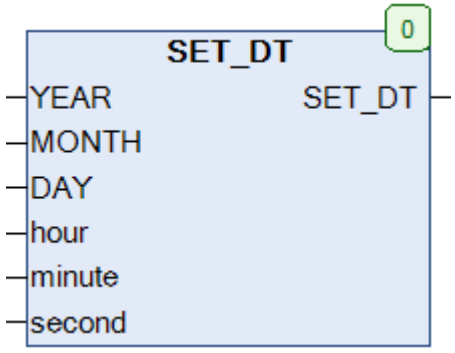


Рис. 12.90. Внешний вид функции **SET_DT** на языке CFC

Функция **SET_DT** возвращает дату и время типа **DT**, собранную из значений года **YEAR**, месяца **MONTH**, дня **DAY**, часов **hour**, минут **minute** и секунд **second**. Функция не проверяет корректность входных данных, так что пользователь должен ограничить диапазон их возможных значений (например, 1-12 для **MONTH** и т.д.).

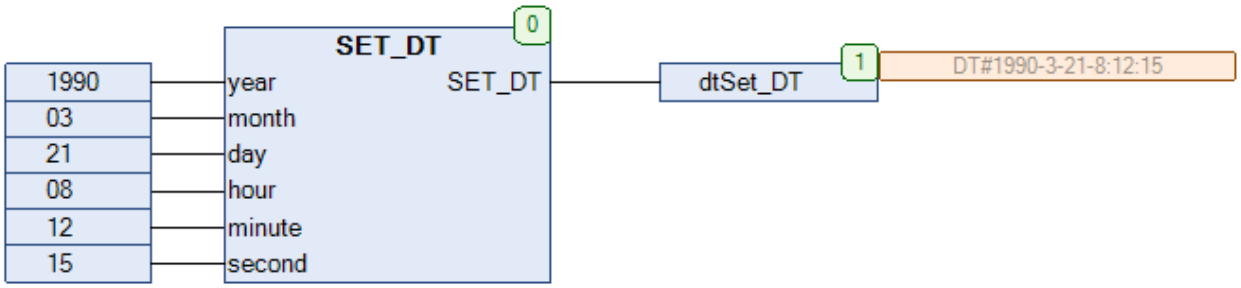


Рис. 12.91. Пример работы с функцией **SET_DT** на языке CFC

12.48. SET_TOD

Тип модуля: функция	Переменная	Тип	Описание
Входы	hour	INT	Часы.
	minute	INT	Минуты.
	second	INT	Секунды.
Выходы	SET_TOD	DT	Дата и время в формате TOD.

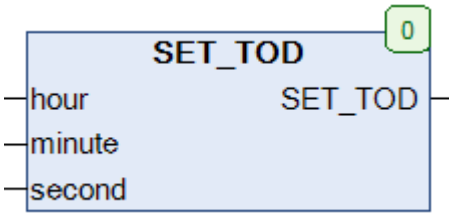


Рис. 12.92. Внешний вид функции **SET_TOD** на языке CFC

Функция **SET_TOD** возвращает время суток типа **TOD**, собранное из значений часов **hour**, минут **minute** и секунд **second**. Функция не проверяет корректность входных данных, так что пользователь должен ограничить диапазон их возможных значений (например, 1-24 для **hour** и т.д.).

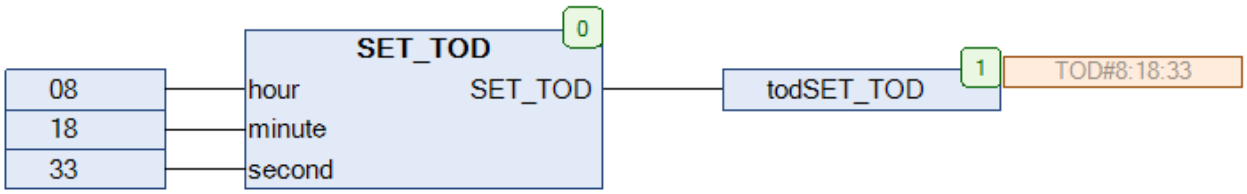


Рис. 12.93. Пример работы с функцией **SET_TOD** на языке CFC

12.49. SUN_MIDDAY

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	LON	REAL	Долгота (в градусах).
	UTC	DATE	Дата в UTC.
Выходы	SUN_MIDDAY	TOD	Время истинного полудня.
Используемые модули	DAY_OF_YEAR		

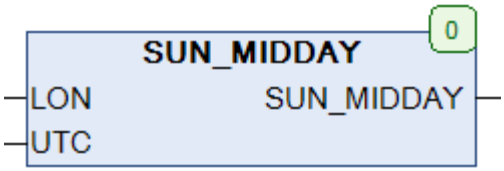


Рис. 12.94. Внешний вид функции **SUN_MIDDAY** на языке CFC

Функция **SUN_MIDDAY** вычисляет время истинного полудня (в UTC) для даты **UTC** на долготе **LON**. Подробная информация о формуле, по которой производится вычисление, приведена здесь.

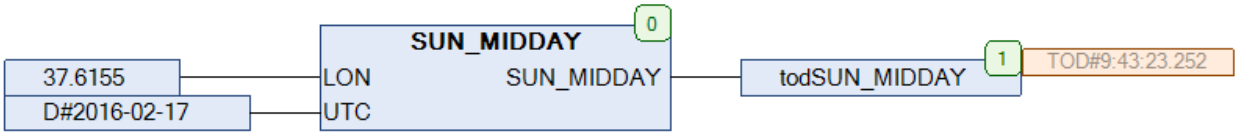


Рис. 12.95. Пример работы с функцией **SUN_MIDDAY** на языке CFC

12.50. SUN_POS

Тип модуля: ФБ	Переменная	Тип	Описание
Входы	latitude	REAL	Широта (в градусах).
	longitude	REAL	Долгота (в градусах).
	utc	DT	Дата и время в UTC.
Выходы	B	REAL	Азимут (в градусах).
	H	REAL	Высота солнца над горизонтом (в градусах)
	HR	REAL	Высота солнца над горизонтом (в градусах) с учетом рефракции.
Используемые модули	MODR, RAD, DEG, REFRACTION		

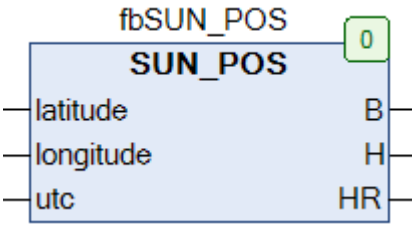


Рис. 12.96. Внешний вид ФБ **SUN_POS** на языке CFC

Функциональный блок **SUN_POS** вычисляет азимут **B**, высоту солнца над горизонтом **H** и высоту солнца над горизонтом с учетом рефракции **HR** для заданной широты **latitude** и долготы **longitude** в момент **utc** (дата и время по UTC). Расчет рефракции производится для нормального атмосферного давления (101325 Па) и температуры 10 °C. Погрешность расчета в диапазоне дат 2000…2050 не превышает 0.1 градуса.

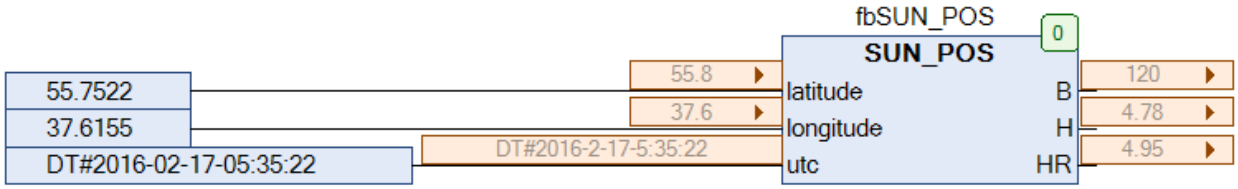


Рис. 12.97. Пример работы с ФБ **SUN_POS** на языке CFC

12.51. SUN_TIME

Тип модуля: ФБ	Переменная	Тип	Описание
Входы	latitude	REAL	Широта (в градусах).
	longitude	REAL	Долгота (в градусах).
	utc	DATE	Дата в UTC.
	H	REAL	Высота солнца над горизонтом (в градусах) на восходе.
Выходы	MIDDAY	TOD	Время солнечного полудня (в UTC).
	SUN_RISE	TOD	Время восхода солнца (в UTC).
	SUN_SET	TOD	Время заказа солнца (в UTC).
	SUN_DECLINATION	REAL	Склонение солнца над горизонтом в полдень (в градусах).
Используемые модули	SUN_MIDDAY, DAY_OF_YEAR, DEG, HOUR_TO_TIME, RAD		

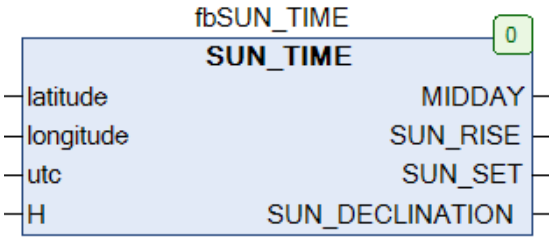


Рис. 12.98. Внешний вид ФБ **SUN_TIME** на языке CFC

Функциональный блок **SUN_TIME** вычисляет для заданной широты **latitude**, долготы **longitude** и даты **utc** (по UTC) время восхода солнца **SUN_RISE**, время заката солнца **SUN_SET**, время солнечного полудня **MIDDAY**, а также склонение солнца над горизонтом в полдень **SUN_DECLINATION**. Функциональный блок использует алгоритм, основанный на уравнении восхода, что минимизирует нагрузку на ПЛК. Не имеет смысла вызывать данный ФБ более одного раза в день – для расчета позиции солнца в течение дня должен использоваться ФБ **SUN_POS**. ФБ возвращает корректные результаты в пределах 65° Южной широты … 65° Северной широты (-65 < latitude < 65).

Вход **H** используется для задания высоты солнечного диска. По умолчанию он имеет значение -0.83(3), которое используется при наблюдениях на земле. При наблюдениях за светилом на морском горизонте или возвышении это значение должно быть скорректировано (см. ссылку на информацию об уравнении восхода).

Более подробная информация приведена здесь и здесь.

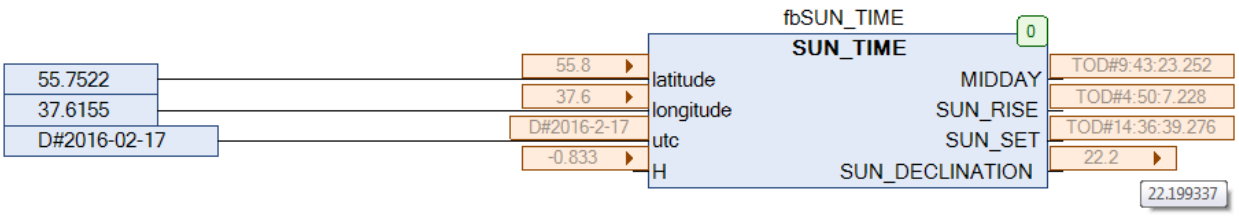


Рис. 12.99. Пример работы с ФБ **SUN_TIME** на языке CFC

12.52. TIMECHECK

Тип модуля: функция	Переменная	Тип	Описание
Входы	TD	TOD	Заданное время суток.
	START	TOD	Нижний предел контролируемого диапазона.
	STOP	TOD	Верхний предел контролируемого диапазона.
Выходы	TIMECHECK	BOOL	Флаг «TD принадлежит интервалу».

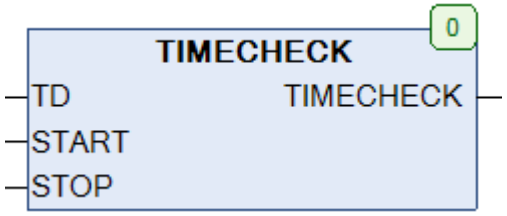


Рис. 12.100. Внешний вид функции **TIMECHECK** на языке CFC

Функция **TIMECHECK** возвращает **TRUE**, если заданное время суток **TD** принадлежит интервалу $\mathrm{START} < \mathrm{TD} < \mathrm{STOP}$. При этом значение **START** может превышать значение **STOP** (в том случае, если в контролируемый диапазон входит часть следующих суток).

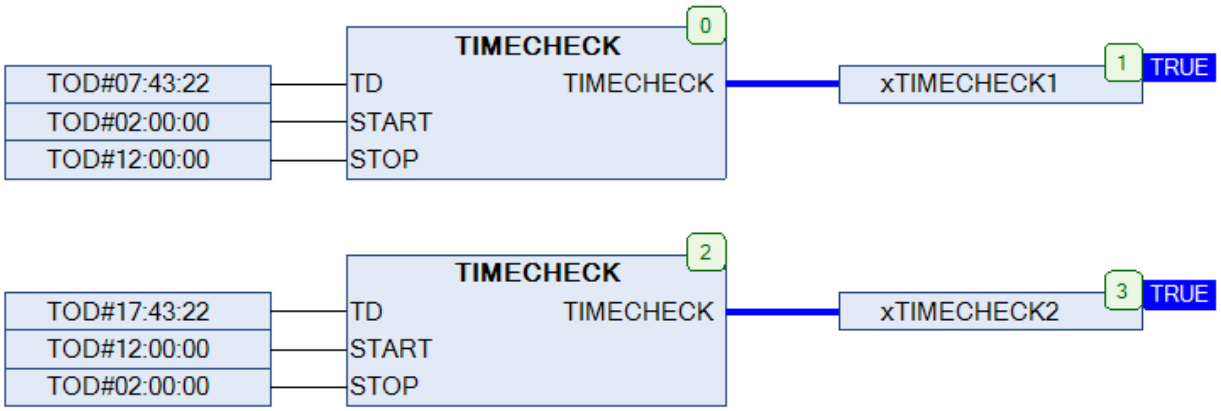


Рис. 12.101. Пример работы с функцией **TIMECHECK** на языке CFC

12.53. UTC_TO_LTIME

Тип модуля: функция	Переменная	Тип	Описание
Входы	UTC	DT	Дата и время в UTC.
	DST_ENABLE	BOOL	Сигнал включения учета летнего времени.
	TIME_ZONE_OFFSET	INT	Смещение местного времени относительно UTC в минутах.
Выходы	UTC_TO_LTIME	DT	Местные дата и время.
Используемые модули	DST		

Обратите внимание, что в текущей версии библиотеки функция работает **некорректно** (пруф) для UTC+9 и выше. Для корректной работы необходимо отредактировать код функции следующим образом:

```
UTC_TO_LTIME
1  FUNCTION UTC_TO_LTIME : DT
2  VAR_INPUT
3      UTC : DT;
4      DST_ENABLE : BOOL;
5      TIME_ZONE_OFFSET : INT;
6  END_VAR
7  VAR
8      tmp:    DINT;   // bug fix
9  END_VAR
10
11 (*
```

```
1  tmp := TIME_ZONE_OFFSET * 60 + BOOL_TO_INT(DST_ENABLE AND DST(UTC)) * 3600;
2  IF tmp < 0 THEN
3      tmp := ABS(tmp);
4      UTC_TO_LTIME := DWORD_TO_DT(DT_TO_DWORD(UTC) -  DINT_TO_DWORD(tmp));    // bug fix
5  ELSE
6      UTC_TO_LTIME := DWORD_TO_DT(DT_TO_DWORD(UTC) +  DINT_TO_DWORD(tmp));    // bug fix
7  END_IF;
```

Рис. 12.102. Исправление исходного кода функции **UTC_TO_LTIME** для обеспечения корректной работы

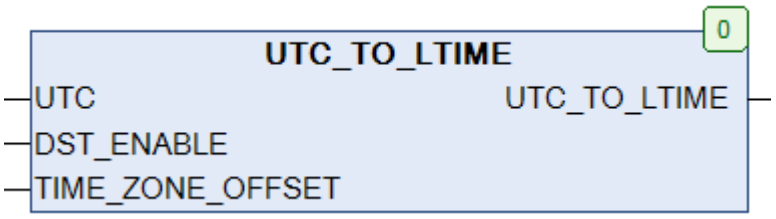


Рис. 12.103. Внешний вид функции **UTC_TO_LTIME** на языке CFC

Функция **UTC_TO_LTIME** конвертирует дату и время из UTC в местное время. Вход **TIME_ZONE_OFFSET** используется для задания значения смещения местного времени относительно **UTC** в минутах. Если вход **DST_ENABLE** имеет значение **TRUE**, то функция учитывает переход на летнее время.

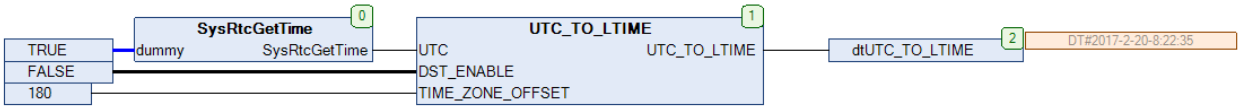


Рис. 12.104. Пример работы с функцией **UTC_TO_LTIME** на языке CFC

12.54. WORK_WEEK

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	idate	DATE	Заданная дата.
Выходы	WORK_WEEK	INT	Номер недели в году.
Используемые модули	YEAR_OF_DATE, YEAR_BEGIN, DAY_OF_WEEK, LEAP_YEAR		

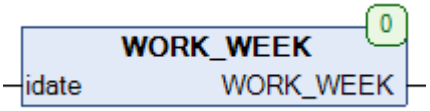


Рис. 12.105. Внешний вид функции **WORK_WEEK** на языке CFC

Функция **WORK_WEEK** возвращает номер недели в году для заданной даты **idate**. Год начинается с 1-ой недели.

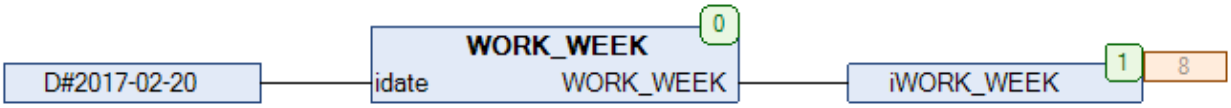


Рис. 12.106. Пример работы с функцией **WORK_WEEK** на языке CFC

12.55. YEAR_BEGIN

Тип модуля: функция	Переменная	Тип	Описание
Входы	y	INT	Заданный год.
Выходы	YEAR_BEGIN	DATE	Дата первого дня года.

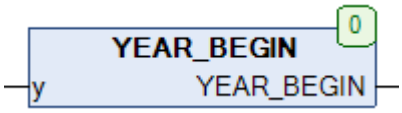


Рис. 12.107. Внешний вид функции **YEAR_BEGIN** на языке CFC

Функция **YEAR_BEGIN** возвращает дату первого дня для заданного года **y**.

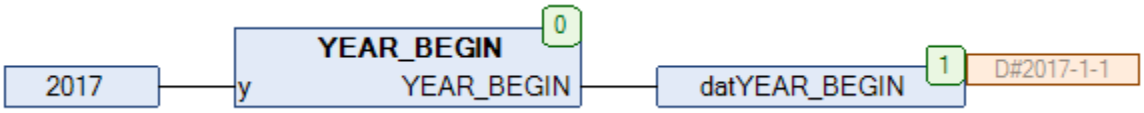


Рис. 12.108. Пример работы с функцией **YEAR_BEGIN** на языке CFC

12.56. YEAR_END

Тип модуля: функция	Переменная	Тип	Описание
Входы	y	INT	Заданный год.
Выходы	YEAR_END	DATE	Дата последнего дня года.

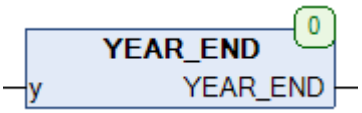


Рис. 12.109. Внешний вид функции **YEAR_END** на языке CFC

Функция **YEAR_END** возвращает дату последнего дня для заданного года **y**.

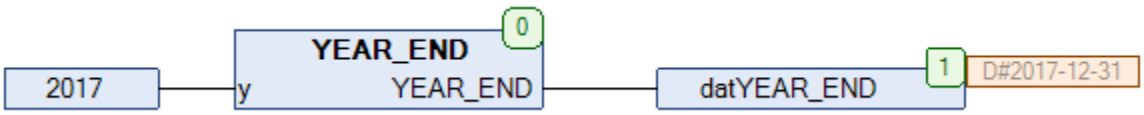


Рис. 12.110. Пример работы с функцией **YEAR_END** на языке CFC

12.57. YEAR_OF_DATE

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	IDATE	DATE	Исходная дата.
Выходы	YEAR_OF_DATE	INT	Год.

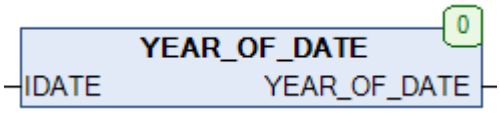


Рис. 12.111. Внешний вид функции **YEAR_OF_DATE** на языке CFC

Функция **YEAR_OF_DATE** возвращает значение года, вырезанное из исходной даты **IDATE** типа **DATE**.

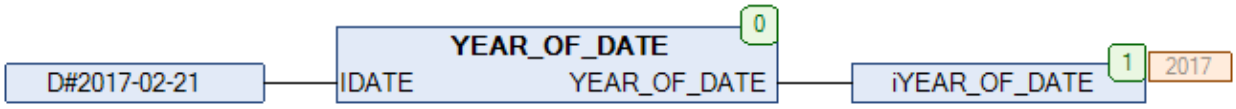


Рис. 12.112. Пример работы с функцией **YEAR_OF_DATE** на языке CFC

13. Работа со строками

13.1. BIN_TO_BYTE

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	BIN	STRING(12)	Строка бит (STRING).
Выходы	BIN_TO_BYTE	BYTE	Строка бит (BYTE).

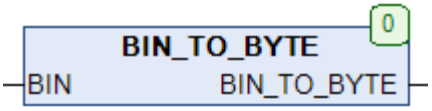


Рис. 13.1. Внешний вид функции **BIN_TO_BYTE** на языке CFC

Функция **BIN_TO_BYTE** конвертирует входную строку **BIN**, состоящую из символов **0** и **1**, в значение типа **BYTE**.

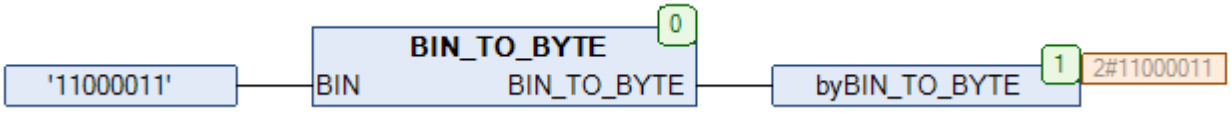


Рис. 13.2. Пример работы с функцией **BIN_TO_BYTE** на языке CFC

13.2. BIN_TO_DWORD

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	BIN	STRING(40)	Строка бит (STRING).
Выходы	BIN_TO_DWORD	DWORD	Строка бит (BYTE).

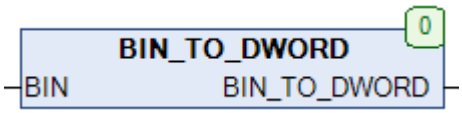


Рис. 13.3. Внешний вид функции **BIN_TO_DWORD** на языке CFC

Функция **BIN_TO_DWORD** конвертирует входную строку **BIN**, состоящую из символов **0** и **1**, в значение типа **DWORD**.

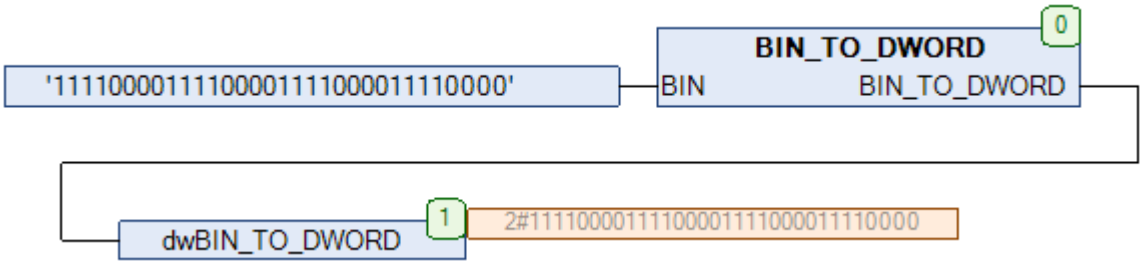


Рис. 13.4. Пример работы с функцией **BIN_TO_DWORD** на языке CFC

13.3. BYTE_TO_STRB

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	IN	BYTE	Строка бит (BYTE).
Выходы	BYTE_TO_STRB	STRING(8)	Строка бит (STRING).

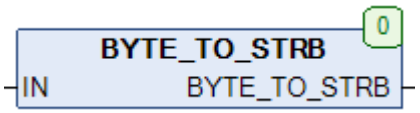


Рис. 13.5. Внешний вид функции **BYTE_TO_STRB** на языке CFC

Функция **BYTE_TO_STRB** конвертирует значение **IN** типа **BYTE** в строку, состоящую из символов **0** и **1**.

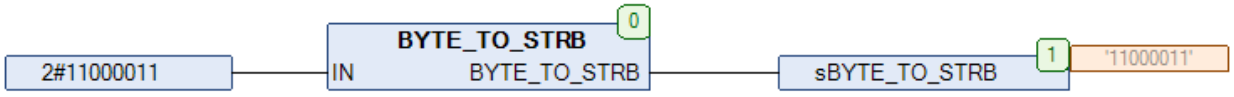


Рис. 13.6. Пример работы с функцией **BYTE_TO_STRB** на языке CFC

13.4. BYTE_TO_STRH

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	IN	BYTE	Строка бит (BYTE).
Выходы	BYTE_TO_STRH	STRING(2)	Строка бит в HEX (STRING).

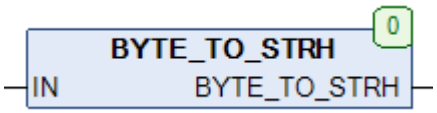


Рис. 13.7. Внешний вид функции **BYTE_TO_STRH** на языке CFC

Функция **BYTE_TO_STRH** конвертирует значение **IN** типа **BYTE** в строку, содержащую это значение в 16-ричной системе счисления.

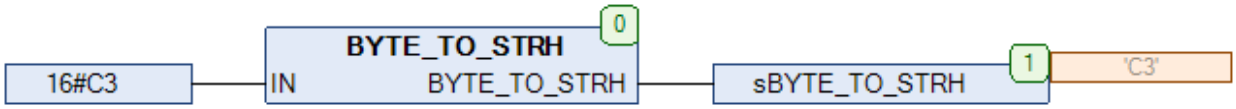


Рис. 13.8. Пример работы с функцией **BYTE_TO_STRH** на языке CFC

13.5. CAPITALIZE

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	str	STRING	Исходная строка.
Выходы	CAPITALIZE	STRING	Обработанная строка.
Используемые модули	TO_UPPER		

Рис. 13.9. Внешний вид функции **CAPITALIZE** на языке CFC

Функция **CAPITALIZE** переводит первые символы слов из строки **str** в верхний регистр. Слова должны быть разделены пробелами. Для работы с символами верхней половины таблицы ASCII глобальная переменная EXTENDED_ASCII должна иметь значение **TRUE**.

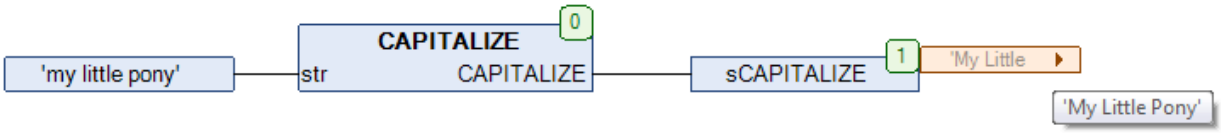


Рис. 13.10. Пример работы с функцией **CAPITALIZE** на языке CFC

13.6. CHARCODE

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	STR	STRING(10)	Символ/идентификатор.
Выходы	CHARCODE	BYTE	Код символа/идентификатора.
Используемые модули	CODE		

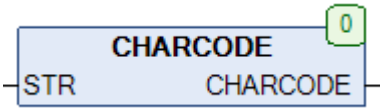


Рис. 13.11. Внешний вид функции **CHARCODE** на языке CFC

Функция **CHARCODE** возвращает код для заданного ASCII-символа или идентификатора **STR**. Список идентификаторов приведен в таблице ниже. В случае получения неизвестного символа или идентификатора функция возвращает **0**. Работа функции основана на использовании глобальной переменной CHARNAMES.

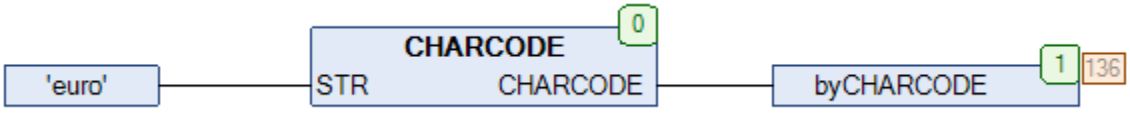


Рис. 13.12. Пример работы с функцией **CHARCODE** на языке CFC

Обозначение	Идентификатор	Код		Обозначение	Идентификатор	Код
"	quot	34		½	frac12	189
&	amp	38		¾	frac34	190
>	lt	60		¿	iquest	191
<	gt	62		À	Agrave	192
€	euro	136		Á	Aacute	193
	nbsp	160		Â	Acirc	194
¡	iexcl	161		Ã	Atilde	195
¢	cent	162		Ä	Auml	196
£	pound	163		Å	Aring	197
¤	curren	164		Æ	AElig	198
¥	yen	165		Ç	Ccedil	199
¦	brvbar	166		È	Egrave	200
§	sect	167		É	Eacute	201
¨	uml	168		Ê	Ecirc	202
©	copy	169		Ë	Euml	203
ª	ordf	170		Ì	Igrave	204
«	laquo	171		Í	Iacute	205
¬	not	172		Î	Icirc	206
-	shy	173		Ï	Iuml	207
®	reg	174		Ð	ETH	208
¯	macr	175		Ñ	Ntilde	209
°	deg	176		Ò	Ograve	210
±	plusmn	177		Ó	Oacute	211
²	sup2	178		Ô	Ocirc	212
³	sup3	179		Õ	Otilde	213
´	acute	180		Ö	Ouml	214
µ	micro	181		×	times	215
¶	para	182		Ø	Oslash	216
·	middot	183		Ù	Ugrave	217
¸	cedil	184		Ú	Uacute	218
¹	sup1	185		Û	Ucirc	219
º	ordm	186		Ü	Uuml	220
»	raquo	187		Ý	Yacute	221
ß	szlig	223		ð	eth	240
à	agrave	224		ñ	ntilde	241
á	aacute	225		ò	ograve	242
â	acirc	226		ó	oacute	243
ã	atilde	227		ô	ocirc	244
ä	auml	228		õ	otilde	245
å	aring	229		ö	ouml	246
æ	aelig	230		÷	divide	247
ç	ccedil	231		ø	oslash	248
è	egrave	232		ù	ugrave	249
é	eacute	233		ú	uacute	250
ê	ecirc	234		û	ucirc	251
ë	euml	235		ü	uuml	252
ì	igrave	236		ý	yacute	253
í	iacute	237		þ	thorn	254
î	icirc	238		ÿ	yuml	255
ï	iuml	239				

13.7. CHARNAME

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	C	BYTE	Код идентификатора.
Выходы	CHARNAME	STRING(10)	Идентификатор.
Используемые модули	CHR_TO_STRING		

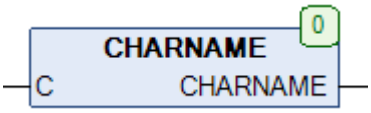


Рис. 13.13. Внешний вид функции **CHARNAME** на языке CFC

Функция **CHARNAME** возвращает идентификатор по заданному коду **C**. Список идентификаторов приведен в таблице в п. 13.6. Работа функции основана на использовании глобальной переменной CHARNAMES.

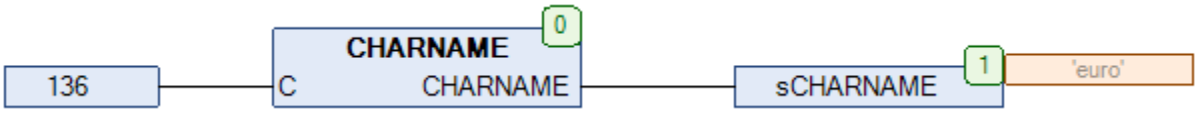


Рис. 13.14. Пример работы с функцией **CHARNAME** на языке CFC

13.8. CHR_TO_STRING

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	C	BYTE	Код ASCII-символа.
Выходы	CHR_TO_STRING	STRING(1)	ASCII-символ.

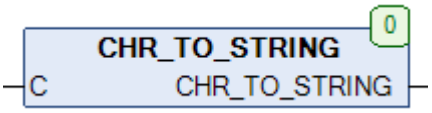


Рис. 13.15. Внешний вид функции **CHR_TO_STRING** на языке CFC

Функция **CHR_TO_STRING** возвращает символ заданного ASCII-кода **C**.

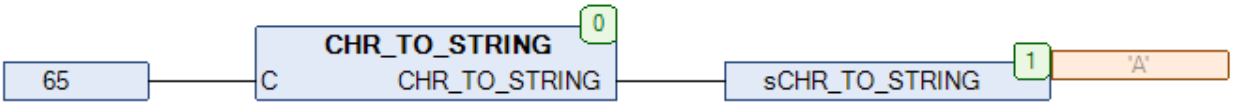


Рис. 13.16. Пример работы с функцией **CHR_TO_STRING** на языке CFC

13.9. CLEAN

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	IN	STRING	Исходная строка.
	CX	STRING(80)	Допустимые символы.
Выходы	CLEAN	STRING	Обработанная строка.

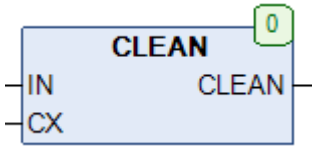


Рис. 13.17. Внешний вид функции **CLEAN** на языке CFC

Функция **CLEAN** вырезает из строки **IN** все символы, не входящие в строку **CX**.

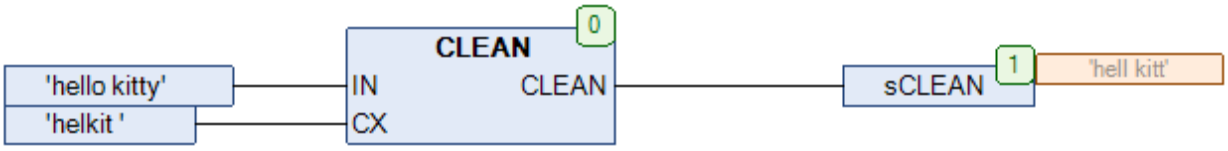


Рис. 13.18. Пример работы с функцией **CLEAN** на языке CFC

13.10. CODE

Тип модуля: функция	Переменная	Тип	Описание
Входы	STR	STRING	Исходная строка.
	POS	INT	Позиция символа.
Выходы	CODE	BYTE	ASCII-код символа.

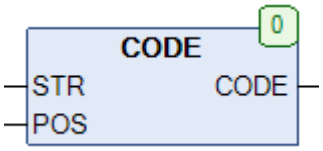


Рис. 13.19. Внешний вид функции **CODE** на языке CFC

Функция **CODE** возвращает код ASCII-символа из строки **STR** с позиции **POS**. Если POS = 0 или POS > длина строки, то функция возвращает **0**.

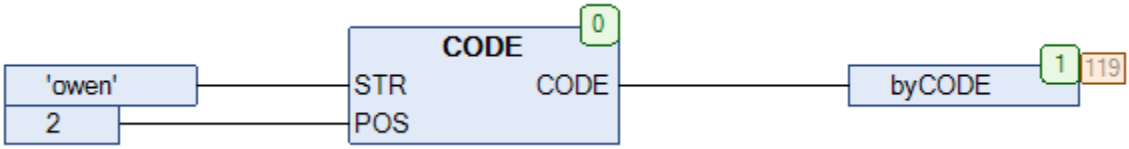


Рис. 13.20. Пример работы с функцией **CODE** на языке CFC

13.11. COUNT_CHAR

Тип модуля: функция	Переменная	Тип	Описание
Входы	STR	STRING	Исходная строка.
	chr	BYTE	ASCII-код символа.
Выходы	COUNT_CHAR	INT	Кол-во символов в строке.

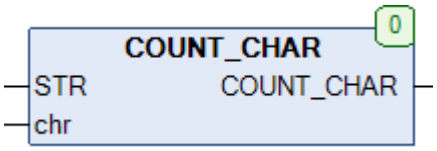


Рис. 13.21. Внешний вид функции **COUNT_CHAR** на языке CFC

Функция **COUNT_CHAR** возвращает число символов с ASCII-кодом **chr** в строке **STR**.

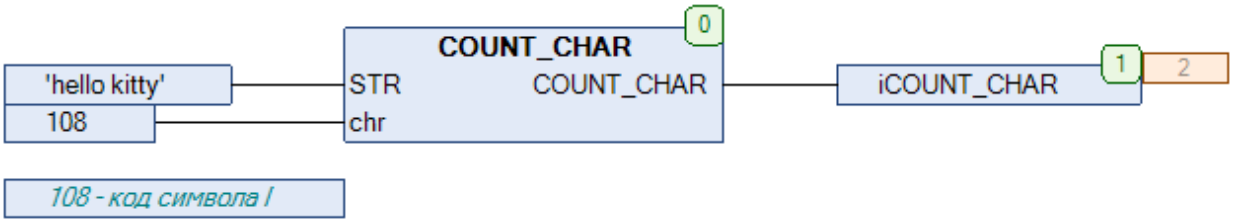


Рис. 13.22. Пример работы с функцией **COUNT_CHAR** на языке CFC

13.11a. COUNT_SUBSTRING

Тип модуля: функция	Переменная	Тип	Описание
Входы	SEARCH	STRING	Искомый фрагмент.
	STR	STRING	Исходная строка.
Выходы	COUNT_SUBSTRING	INT	Кол-во вхождений фрагмента в строку.

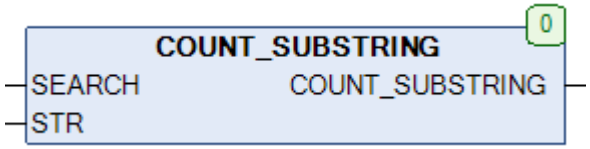


Рис. 13.21a. Внешний вид функции **COUNT_SUBSTRING** на языке CFC

Функция **COUNT_SUBSTRING** возвращает число вхождений строки **SEARCH** в строку **STR**.

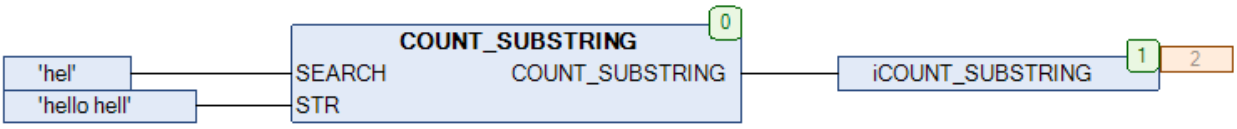


Рис. 13.22a. Пример работы с функцией **COUNT_SUBSTRING** на языке CFC

13.12. DEC_TO_BYTE

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	DEC	STRING(10)	Значение в символьном виде.
Выходы	DEC_TO_BYTE	BYTE	Значение в целочисленном виде.

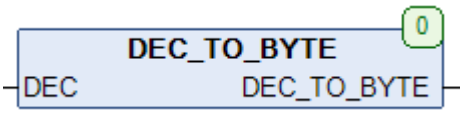


Рис. 13.23. Внешний вид функции DEC_TO_BYTE на языке CFC

Функция **DEC_TO_BYTE** конвертирует символьное значение **DEC** в целочисленное значение типа **BYTE**. Учитываются только символы ‘0’ – ‘9’ в пределах допустимого для типа **BYTE** диапазона ‘0’ – ‘255’.

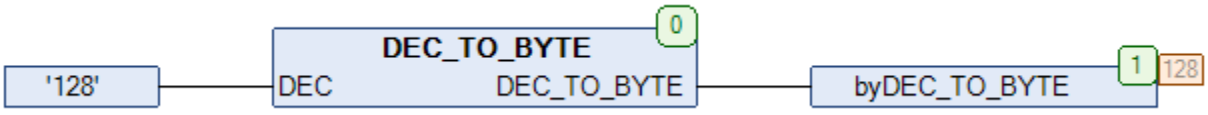


Рис. 13.24. Пример работы с функцией **DEC_TO_BYTE** на языке CFC

13.13. DEC_TO_DWORD

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	DEC	STRING(20)	Значение в символьном виде.
Выходы	DEC_TO_DWORD	DWORD	Значение в целочисленном виде.

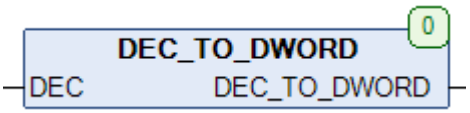


Рис. 13.25. Внешний вид функции **DEC_TO_DWORD** на языке CFC

Функция **DEC_TO_DWORD** конвертирует символьное значение **DEC** в целочисленное значение типа **DWORD**. Учитываются только символы ‘0’ – ‘9’ в пределах допустимого для типа DWORD диапазона ‘0’ – ‘4294967295’.

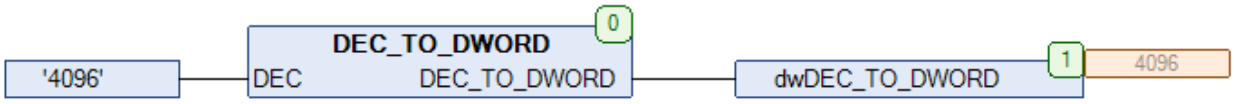


Рис. 13.26. Пример работы с функцией **DEC_TO_DWORD** на языке CFC

13.14. DEC_TO_INT

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	DEC	STRING(10)	Значение в символьном виде.
Выходы	DEC_TO_INT	INT	Значение в целочисленном виде.

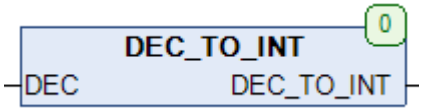


Рис. 13.27. Внешний вид функции **DEC_TO_INT** на языке CFC

Функция **DEC_TO_INT** конвертирует символьное значение **DEC** в целочисленное значение типа **INT**. Учитываются только символы ‘0’ – ‘9’ и '-' в пределах допустимого для типа INT диапазона ‘-32768’ – ‘32767’.

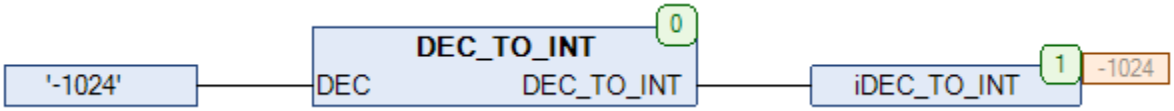


Рис. 13.28. Пример работы с функцией **DEC_TO_INT** на языке CFC

13.15. DEL_CHARS

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	IN	STRING	Исходная строка.
	CX	STRING(80)	Запрещенные символы.
Выходы	DEL_CHARS	STRING	Обработанная строка.

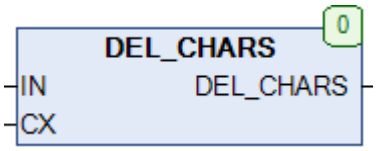


Рис. 13.29. Внешний вид функции **DEL_CHARS** на языке CFC

Функция **DEL_CHARS** вырезает из строки **IN** все символы, входящие в строку **CX**.

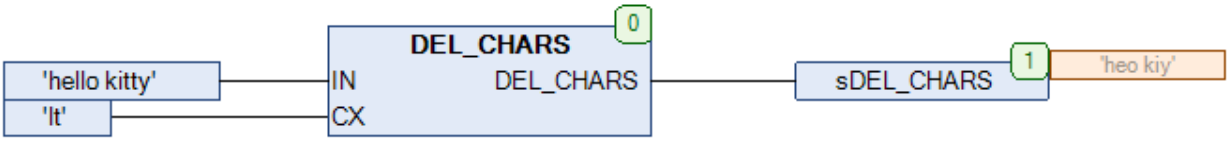


Рис. 13.30. Пример работы с функцией **DEL_CHARS** на языке CFC

13.16. DT_TO_STRF

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	DTI	DT	Дата и время.
	MS	INT	Число миллисекунд.
	FMT	STRING	Формат строкового представления даты и времени.
	LANG	INT	Язык приложения.
Выходы	DT_TO_STRF	STRING	Строка с форматированной меткой времени.
Используемые модули	YEAR_OF_DATE, MONTH_OF_DATE, MONTH_TO_STRING, DAY_OF_MONTH, DAY_OF_WEEK, WEEKDAY_TO_STRING, HOUR, MINUTE, SECOND		

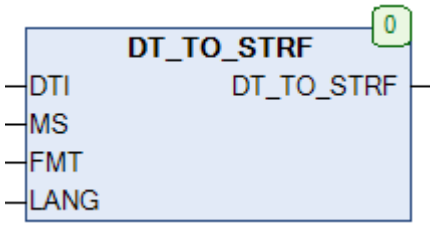


Рис. 13.31. Внешний вид функции **DT_TO_STRF** на языке CFC

Функция **DT_TO_STRF** конвертирует значение даты и времени **DTI** типа **DT** в форматированную строку. Вход **MS** позволяет указать число миллисекунд. Вход **LANG** определяет язык приложения (см. глобальную переменную LANGUAGE). Вход **FMT** представляет собой строку, состоящую из текста и заполнителей, которая определяет форматирование даты и времени. Список заполнителей приведен ниже.

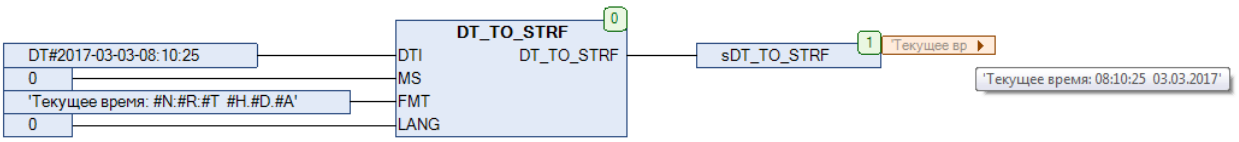


Рис. 13.32. Пример работы с функцией **DT_TO_STRF** на языке CFC

Заполнитель	Описание
#A	Полное значение года (2008).
#B	Сокращенное значение года (08).
#C	Значение месяца без ведущего нуля (5).
#D	Значение месяца с ведущим нулем (05).
#E	Сокращенное название месяца (Jan).
#F	Полное название месяца (January).
#G	Значение дня без ведущего нуля (5).
#H	Значение дня с ведущим нулем (05).
#I	Номер дня недели (1 – понедельник, 7 – воскресение).
#J	Сокращенное название дня (Mo).
#K	Полное название дня (Monday).
#L	Утро/вечер (AM/PM).
#M	Значение часа в 24-часовом формате без ведущего нуля (2, 22).
#N	Значение часа в 24-часовом формате с ведущим нулем (02, 22).
#O	Значение часа в 12-часовом формате без ведущего нуля (1, 11).
#P	Значение часа в 12-часовом формате с ведущим нулем (01, 11).
#Q	Значение минут без ведущего нуля (5).
#R	Значение минут с ведущим нулем (05).
#S	Значение секунд без ведущего нуля (5).
#T	Значение секунд с ведущим нулем (05).
#U	Значение миллисекунд без ведущего нуля (5).
#V	Значение миллисекунд с ведущим нулем (005).
#W	Значение дня с ведущим нулем и пробелом (‘ 05’).
#X	Значение месяца с ведущим нулем и пробелом (‘ 05’).

13.17. DWORD_TO_STRB

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	IN	DWORD	Набор бит.
Выходы	DWORD_TO_STRB	STRING(32)	Строка бит.

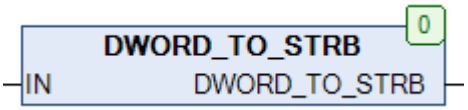


Рис. 13.33. Внешний вид функции **DWORD_TO_STRB** на языке CFC

Функция **DWORD_TO_STRB** конвертирует переменную **IN** типа **DWORD**, представляющую собой набор бит, в строку фиксированной длины, состоящую из символов ‘0’ и ‘1’. При необходимости можно подавать на вход функции переменные типа **BYTE** и **WORD**, обрезая строку до нужных размеров с помощью функции **RIGHT** из библиотеки **Standard**.

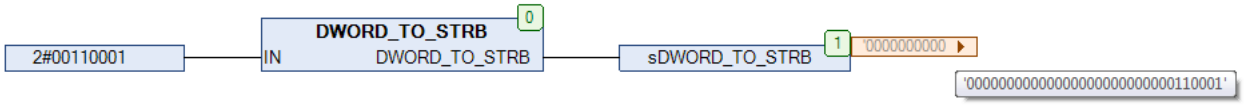


Рис. 13.34. Пример работы с функцией **DWORD_TO_STRB** на языке CFC

13.18. DWORD_TO_STRF

Тип модуля: функция	Переменная	Тип	Описание
Входы	IN	DWORD	Заданное значение.
	N	INT	Длина строки.
Выходы	DWORD_TO_STRF	STRING(20)	Значение в виде строки заданной длины.
Используемые модули	FIX		

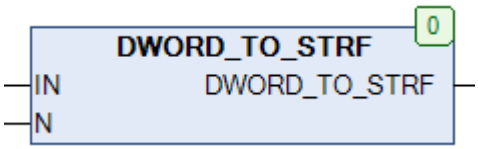


Рис. 13.35. Внешний вид функции **DWORD_TO_STRF** на языке CFC

Функция **DWORD_TO_STRF** конвертирует значение переменной **IN** типа **DWORD** в строку с заданной длиной **N**. Если **N** > кол-ва разрядов I**N**, то строка дополняется ведущими нулями. Если **N** < кол-ва разрядов **IN**, то старшие разряды обрезаются.

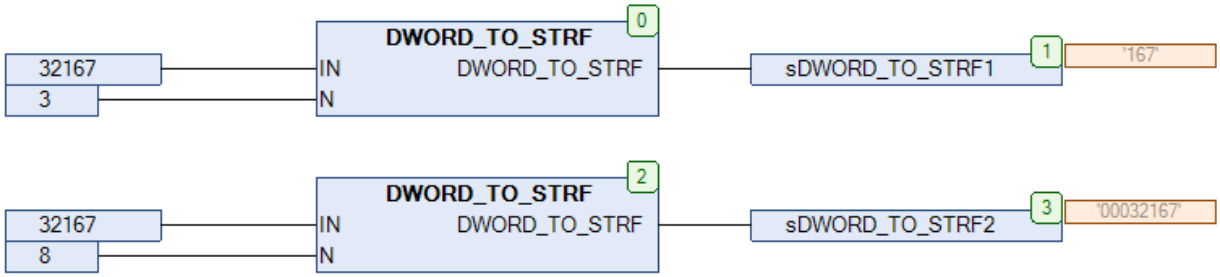


Рис. 13.36. Пример работы с функцией **DWORD_TO_STRF** на языке CFC

13.19. DWORD_TO_STRH

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	IN	DWORD	Заданное значение.
Выходы	DWORD_TO_STRH	STRING(8)	Значение в виде HEX-форматированной строки.

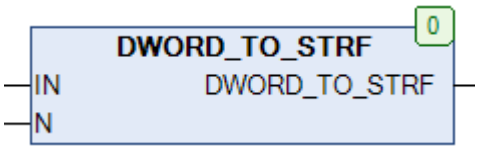


Рис. 13.37. Внешний вид функции **DWORD_TO_STRH** на языке CFC

Функция **DWORD_TO_STRH** конвертирует значение переменной **IN** типа **DWORD** в строку фиксированной длины, содержащую это значение в 16-ричной системе счисления.

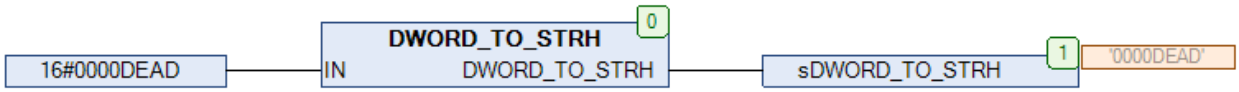


Рис. 13.38. Пример работы с функцией **DWORD_TO_STRH** на языке CFC

13.20. EXEC

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	str	STRING	Математическое выражение.
Выходы	EXEC	STRING	Результат.
Используемые модули	UPPERCASE, TRIM, FIND_NONUM		

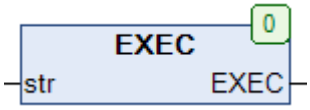


Рис. 13.39. Внешний вид функции **EXEC** на языке CFC

Функция **EXEC** вычисляет математическое выражение, заданное в виде строки **str**, и возвращает его результат (также в виде строки). Выражение может включать только два операнда и не должно содержать скобки. В случае некорректной операции (например, деления на ноль) функция возвращает значение '**ERROR**'.

Список возможных операторов: +, -, *, /, ^, SIN, CON, TAN, SQRT.

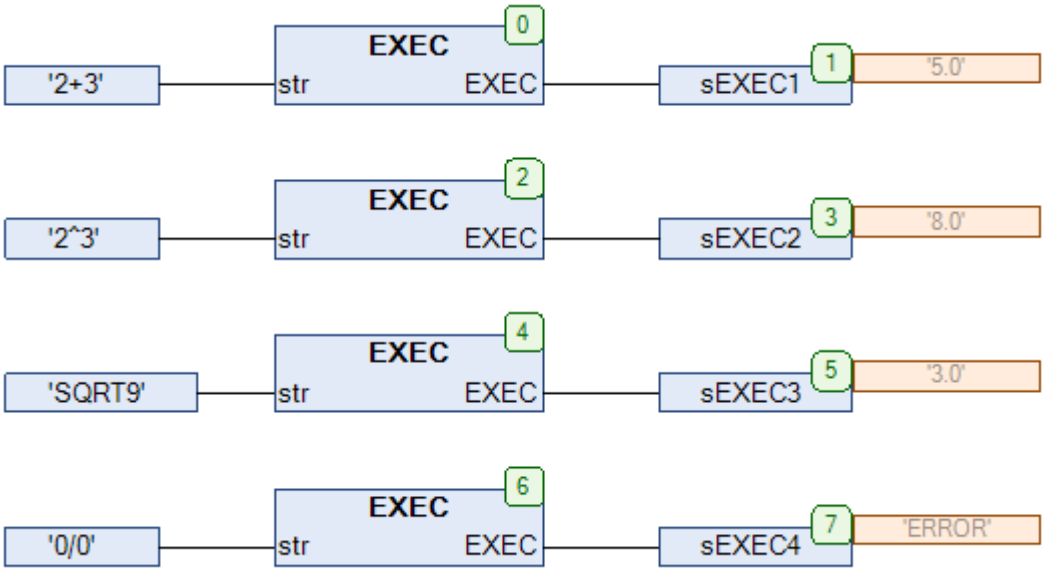


Рис. 13.40. Пример работы с функцией **EXEC** на языке CFC

13.21. FILL

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	C	BYTE	ASCII-код символа.
	L	INT	Длина строки.
Выходы	FILL	STRING	Строка заданной длины, заполненная заданным символом.
Используемые модули	CHR_TO_STRING		

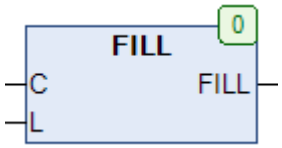


Рис. 13.41. Внешний вид функции **FILL** на языке CFC

Функция **FILL** формирует строку длиной **L**, заполненную символом с ASCII-кодом **C**.

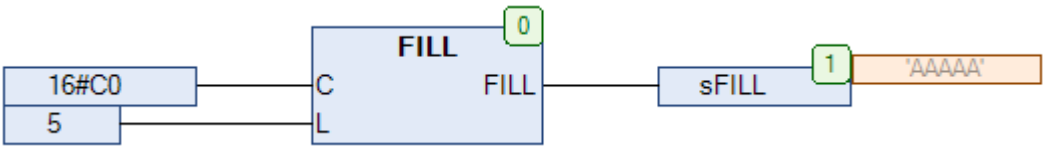


Рис. 13.42. Пример работы с функцией **FILL** на языке CFC

13.22. FIND_CHAR

Тип модуля: функция	Переменная	Тип	Описание
Входы	str	STRING	Исходная строка.
	pos	POS	Начальная позиция для поиска.
Выходы	FIND_CHAR	INT	Позиция первого символа, который не является управляющим.

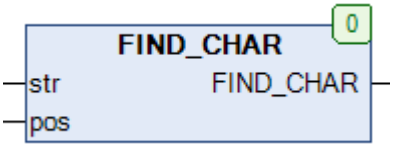


Рис. 13.43. Внешний вид функции **FIND_CHAR** на языке CFC

Функция **FIND_CHAR** анализирует исходную строку **str** с позиции **pos** и возвращает позицию первого символа, который не является управляющим. Для работы с символами верхней половины таблицы ASCII глобальная переменная EXTENDED_ASCII должна иметь значение **TRUE**. Управляющими считаются символы с ASCII-кодом **< 27** и **127**.

На рисунке ниже приведена строка ‘$1812’, состоящая из трех символов. Символ $ означает, что следующие за ними цифры будут интерпретированы как ASCII-код – т.е. запись ‘$18’ представляет собой управляющий символ с ASCII-кодом 18 (HEX).

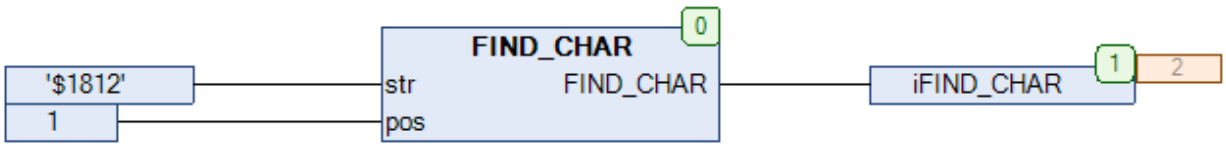


Рис. 13.44. Пример работы с функцией **FIND_CHAR** на языке CFC

13.23. FIND_CTRL

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	str	STRING	Исходная строка.
	pos	POS	Начальная позиция для поиска.
Выходы	FIND_CTRL	INT	Позиция первого управляющего символа в строке.

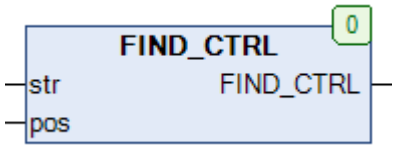


Рис. 13.45. Внешний вид функции **FIND_CTRL** на языке CFC

Функция **FIND_CTRL** анализирует исходную строку **str** с позиции **pos** и возвращает позицию первого управляющего символа. Для работы с символами верхней половины таблицы ASCII глобальная переменная EXTENDED_ASCII должна иметь значение **TRUE**. Управляющими считаются символы с ASCII-кодом **< 27** и **127**.

На рисунке ниже приведена строка ‘1$182’, состоящая из трех символов. Символ $ означает, что следующие за ними цифры будут интерпретированы как ASCII-код – т.е. запись ‘$18’ представляет собой управляющий символ с ASCII-кодом 18 (HEX).

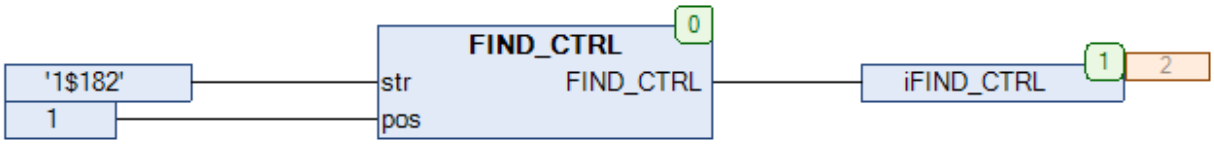


Рис. 13.46. Пример работы с функцией **FIND_CTRL** на языке CFC

13.24. FIND_NONUM

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	str	STRING	Исходная строка.
	pos	POS	Начальная позиция для поиска.
Выходы	FIND_NONUM	INT	Позиция первого символа, который не является числом.

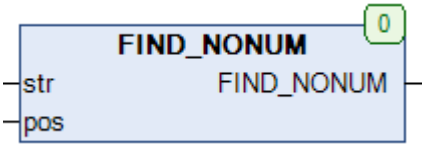


Рис. 13.47. Внешний вид функции **FIND_NONUM** на языке CFC

Функция **FIND_NONUM** анализирует исходную строку **str** с позиции **pos** и возвращает позицию первого символа, который не является числом (символом ‘1’ – ‘9’ или ‘.’).

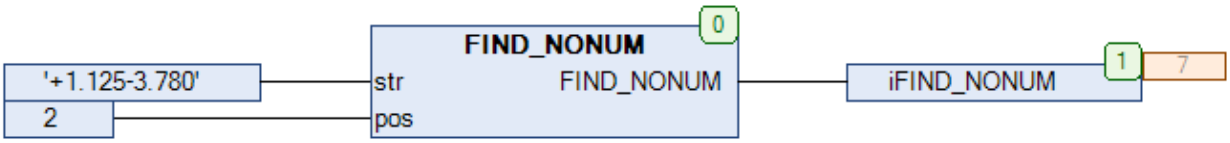


Рис. 13.48. Пример работы с функцией **FIND_NONUM** на языке CFC

13.25. FIND_NUM

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	str	STRING	Исходная строка.
	pos	POS	Начальная позиция для поиска.
Выходы	FIND_NUM	INT	Позиция первого символа, который является числом.

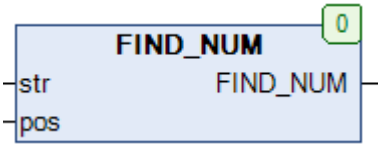


Рис. 13.49. Внешний вид функции **FIND_NUM** на языке CFC

Функция **FIND_NUM** анализирует исходную строку **str** с позиции **pos** и возвращает позицию первого символа, который является числом (символом ‘1’ – ‘9’ или ‘.’).

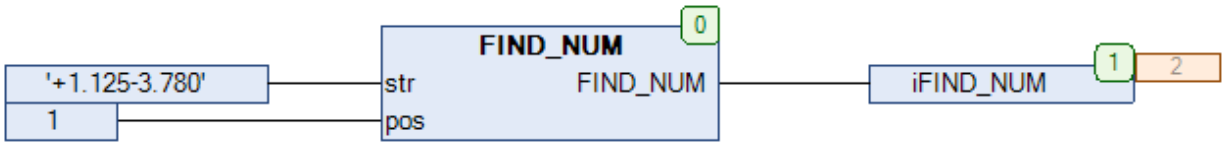


Рис. 13.50. Пример работы с функцией **FIND_NUM** на языке CFC

13.26. FINDB

Тип модуля: функция	Переменная	Тип	Описание
Входы	str1	STRING	Анализируемая строка.
	str2	STRING	Искомая строка.
Выходы	FINDB	INT	Позиция последнего вхождения искомой строки.

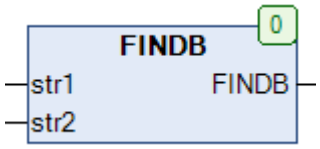


Рис. 13.51. Внешний вид функции **FINDB** на языке CFC

Функция **FINDB** возвращает позицию последнего вхождения строки **str2** в строку **str1**.

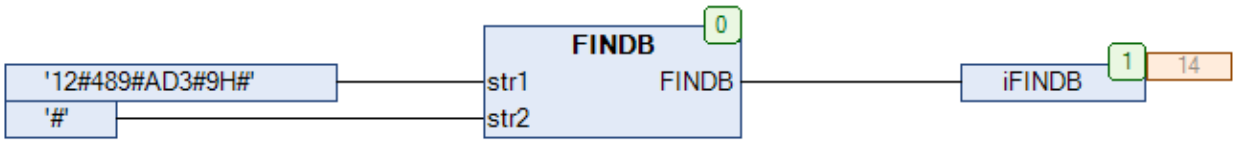


Рис. 13.52. Пример работы с функцией **FINDB** на языке CFC

13.27. FINDB_NONUM

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	str	STRING	Исходная строка.
Выходы	FIND_NONUM	INT	Позиция последнего символа, который не является числом.

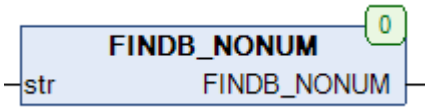


Рис. 13.53. Внешний вид функции **FINDB_NONUM** на языке CFC

Функция **FINDB_NONUM** возвращает позицию последнего символа строки **str**, который не является числом (символом ‘1’ – ‘9’ или ‘.’).

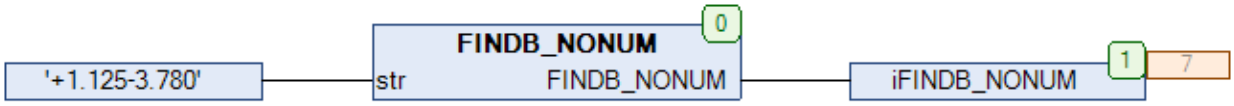


Рис. 13.54. Пример работы с функцией **FINDB_NONUM** на языке CFC

13.28. FINDB_NUM

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	str	STRING	Исходная строка.
Выходы	FINDB_NUM	INT	Позиция последнего символа, который является числом.

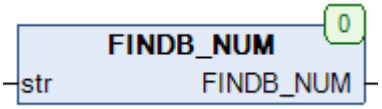


Рис. 13.55. Внешний вид функции **FINDB_NUM** на языке CFC

Функция **FINDB_NUM** возвращает позицию последнего символа строки **str**, который является числом (символом ‘1’ – ‘9’ или ‘.’).

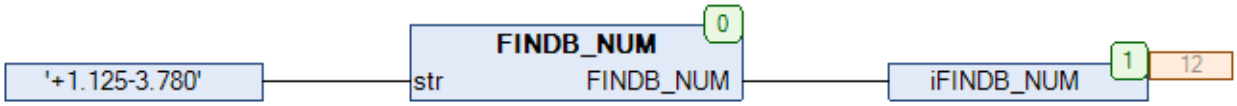


Рис. 13.56. Пример работы с функцией **FINDB_NUM** на языке CFC

13.29. FINDP

Тип модуля: функция	Переменная	Тип	Описание
Входы	str	STRING	Анализируемая строка.
	src	STRING	Искомая строка.
	pos	INT	Начальная позиция для поиска.
Выходы	FINDP	INT	Позиция первого вхождения искомой строки.

Рис. 13.57. Внешний вид функции **FINDP** на языке CFC

Функция **FINDP** анализирует исходную строку **str** с позиции **pos** и возвращает позицию первого вхождения в нее строки **src**.

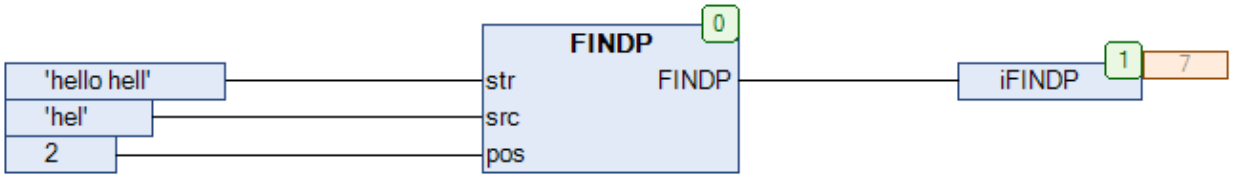


Рис. 13.58. Пример работы с функцией **FINDP** на языке CFC

13.30. FIX

Тип модуля: функция	Переменная	Тип	Описание
Входы	str	STRING	Исходная строка.
	C	BYTE	ASCII-код символа-заполнителя.
	L	INT	Длина дополненной строки.
	M	INT	Режим заполнения.
Выходы	FIX	STRING	Дополненная строка.
Используемые модули	FILL		

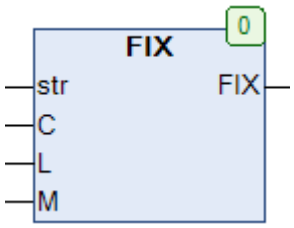


Рис. 13.59. Внешний вид функции **FIX** на языке CFC

Функция **FIX** дополняет исходную строку **str** до длины **L** с помощью символа-заполнителя с ASCII-кодом **C**. Переменная **M** определяет режим дополнения:

- M=0 – строка дополняется справа;
- M=1 – строка дополняется слева;
- M=2 – строка дополняется с обоих сторон. Если длина **str** является нечетной, то правая часть дополненной строки будет больше левой.

Если L ≤ длина str, то исходная строка не дополняется, а обрезается слева (для M=0 и M=2) или справа (для M=1).

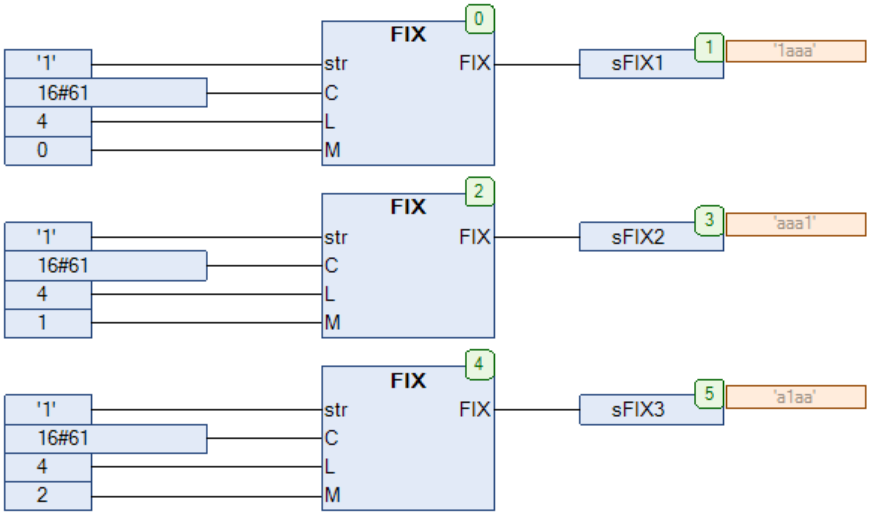


Рис. 13.60. Пример работы с функцией **FIX** на языке CFC

13.31. FLOAT_TO_REAL

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	FLT	STRING(20)	Значение с плавающей точкой в формате STRING.
Выходы	FLOAT_TO_REAL	REAL	Значение с плавающей точкой в формате REAL.
Используемые модули	DEC_TO_INT, EXPN		

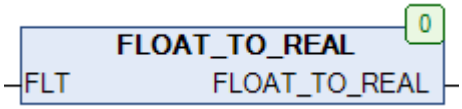


Рис. 13.61. Внешний вид функции **FLOAT_TO_REAL** на языке CFC

Функция **FLOAT_TO_REAL** конвертирует строку **FLT**, содержащую значение с плавающей точкой, в переменную типа **REAL**.

Строка **FLT** может содержать цифры и символы «.», «,», «-», «E», «e».

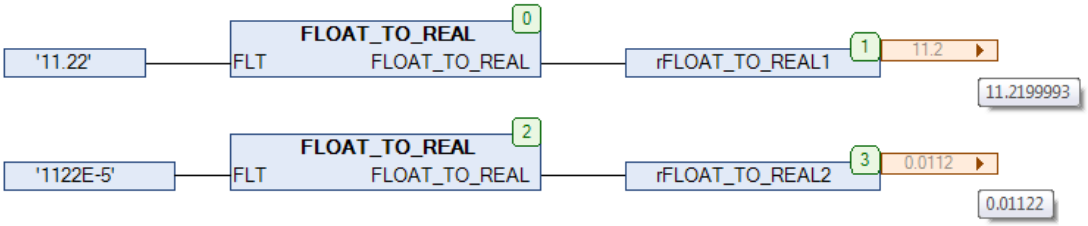


Рис. 13.62. Пример работы с функцией **FLOAT_TO_REAL** на языке CFC

13.32. FSTRING_TO_BYTE

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	IN	STRING(12)	Значение байта в виде форматированной строки.
Выходы	FSTRING_TO_BYTE	BYTE	Значение байта.
Используемые модули	BIN_TO_BYTE, OCT_TO_BYTE, HEX_TO_BYTE, DEC_TO_BYTE		

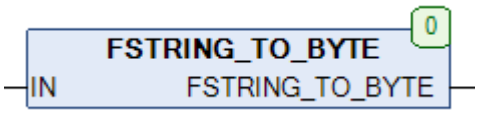


Рис. 13.63. Внешний вид функции **FSTRING_TO_BYTE** на языке CFC

Функция **FSTRING_TO_BYTE** конвертирует форматированную строку **IN**, содержащую значение байта, в переменную типа **BYTE**. Поддерживаются следующие форматы:

- двоичный (‘2#01100100’);
- восьмеричный (‘8#144’);
- шестнадцатеричный (‘16#64FS’);
- десятичный (‘100’).

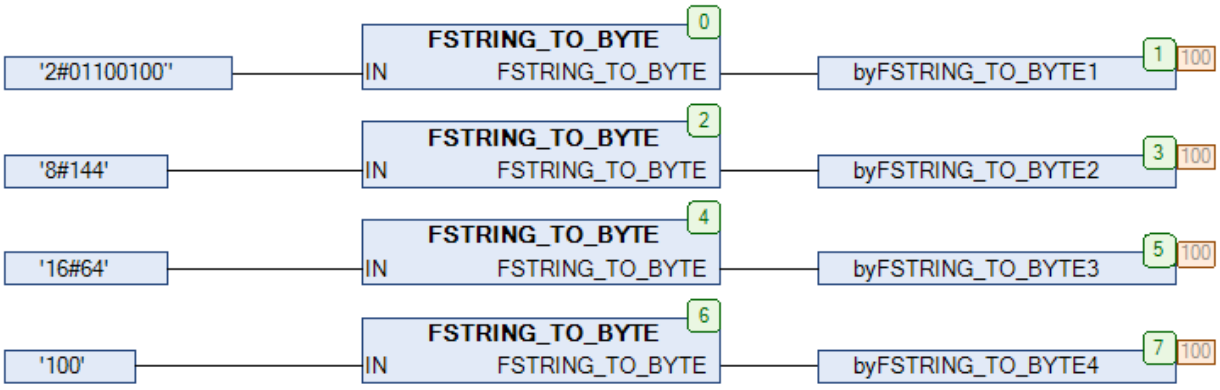


Рис. 13.64. Пример работы с функцией **FSTRING_TO_BYTE** на языке CFC

13.33. FSTRING_TO_DT

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	SDT	STRING(60)	Значение даты и времени в виде строки.
	FMT	STRING(60)	Формат строки.
Выходы	FSTRING_TO_DT	DT	Дата и время.
Используемые модули	SET_DT		

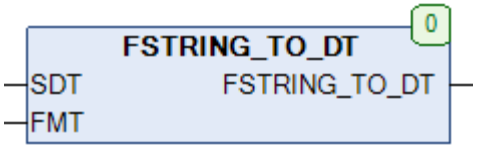


Рис. 13.65. Внешний вид функции **FSTRING_TO_DT** на языке CFC

Функция **FSTRING_TO_DT** конвертирует форматированную строку **SDT**, содержащую значение даты и времени, в переменную типа **DT**. Вход **FMT** определяет используемое форматирование, т.е. представляет строку **SDT** в виде последовательности заполнителей. Список заполнителей приведен ниже.

Заполнитель	**Описание**
#Y	Значение года (08 или 2008).
#M	Значение месяца в виде числа (1 или 01).
#N	Значение месяца в виде названия (Jan или January). Регистр не учитывается.
#D	Значение дня (1 или 01).
#h	Значение часов (1 или 01).
#m	Значение минут (1 или 01).
#s	Значение секунд (1 или 01).
*	Заполнитель для статического текста (см. рис. 13.66).

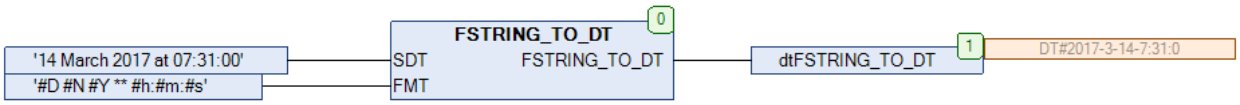


Рис. 13.66. Пример работы с функцией **FSTRING_TO_DT** на языке CFC

13.34. FSTRING_TO_DWORD

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	IN	STRING(40)	Значение DWORD в виде форматированной строки.
Выходы	FSTRING_TO_DWORD	DWORD	Значение DWORD.
Используемые модули	BIN_TO_DWORD, OCT_TO_DWORD, HEX_TO_DWORD, DEC_TO_DWORD		

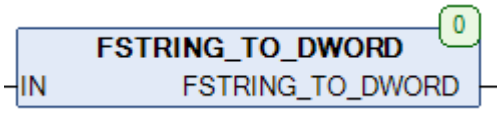


Рис. 13.67. Внешний вид функции **FSTRING_TO_DWORD** на языке CFC

Функция **FSTRING_TO_DWORD** конвертирует форматированную строку **IN**, содержащую значение **DWORD**, в переменную типа **DWORD**. Поддерживаются следующие форматы:

- двоичный (‘2#01100100’);
- восьмеричный (‘8#144’);
- шестнадцатеричный (‘16#64FS’);
- десятичный (‘100’).

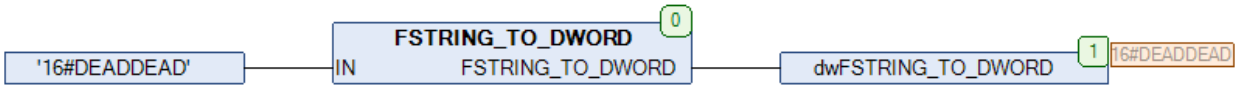


Рис. 13.68. Пример работы с функцией **FSTRING_TO_DWORD** на языке CFC

13.35. FSTRING_TO_MONTH

Тип модуля: функция	Переменная	Тип	Описание
Входы	MTH	STRING(20)	Значение месяца в виде строки.
	LANG	INT	Язык приложения.
Выходы	FSTRING_TO_MONTH	INT	Значение месяца в виде числа.
Используемые модули	TRIM, CAPITALIZE, LOWERCASE		

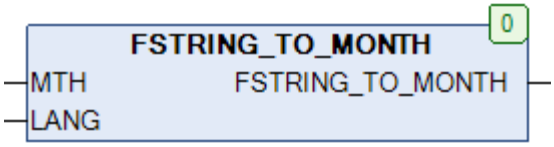


Рис. 13.69. Внешний вид функции **FSTRING_TO_MONTH** на языке CFC

Функция **FSTRING_TO_MONTH** конвертирует строку, содержащую значение месяца в виде числа или названия, в переменную типа **INT**, содержащую номер месяца. Вход **LANG** определяет язык приложения (см. глобальную переменную LANGUAGE).

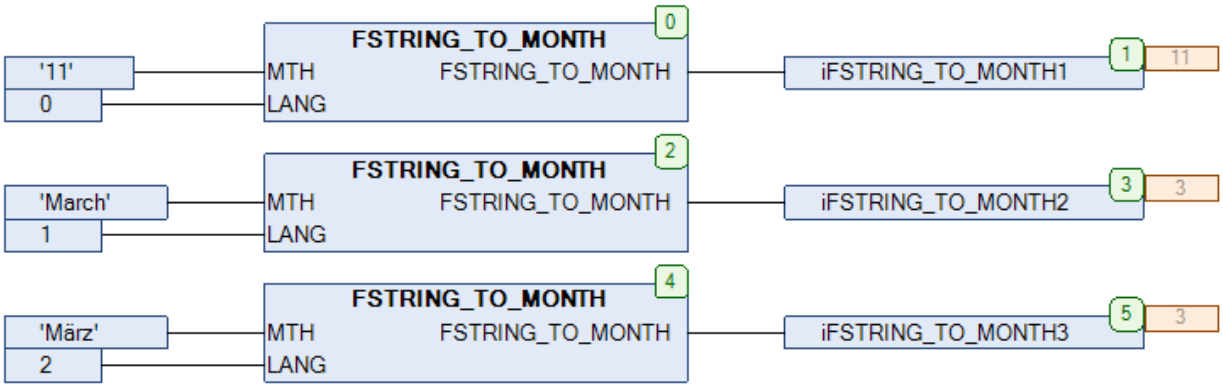


Рис. 13.70. Пример работы с функцией **FSTRING_TO_MONTH** на языке CFC

13.36. FSTRING_TO_WEEK

Тип модуля: функция	Переменная	Тип	Описание
Входы	WEEK	STRING(60)	Название или номер для недели.
	LANG	INT	Язык приложения.
Выходы	FSTRING_TO_WEEK	BYTE	Номер дня недели в виде битовой маски.
Используемые модули	FSTRING_TO_WEEKDAY		

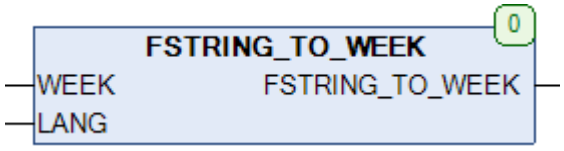


Рис. 13.71. Внешний вид функции **FSTRING_TO_WEEK** на языке CFC

Функция **FSTRING_TO_WEEK** конвертирует строку **WEEK**, содержащую номер или название (полное или сокращенное, регистронезависимое) дня недели, в битовую маску (типа **BYTE**), где бит, имеющий значение **TRUE**, определяет текущий день недели (**бит 6** – понедельник, **бит 0** – воскресение). Вход **LANG** определяет язык приложения (см. глобальную переменную LANGUAGE).

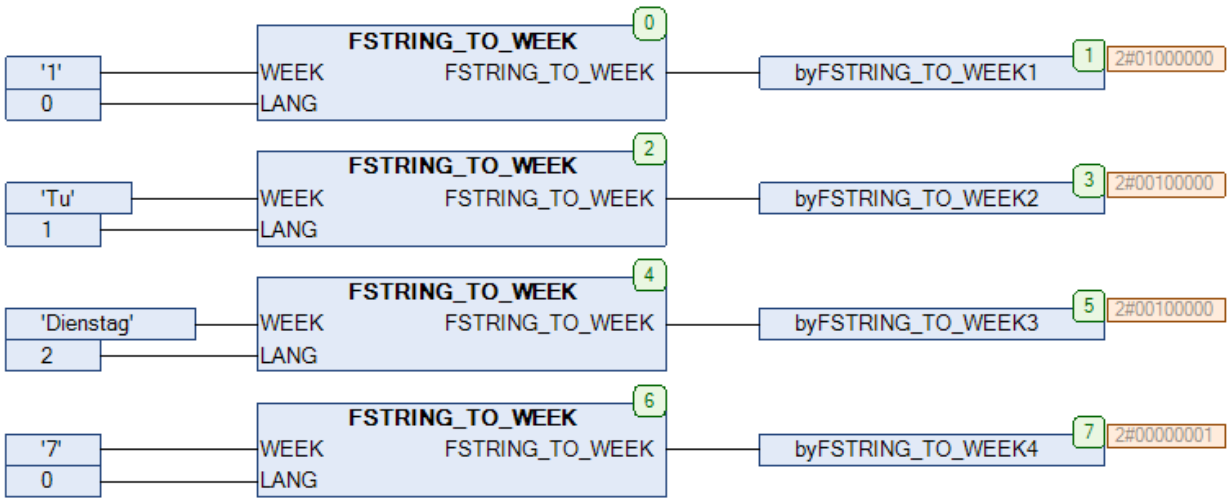


Рис. 13.72. Пример работы с функцией **FSTRING_TO_WEEK** на языке CFC

13.37. FSTRING_TO_WEEKDAY

Тип модуля: функция	Переменная	Тип	Описание
Входы	WEEK	STRING(20)	Название или номер для недели.
	LANG	INT	Язык приложения.
Выходы	FSTRING_TO_WEEKDAY	INT	Номер дня недели.
Используемые модули	TRIM, CAPITALIZE, LOWERCASE		

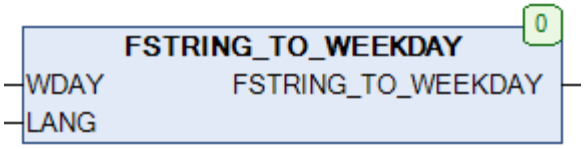


Рис. 13.73. Внешний вид функции **FSTRING_TO_WEEKDAY** на языке CFC

Функция **FSTRING_TO_WEEKDAY** конвертирует строку **WEEK**, содержащую номер или название (полное или сокращенное, регистронезависимое) дня недели в переменную типу **INT**, содержащую номер дня недели (**1** – понедельник, **7** – воскресение). Вход **LANG** определяет язык приложения (см. глобальную переменную LANGUAGE).

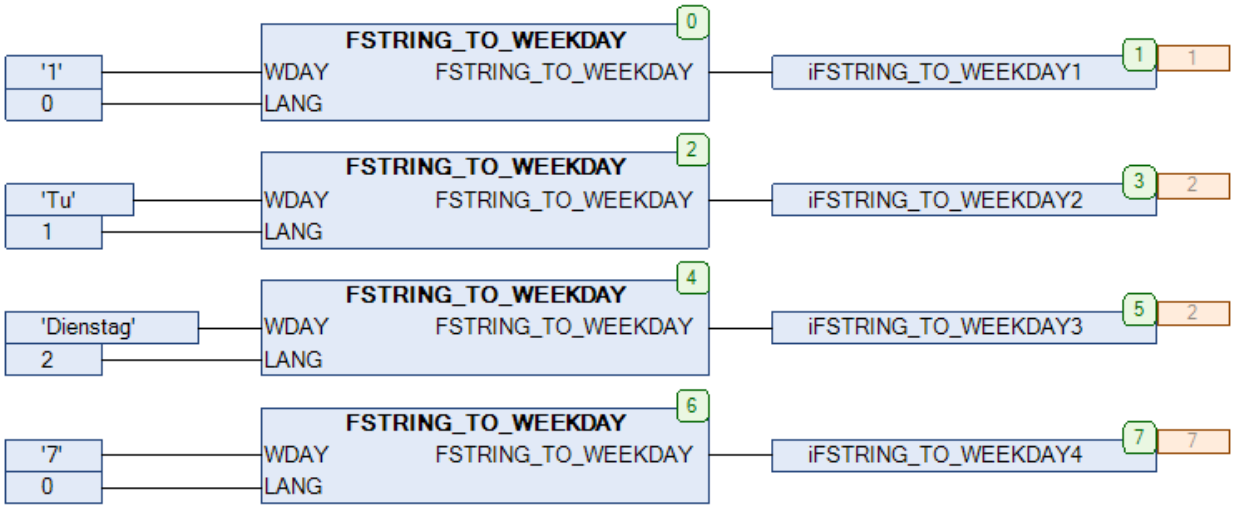


Рис. 13.74. Пример работы с функцией **FSTRING_TO_WEEKDAY** на языке CFC

13.38. HEX_TO_BYTE

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	HEX	STRING(5)	HEX-значение в виде строки.
Выходы	HEX_TO_BYTE	BYTE	HEX-значение в виде байта.

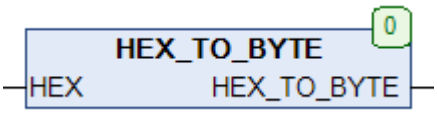


Рис. 13.75. Внешний вид функции HEX_TO_BYTE на языке CFC

Функция **HEX_TO_BYTE** конвертирует строку **HEX**, содержащую значение в 16-ричной системе счисления, в соответствующее значение типа **BYTE**. Исходная строка должна содержать только числа и символы от ‘A’ до ‘F’ (в любом регистре).

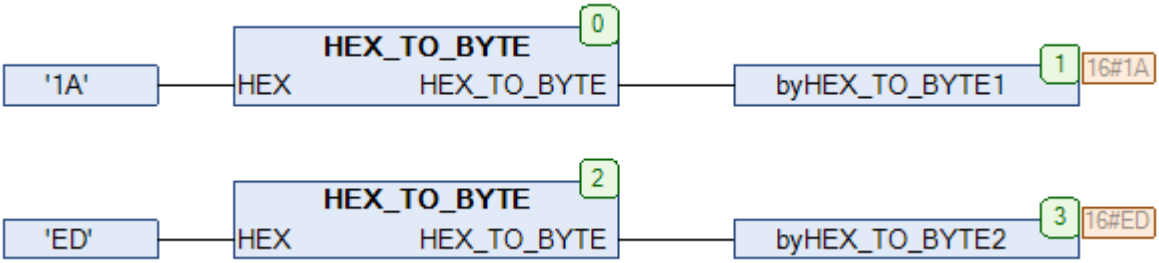


Рис. 13.76. Пример работы с функцией HEX_TO_BYTE на языке CFC

13.39. HEX_TO_DWORD

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	HEX	STRING(20)	HEX-значение в виде строки.
Выходы	HEX_TO_DWORD	DWORD	HEX-значение в виде DWORD.

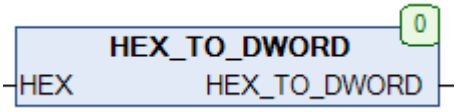


Рис. 13.77. Внешний вид функции **HEX_TO_DWORD** на языке CFC

Функция **HEX_TO_DWORD** конвертирует строку **HEX**, содержащую значение в 16-ричной системе счисления, в соответствующее значение типа **DWORD**. Исходная строка должна содержать только числа и символы от ‘A’ до ‘F’ (в любом регистре).

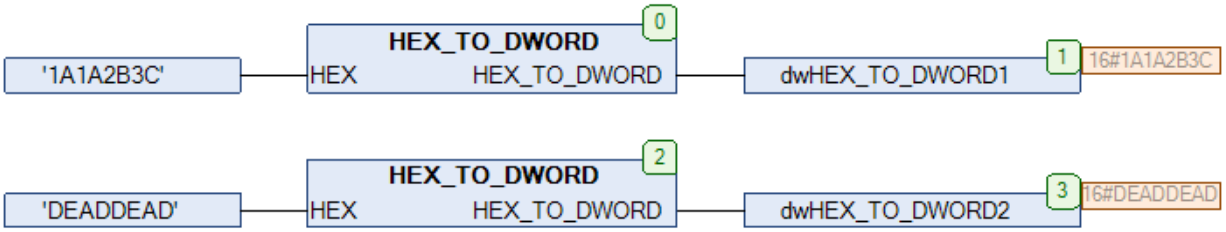


Рис. 13.78. Пример работы с функцией **HEX_TO_DWORD** на языке CFC

13.40. IS_ALNUM

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	str	STRING	Исходная строка.
Выходы	IS_ALNUM	BOOL	Флаг «только буквы и цифры».
Используемые модули	ISC_ALPHA, ISC_NUM		

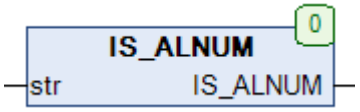


Рис. 13.79. Внешний вид функции **IS_ALNUM** на языке CFC

Функция **IS_ALNUM** возвращает **TRUE**, если исходная строка **str** состоит только из букв и цифр. Если в строку входит хотя бы один символ, не попадающий в эти категории (например, пробел), то функция возвращает **FALSE**.

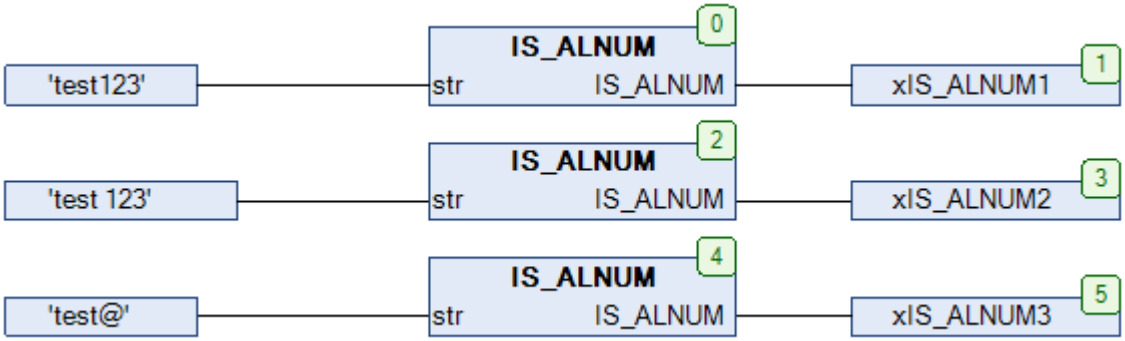


Рис. 13.80. Пример работы с функцией **IS_ALNUM** на языке CFC

13.41. IS_ALPHA

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	str	STRING	Исходная строка.
Выходы	IS_ALPHA	BOOL	Флаг «только буквы».
Используемые модули	ISC_ALPHA		

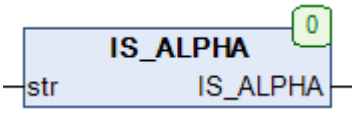


Рис. 13.81. Внешний вид функции **IS_ALPHA** на языке CFC

Функция **IS_ALPHA** возвращает **TRUE**, если исходная строка **str** состоит только из букв. Во всех остальных случаях функция возвращает **FALSE**. Для работы с символами верхней половины таблицы ASCII глобальная переменная EXTENDED_ASCII должна иметь значение **TRUE**.

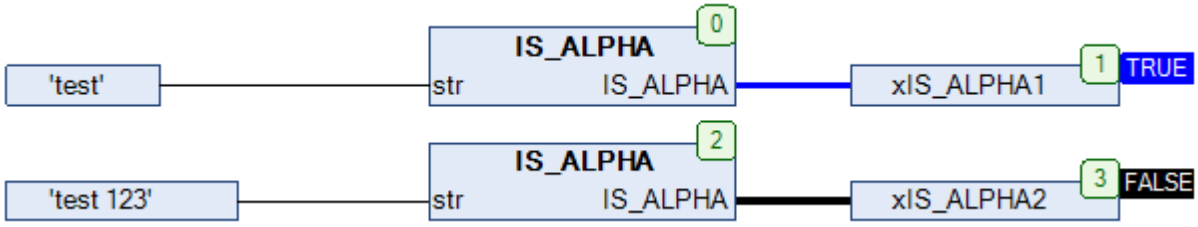


Рис. 13.82. Пример работы с функцией **IS_ALPHA** на языке CFC

13.42. IS_CC

Тип модуля: функция	Переменная	Тип	Описание
Входы	str	STRING	Исходная строка.
	cmp	STRING	Строка с разрешенными символами.
Выходы	IS_CC	BOOL	Флаг «только разрешенные символы».

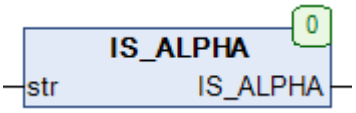


Рис. 13.83. Внешний вид функции **IS_CC** на языке CFC

Функция **IS_CC** возвращает **TRUE**, если исходная строка **str** состоит только из символов, включенных в строку **cmp**. Во всех остальных случаях функция возвращает **FALSE**.

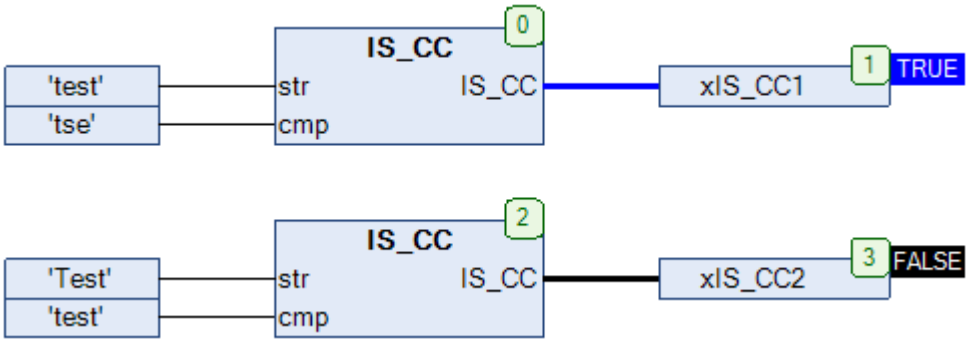


Рис. 13.84. Пример работы с функцией **IS_CC** на языке CFC

13.43. IS_CTRL

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	str	STRING	Исходная строка.
Выходы	IS_CTRL	BOOL	Флаг «только управляющие символы».
Используемые модули	ISC_CTRL		

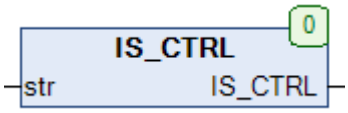


Рис. 13.85. Внешний вид функции **IS_CTRL** на языке CFC

Функция **IS_CTRL** возвращает **TRUE**, если исходная строка **str** состоит только из управляющих символов. Управляющими считаются символы с ASCII-кодом < 27 и 127.

На рисунке ниже в строках используется фрагмент ‘$18’. Символ $ означает, что следующие за ними цифры будут интерпретированы как ASCII-код – т.е. запись ‘$18’ представляет собой управляющий символ с ASCII-кодом 18 (HEX).

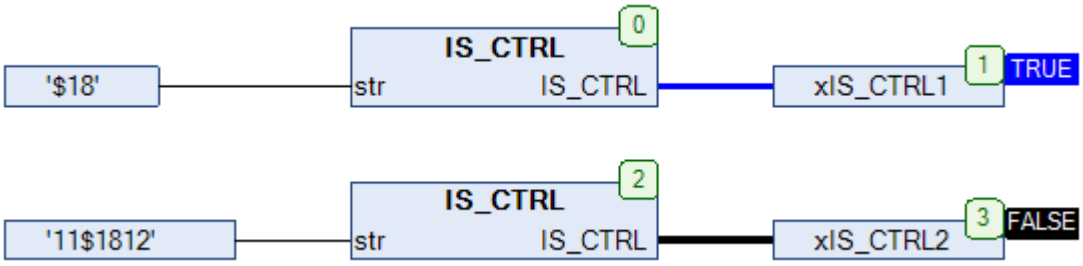


Рис. 13.86. Пример работы с функцией **IS_CTRL** на языке CFC

13.44. IS_HEX

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	str	STRING	Исходная строка.
Выходы	IS_HEX	BOOL	Флаг «только HEX-символы».
Используемые модули	ISC_HEX		

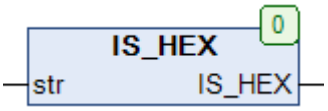


Рис. 13.87. Внешний вид функции **IS_HEX** на языке CFC

Функция **IS_HEX** возвращает **TRUE**, если исходная строка **str** состоит только из HEX-символов (‘0’ – ‘9’, ‘a’ – ‘f’, ‘A’ – ‘F’). Во всех остальных случаях функция возвращает **FALSE**.

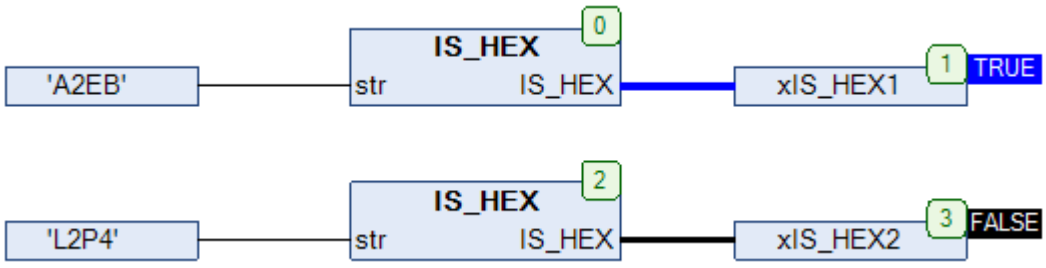


Рис. 13.88. Пример работы с функцией **IS_HEX** на языке CFC

13.45. IS_LOWER

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	str	STRING	Исходная строка.
Выходы	IS_LOWER	BOOL	Флаг «все символы в нижнем регистре».
Используемые модули	ISC_LOWER		

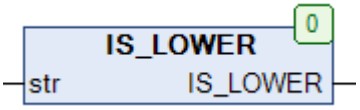


Рис. 13.89. Внешний вид функции **IS_LOWER** на языке CFC

Функция **IS_LOWER** возвращает **TRUE**, если исходная строка **str** состоит только из букв в нижнем регистре. Во всех остальных случаях функция возвращает **FALSE**. Для работы с символами верхней половины таблицы ASCII глобальная переменная EXTENDED_ASCII должна иметь значение **TRUE**.

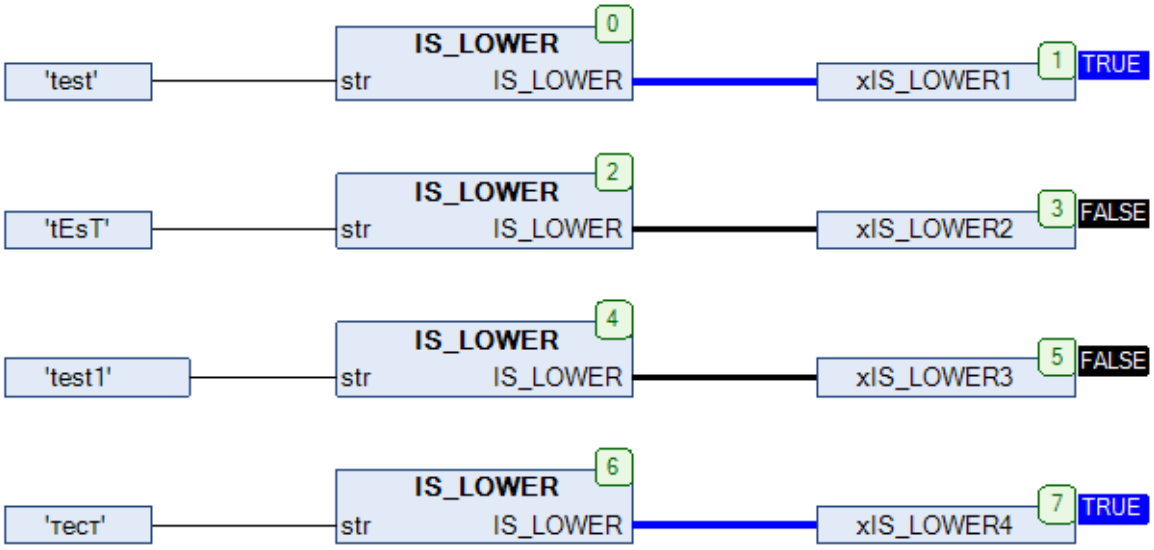


Рис. 13.90. Пример работы с функцией **IS_LOWER** на языке CFC

13.46. IS_NCC

Тип модуля: функция	Переменная	Тип	Описание
Входы	str	STRING	Исходная строка.
	cmp	STRING	Строка с запрещенными символами.
Выходы	IS_NCC	BOOL	Флаг «без запрещенных символов».

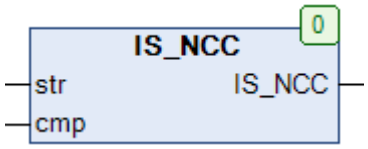


Рис. 13.91. Внешний вид функции **IS_NCC** на языке CFC

Функция **IS_NCC** возвращает **TRUE**, если исходная строка **str** не включает в себя ни один символ из строки **cmp**. Во всех остальных случаях функция возвращает **FALSE**.

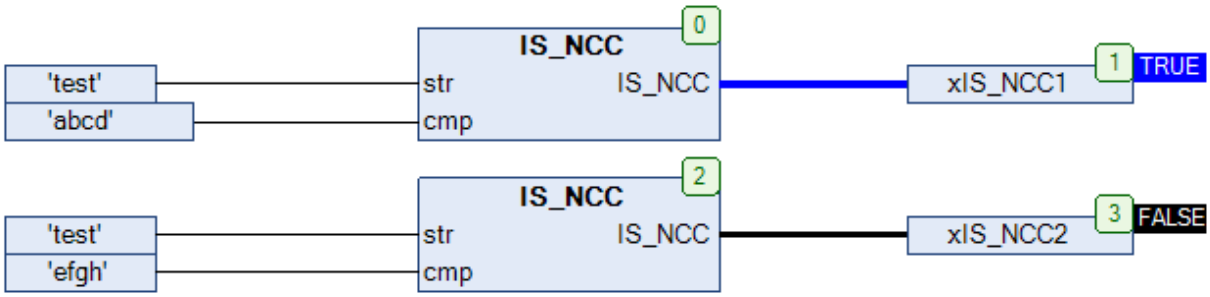


Рис. 13.92. Пример работы с функцией **IS_NCC** на языке CFC

13.47. IS_NUM

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	str	STRING	Исходная строка.
Выходы	IS_NUM	BOOL	Флаг «только числа».
Используемые модули	ISC_NUM		

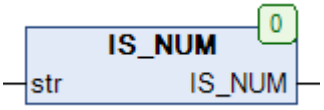


Рис. 13.93. Внешний вид функции **IS_NUM** на языке CFC

Функция **IS_NUM** возвращает **TRUE**, если исходная строка **str** состоит только из чисел. Во всех остальных случаях функция возвращает **FALSE**.

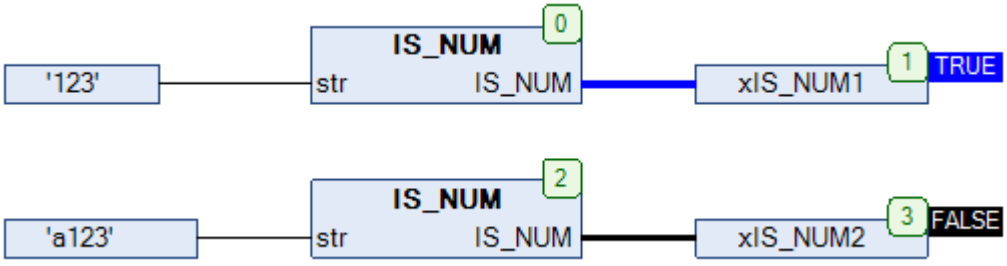


Рис. 13.94. Пример работы с функцией **IS_NUM** на языке CFC

13.48. IS_UPPER

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	str	STRING	Исходная строка.
Выходы	IS_UPPER	BOOL	Флаг «все символы в верхнем регистре».
Используемые модули	ISC_UPPER		

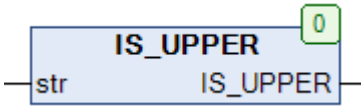


Рис. 13.95. Внешний вид функции IS_UPPER на языке CFC

Функция **IS_UPPER** возвращает **TRUE**, если исходная строка **str** состоит только из букв в верхнем регистре. Во всех остальных случаях функция возвращает **FALSE**. Для работы с символами верхней половины таблицы ASCII глобальная переменная EXTENDED_ASCII должна иметь значение **TRUE.**

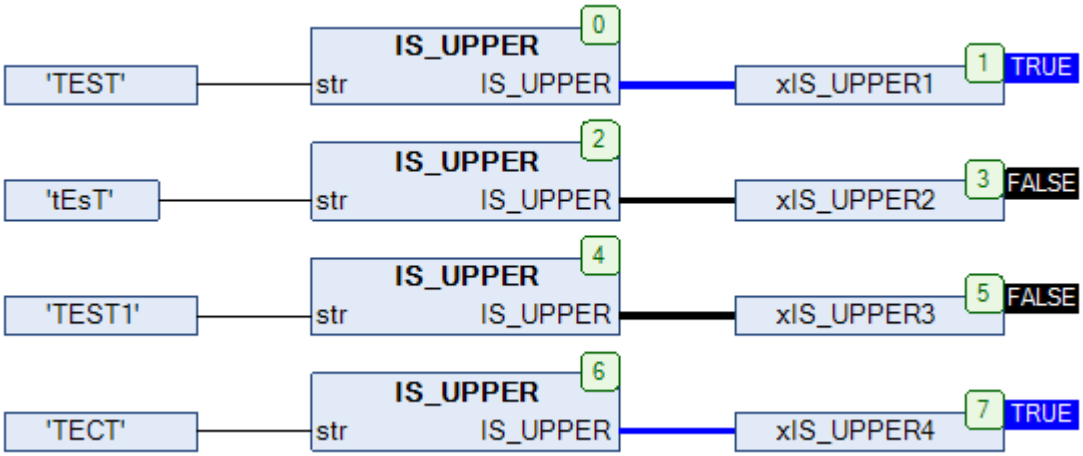


Рис. 13.96. Пример работы с функцией **IS_UPPER** на языке CFC

13.49. ISC_ALPHA

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	IN	BYTE	ASCII-код символа.
Выходы	ISC_ALPHA	BOOL	Флаг «буква».

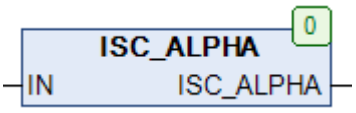


Рис. 13.97. Внешний вид функции **ISC_ALPHA** на языке CFC

Функция **ISC_ALPHA** возвращает **TRUE**, если символ с ASCII-кодом **IN** является буквой. Во всех остальных случаях функция возвращает **FALSE**. Для работы с символами верхней половины таблицы ASCII глобальная переменная EXTENDED_ASCII должна иметь значение **TRUE.**

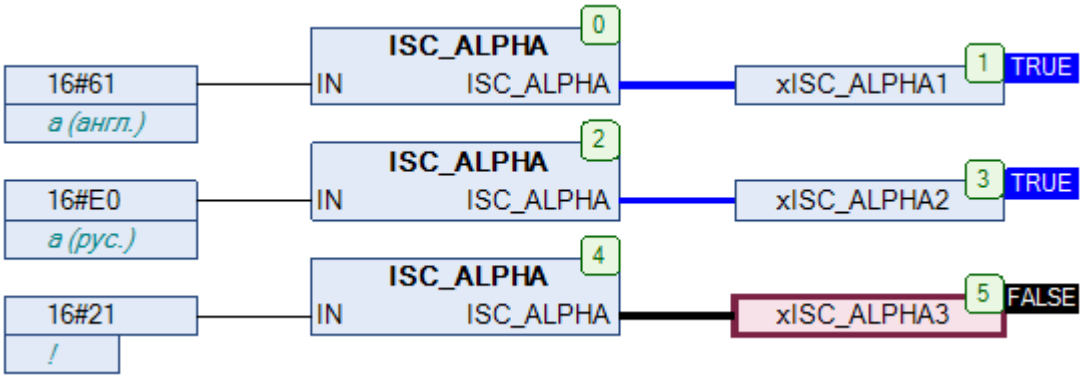


Рис. 13.98. Пример работы с функцией **ISC_ALPHA** на языке CFC

13.50. ISC_CTRL

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	IN	BYTE	ASCII-код символа.
Выходы	ISC_CTRL	BOOL	Флаг «управляющий символ».

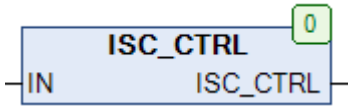


Рис. 13.99. Внешний вид функции **ISC_CTRL** на языке CFC

Функция **ISC_CTRL** возвращает **TRUE**, если символ с ASCII-кодом **IN** является управляющим символом. Управляющими считаются символы с ASCII-кодом < 27 (16#1B) и 127 (16#7F). Во всех остальных случаях функция возвращает **FALSE**.

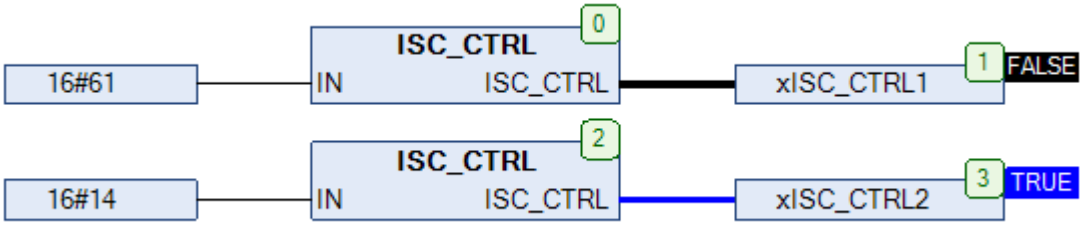


Рис. 13.100. Пример работы с функцией **ISC_CTRL** на языке CFC

13.51. ISC_HEX

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	IN	BYTE	ASCII-код символа.
Выходы	ISC_HEX	BOOL	Флаг «HEX-символ».

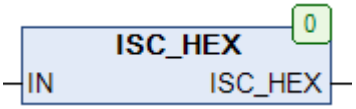


Рис. 13.101. Внешний вид функции **ISC_HEX** на языке CFC

Функция **ISC_HEX** возвращает **TRUE**, если символ с ASCII-кодом **IN** является HEX-символом (‘0’ – ‘9’, ‘a’ – ‘f’, ‘A’ – ‘F’). Во всех остальных случаях функция возвращает **FALSE**.

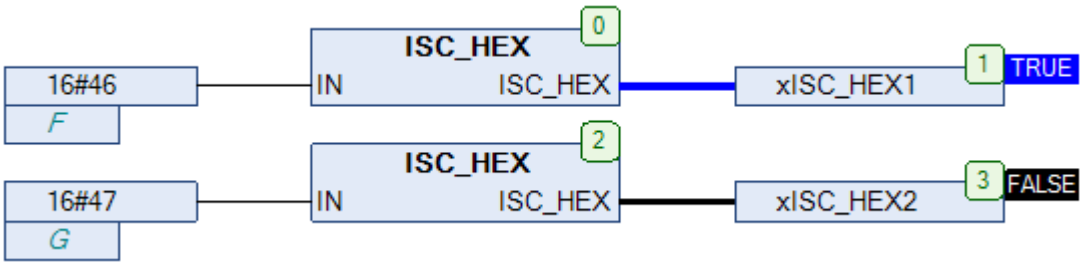


Рис. 13.102. Пример работы с функцией **ISC_HEX** на языке CFC

13.52. ISC_LOWER

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	IN	BYTE	ASCII-код символа.
Выходы	ISC_LOWER	BOOL	Флаг «буква в нижнем регистре».

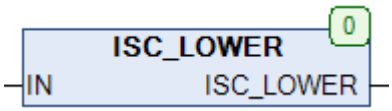


Рис. 13.103. Внешний вид функции **ISC_LOWER** на языке CFC

Функция **ISC_LOWER** возвращает **TRUE**, если символ с ASCII-кодом **IN** является буквой в нижнем регистре. Во всех остальных случаях функция возвращает **FALSE**. Для работы с символами верхней половины таблицы ASCII глобальная переменная EXTENDED_ASCII должна иметь значение **TRUE.**

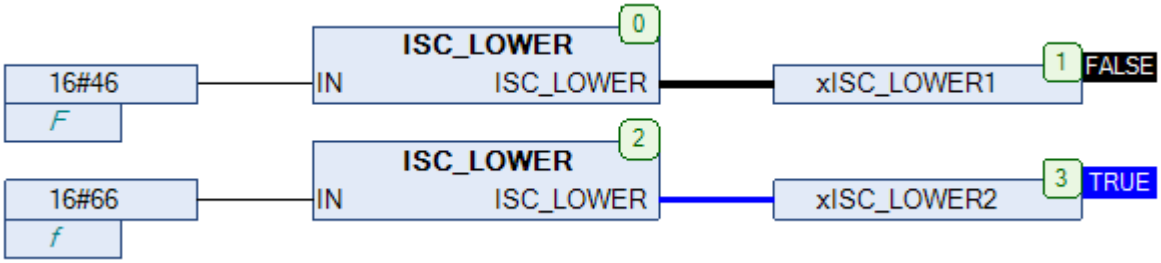


Рис. 13.104. Пример работы с функцией **ISC_LOWER** на языке CFC

13.53. ISC_NUM

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	IN	BYTE	ASCII-код символа.
Выходы	ISC_NUM	BOOL	Флаг «цифра».

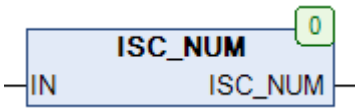


Рис. 13.105. Внешний вид функции **ISC_NUM** на языке CFC

Функция **ISC_NUM** возвращает **TRUE**, если символ с ASCII-кодом **IN** является цифрой. Во всех остальных случаях функция возвращает **FALSE**.

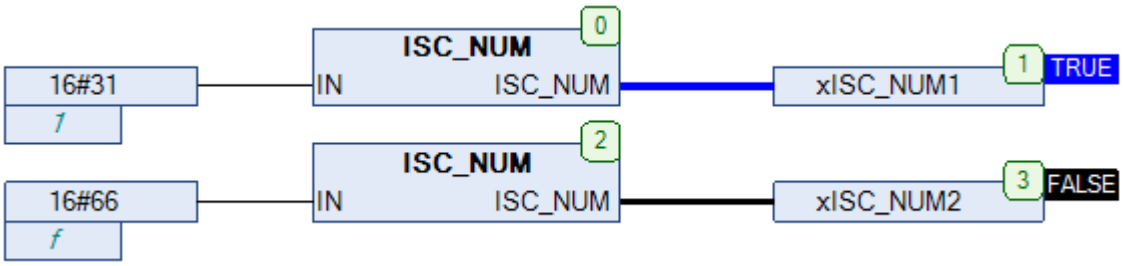


Рис. 13.106. Пример работы с функцией **ISC_NUM** на языке CFC

13.54. ISC_UPPER

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	IN	BYTE	ASCII-код символа.
Выходы	ISC_UPPER	BOOL	Флаг «буква в верхнем регистре».

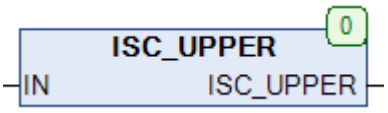


Рис. 13.107. Внешний вид функции **ISC_UPPER** на языке CFC

Функция **ISC_UPPER** возвращает **TRUE**, если символ с ASCII-кодом **IN** является буквой в верхнем регистре. Во всех остальных случаях функция возвращает **FALSE**. Для работы с символами верхней половины таблицы ASCII глобальная переменная EXTENDED_ASCII должна иметь значение **TRUE.**

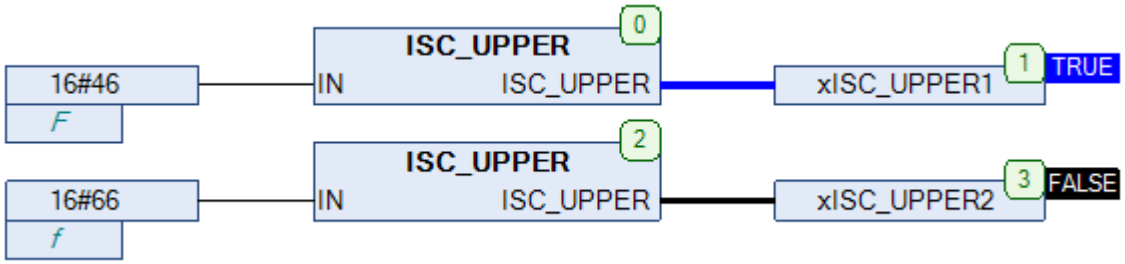


Рис. 13.108. Пример работы с функцией **ISC_UPPER** на языке CFC

13.55. LOWERCASE

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	str	STRING	Исходная строка.
Выходы	LOWERCASE	STRING	Строка в нижнем регистре.
Используемые модули	TO_LOWER		

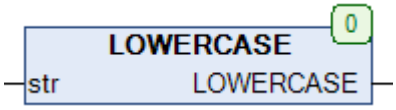


Рис. 13.109. Внешний вид функции **LOWERCASE** на языке CFC

Функция **LOWERCASE** конвертирует исходную строку **str** в соответствующую строку нижнего регистра. Для работы с символами верхней половины таблицы ASCII глобальная переменная EXTENDED_ASCII должна иметь значение **TRUE**.

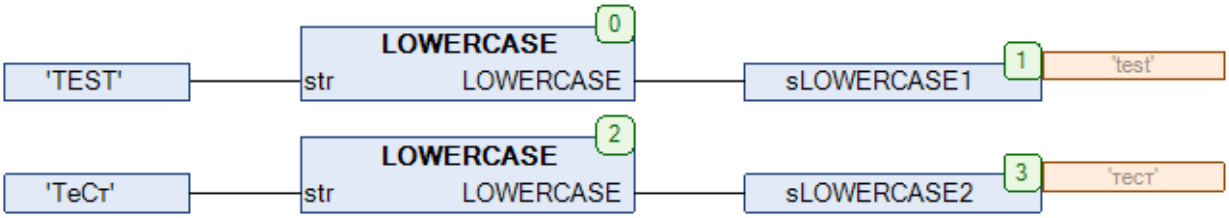


Рис. 13.110. Пример работы с функцией **LOWERCASE** на языке CFC

13.56. MESSAGE_4R

Тип модуля: ФБ	Переменная	Тип	Описание
Входы	M0…M3	STRING	Сообщения 0…3.
	MM	INT	Число используемых сообщений.
	ENQ	BOOL	Сигнал запуска ФБ.
	CLK	BOOL	Сигнал переключения сообщений.
	T1	TIME	Время автоматического переключения сообщений.
Выходы	MX	STRING	Текущее сообщение.
	MN	INT	Номер текущего сообщения (0...4)
	TR	BOOL	Флаг «новое сообщение».
Используемые модули	INC1		

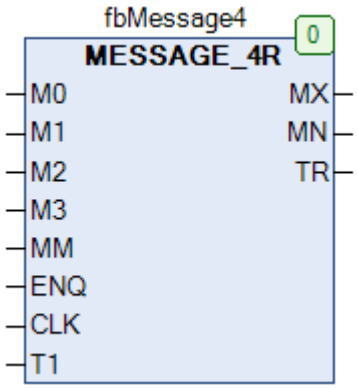


Рис. 13.111. Внешний вид ФБ **MESSAGE_4R** на языке CFC

Функциональный блок **MESSAGE_4R** используется для последовательной циклической генерации на выходе 4-х заданных информационных сообщений по фронту логического сигнала. Когда на входе **CLK** проходит импульс по переднему фронту, то на выход **MX** поступает сообщение **M0**. При следующем импульсе на входе **CLK** на выход попадет сообщение **M1** и т.д. Вход **MM** определяет число используемых сообщений (например, при MM=2 на выход будут поступать только сообщения M0 и M1). Вход **ENQ** используется для запуска/остановки работы блока (блок работает только при ENQ=TRUE). Вход **T1** используется для автоматического переключения сообщений на выходе **MX** – если в течение времени **T1** вход **CLK** сохраняет значение **TRUE**, то на выход **MX** поступает следующее сообщение. Выход **MN** содержит номер текущего сообщения на выходе (**0** – M0, **3** – M3). На выходе **TR** генерируется единичный импульс при смене сообщения.

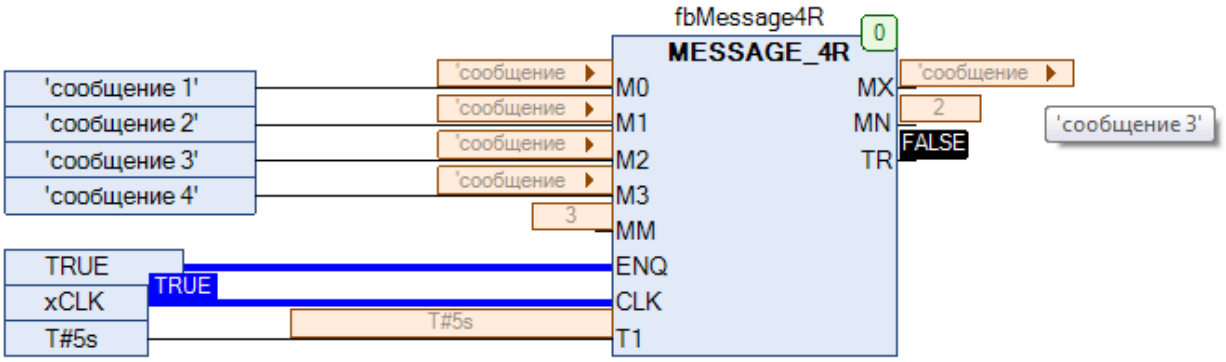


Рис. 13.112. Пример работы с ФБ **MESSAGE_4R** на языке CFC

13.57. MESSAGE_8

Тип модуля: ФБ	Переменная	Тип	Описание
Входы	IN1…IN8	BOOL	Сигналы генерации сообщений.
Выходы	M	STRING	Текущее сообщение.
Параметры	S1…S8	STRING	Сообщения 1…8.

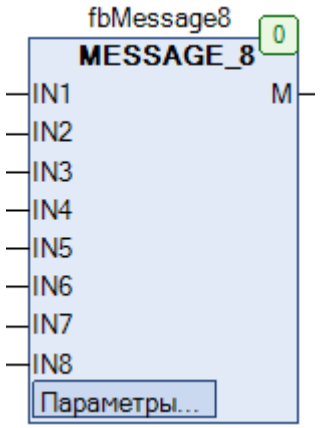


Рис. 13.113. Внешний вид ФБ **MESSAGE_8** на языке CFC

Функциональный блок **MESSAGE_8** используется для генерации на выходе одного из 8-ми заданных сообщений по фронту соответствующего логического сигнала. Если логический сигнал **IN1…IN8** имеет значение **TRUE**, то на выход **M** поступает соответствующее ему сообщение. Если несколько сигналов имеют значение **TRUE**, то на выход поступает сообщение с наименьшим номером (т.е. если IN1=IN2=IN3=TRUE, то на выходе будет сообщение S1; если S3=S5=S8=TRUE, то на выходе будет сообщение S3 и т.д.). Если все сигналы имеют значение **FALSE**, то на выходе – пустая строка.

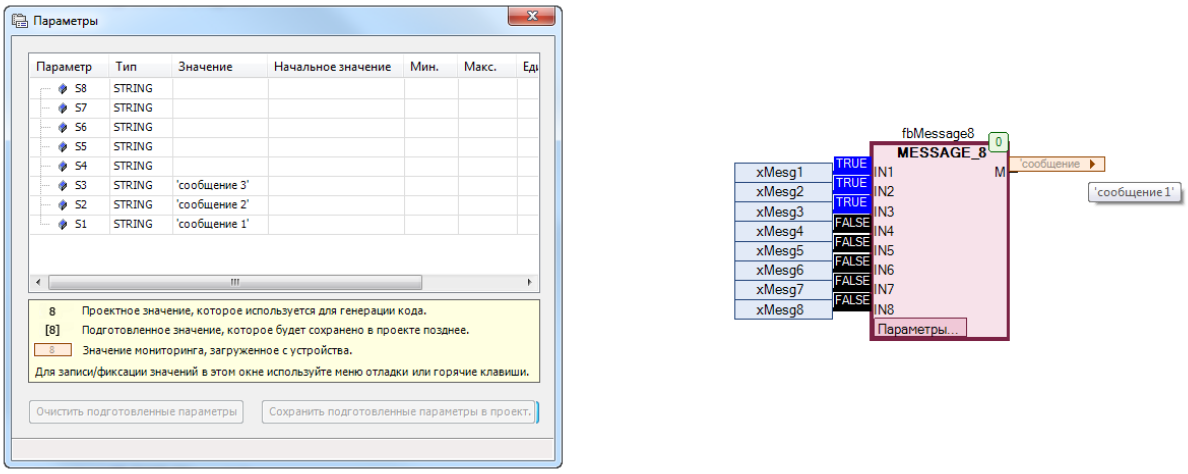


Рис. 13.114. Внешний вид ФБ **MESSAGE_8** на языке CFC

13.58. MIRROR

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	str	STRING	Исходная строка.
Выходы	MIRROR	STRING	Зеркальная строка.

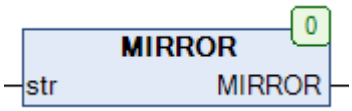


Рис. 13.115. Внешний вид функции **MIRROR** на языке CFC

Функция **MIRROR** конвертирует исходную строку **str** в строку с зеркальным расположением символов.

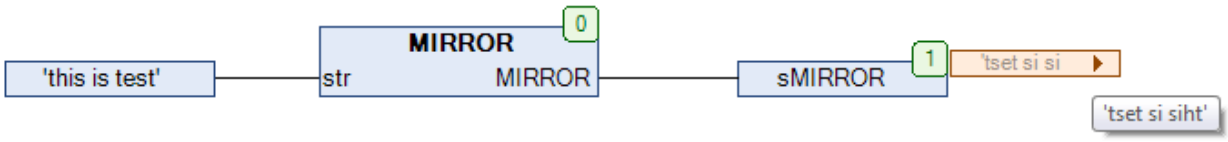


Рис. 13.116. Пример работы с функцией **MIRROR** на языке CFC

13.59. MONTH_TO_STRING

Тип модуля: функция	Переменная	Тип	Описание
Входы	MTH	INT	Номер месяца.
	LANG	INT	Язык приложения.
	LX	INT	Тип названия (полное/сокращенное).
Выходы	MONTH_TO_STRING	STRING(10)	Название месяца.

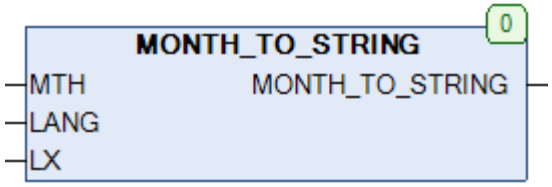


Рис. 13.117. Внешний вид функции **MONTH_TO_STRING** на языке CFC

Функция **MONTH_TO_STRING** возвращает название месяца с номером **MTH**. Вход **LANG** определяет язык приложения (см. глобальную переменную LANGUAGE). Вход **LX** определяет тип названия месяца: **0** – полное (January), **3** – сокращенное (Jan).

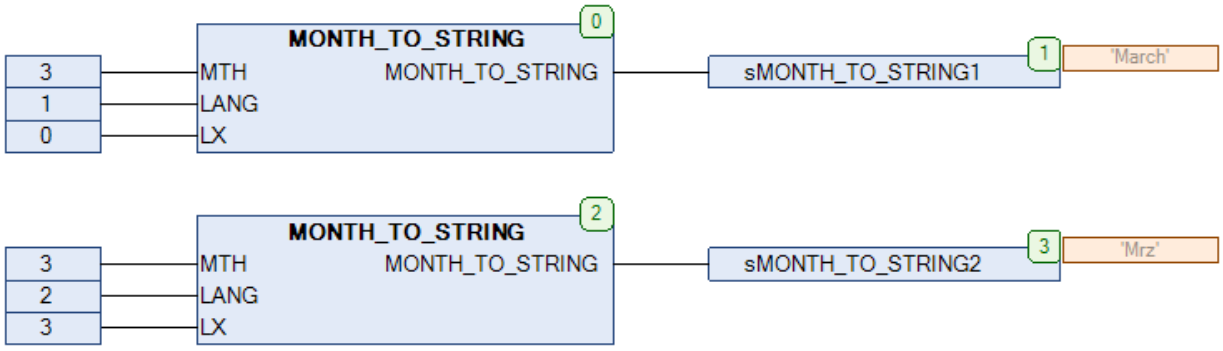


Рис. 13.118. Пример работы с функцией **MONTH_TO_STRING** на языке CFC

13.60. OCT_TO_BYTE

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	OCT	STRING(10)	Восьмеричное значение в символьном виде.
Выходы	OCT_TO_BYTE	BYTE	Восьмеричное значение в целочисленном виде.

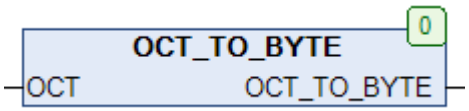


Рис. 13.119. Внешний вид функции **OCT_TO_BYTE** на языке CFC

Функция **OCT_TO_BYTE** конвертирует восьмеричное символьное значение **OCT** в целочисленное восьмеричное значение типа **BYTE**. Учитываются только символы ‘0’ – ‘7’ в пределах допустимого для типа BYTE диапазона ‘0’ – ‘255’.

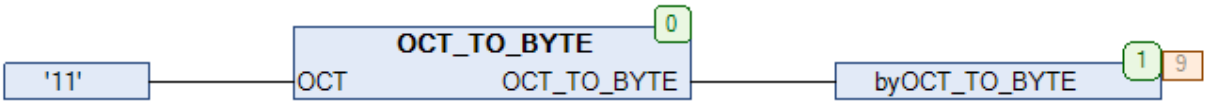


Рис. 13.120. Пример работы с функцией **OCT_TO_BYTE** на языке CFC

13.61. OCT_TO_DWORD

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	OCT	STRING(20)	Восьмеричное значение в символьном виде.
Выходы	OCT_TO_DWORD	DWORD	Восьмеричное значение в целочисленном виде.

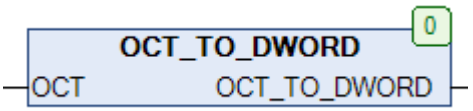


Рис. 13.121. Внешний вид функции **OCT_TO_DWORD** на языке CFC

Функция **OCT_TO_DWORD** конвертирует [восьмеричное](#) символьное значение **OCT** в целочисленное восьмеричное значение типа **DWORD**. Учитываются только символы ‘0’ – ‘7’ в пределах допустимого для типа **DWORD** диапазона ‘0’ – ‘4294967295’.

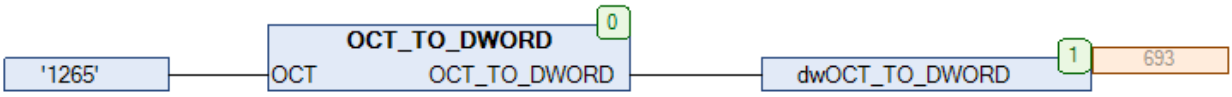


Рис. 13.122. Пример работы с функцией **OCT_TO_DWORD** на языке CFC

13.62. REAL_TO_STRF

Тип модуля: функция	Переменная	Тип	Описание
Входы	IN	REAL	Значение с плавающей точкой в формате REAL.
	N	INT	Кол-во знаков после запятой.
	D	STRING(1)	Символ-разделитель.
Выходы	REAL_TO_STRF	STRING(20)	Значение с плавающей точкой в формате STRING.
Используемые модули	EXP10		

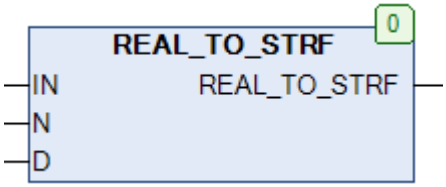


Рис. 13.123. Внешний вид функции **REAL_TO_STRF** на языке CFC

Функция **REAL_TO_STRF** конвертирует переменную **IN** типа **REAL** в строку, содержащую соответствующее значение с плавающей точкой. Вход **N** определяет количество знаков после запятой, до которого будет округлено значение **IN**; он может принимать значения из диапазона **0…7** – это связано с максимальной возможной точностью для типа **REAL**. Вход **D** определяет символ-разделитель для целой и дробной части.

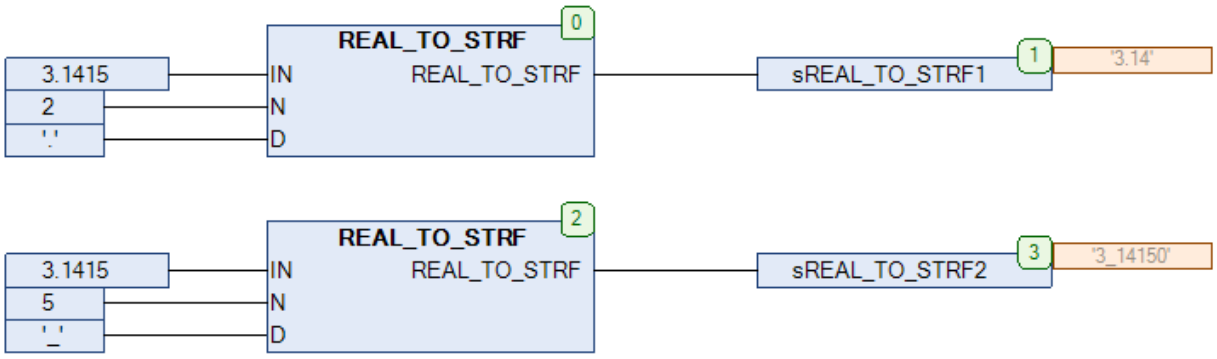


Рис. 13.124. Пример работы с функцией **REAL_TO_STRF** на языке CFC

13.63. REPLACE_ALL

Тип модуля: функция	Переменная	Тип	Описание
Входы	str	STRING	Исходная строка.
	src	STRING	Искомая строка.
	rep	STRING	Замещающая строка.
Выходы	REPLACE_ALL	STRING	Обработанная строка.
Используемые модули	FINDP		

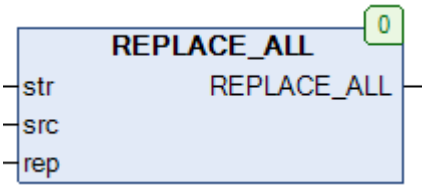


Рис. 13.125. Внешний вид функции **REPLACE_ALL** на языке CFC

Функция **REPLACE_ALL** осуществляет поиск в исходной строке **str** всех текстовых фрагментов **src** и заменяет их на **rep**.

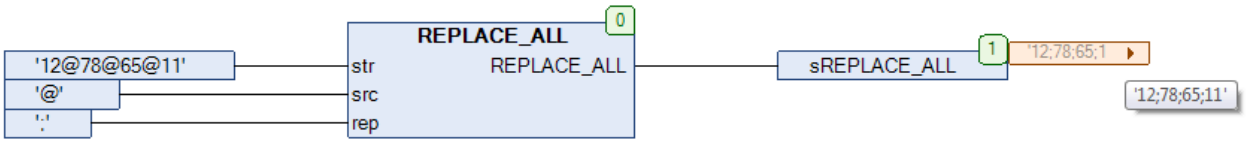


Рис. 13.126. Пример работы с функцией **REPLACE_ALL** на языке CFC

13.64. REPLACE_CHARS

Тип модуля: функция	Переменная	Тип	Описание
Входы	str	STRING	Исходная строка.
	SRC	STRING	Строка с набором искомых символов.
	REP	STRING	Строка с набором замещающих символов.
Выходы	REPLACE_CHARS	STRING	Обработанная строка.

Рис. 13.127. Внешний вид функции **REPLACE_CHARS** на языке CFC

Функция **REPLACE_CHARS** осуществляет поиск в исходной строке **str** всех символов, входящих в строку **SRC**, после чего замещает их символами с идентичной (для строки **SRC**) позицией из строки **REP**.

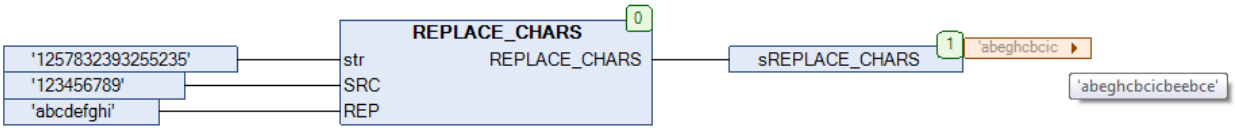


Рис. 13.128. Пример работы с функцией **REPLACE_CHARS** на языке CFC

13.65. REPLACE_UML

Тип модуля: функция	Переменная	Тип	Описание
Входы	str	STRING	Исходная строка.
Выходы	REPLACE_UML	STRING	Обработанная строка.
Используемые модули	TO_UML		

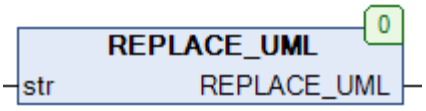


Рис. 13.129. Внешний вид функции **REPLACE_UML** на языке CFC

Функция **REPLACE_UML** заменяет все умляуты в исходной строке **str** на соответствующие сочетания латинских букв:

Умляут	Транслитерация умляута
Ä	Ae
Ö	Oe
Ü	Ue
ä	ae
ö	oe
ü	ue
ß	Ss

Если строка состоит из умляутов в верхнем регистре, но предварительно необходимо перевести ее в нижний регистр с помощью функции LOWERCASE, после чего обработать функцией **REPLACE_UML**.

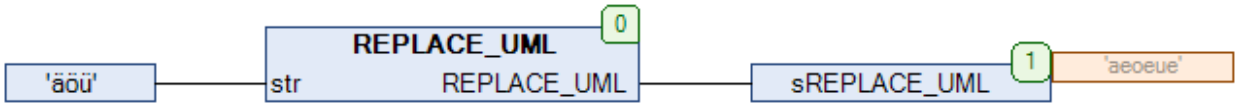


Рис. 13.130. Пример работы с функцией **REPLACE_UML** на языке CFC

13.66. TICKER

Тип модуля: ФБ	Переменная	Тип	Описание
Входы	N	INT	Длина фрагмента сообщения.
	PT	TIME	Время переключения фрагментов.
Выходы	Display	STRING	Фрагмент сообщения.
Входы-выходы	Text	STRING	Сообщение.

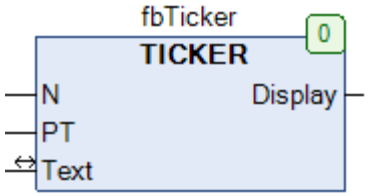


Рис. 13.131. Внешний вид ФБ **TICKER** на языке CFC

Функциональный блок **TICKER** используется для реализации бегущей строки. Фрагмент исходного сообщения **Text** длиной **N** передается на выход **Display**, причем с интервалом **PT** происходит смещение фрагмента на один символ вправо относительно полного текста сообщения. Блок работает корректно только при N < Длина (Text). На рис. 13.132 приведен пример работы с блоком.

Время от начала работы ФБ, с	0	5	10	15	20	25	…
Значение выхода Display	hel	ell	llo	lo	o	hel	…

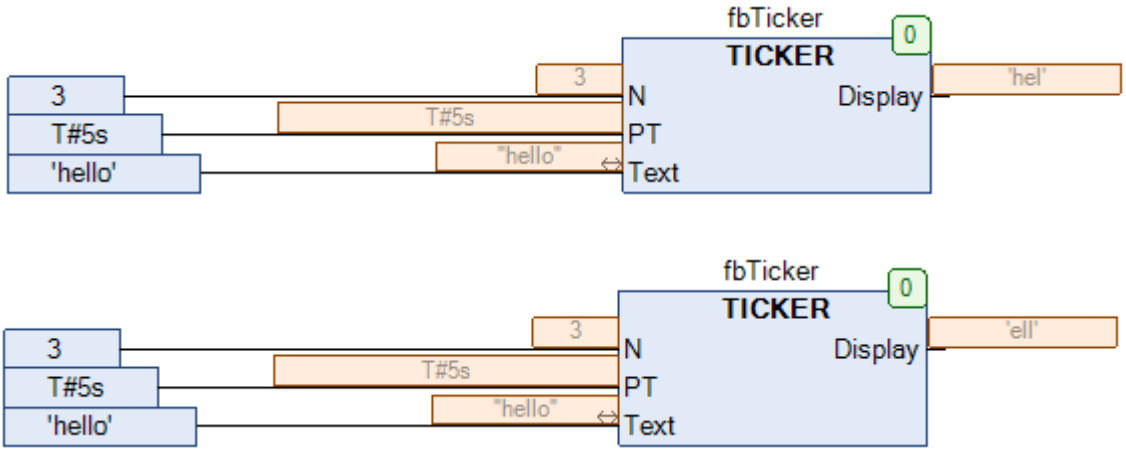


Рис. 13.132. Пример работы с ФБ **TICKER** на языке CFC

13.67. TO_LOWER

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	IN	BYTE	ASCII-код символа верхнего регистра.
Выходы	TO_LOWER	BYTE	ASCII-код символа нижнего регистра.

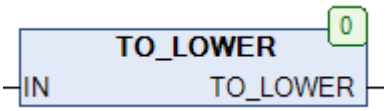


Рис. 13.133. Внешний вид функции **TO_LOWER** на языке CFC

Функция **TO_LOWER** конвертирует ASCII-код символа верхнего регистра в ASCII-код аналогичного символа нижнего регистра. Для работы с символами верхней половины таблицы ASCII глобальная переменная EXTENDED_ASCII должна иметь значение **TRUE**.

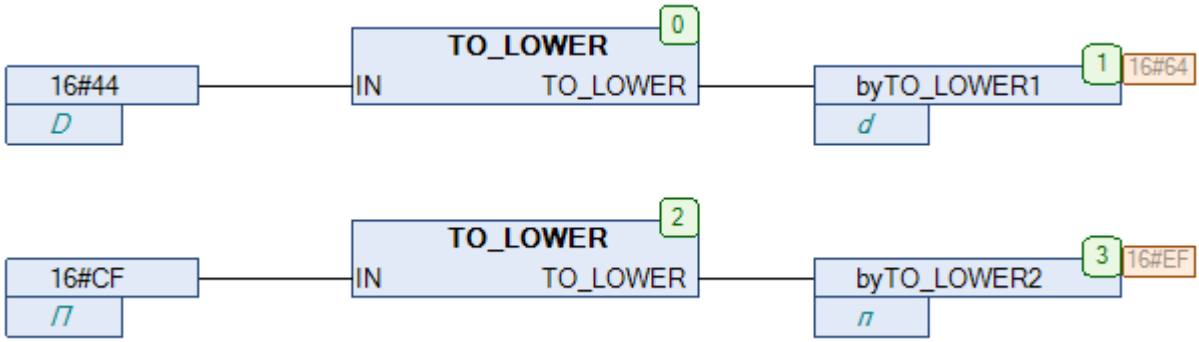


Рис. 13.134. Пример работы с функцией **TO_LOWER** на языке CFC

13.68. TO_UML

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	IN	BYTE	ASCII-код умляута.
Выходы	TO_UML	STRING(2)	Транслитерация умляута.
Используемые модули	CHR_TO_STRING		

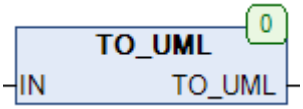


Рис. 13.135. Внешний вид функции **TO_UML** на языке CFC

Функция **TO_UML** конвертирует ASCII-код умляута в строку, содержащую его транслитерацию.

Умляут	**ASCII-код**	**Транслитерация умляута**
Ä	16#C4	Ae
Ö	16#D6	Oe
Ü	16#DC	Ue
ä	16#E4	ae
ö	16#F6	oe
ü	16#FC	ue
ß	16#DF	Ss

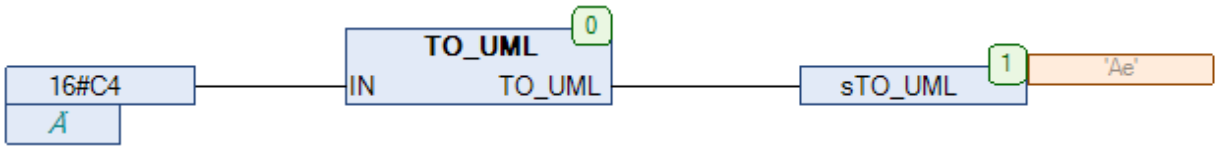


Рис. 13.136. Пример работы с функцией **TO_UML** на языке CFC

13.69. TO_UPPER

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	IN	BYTE	ASCII-код символа нижнего регистра.
Выходы	TO_UPPER	BYTE	ASCII-код символа верхнего регистра.

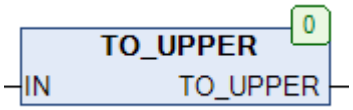


Рис. 13.137. Внешний вид функции **TO_UPPER** на языке CFC

Функция **TO_UPPER** конвертирует ASCII-код символа нижнего регистра в ASCII-код аналогичного символа верхнего регистра. Для работы с символами верхней половины таблицы ASCII глобальная переменная EXTENDED_ASCII должна иметь значение **TRUE**.

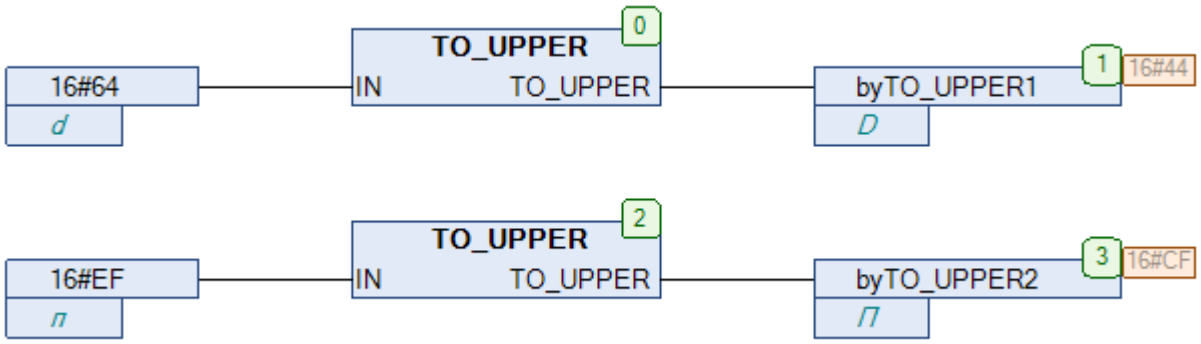


Рис. 13.138. Пример работы с функцией **TO_UPPER** на языке CFC

13.70. TRIM

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	str	STRING	Исходная строка.
Выходы	TRIM	STRING	Строка без пробелов.

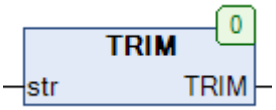


Рис. 13.139. Внешний вид функции **TRIM** на языке CFC

Функция **TRIM** удаляет из строки **str** пробелы и возвращает обработанную строку.

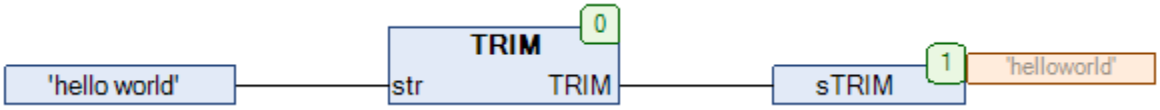


Рис. 13.140. Пример работы с функцией **TRIM** на языке CFC

13.71. TRIM1

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	str	STRING	Исходная строка.
Выходы	TRIM1	STRING	Строка без последовательных пробелов.

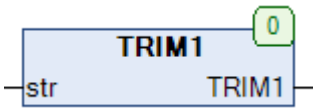


Рис. 13.141. Внешний вид функции **TRIM1** на языке CFC

Функция **TRIM1** заменяет в строке **str** последовательные пробелы на один пробел, а также удаляет все пробелы в начале и конце строки.

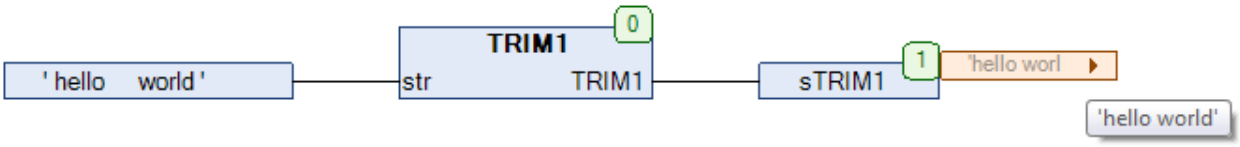


Рис. 13.142. Пример работы с функцией **TRIM1** на языке CFC

13.72. TRIME

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	str	STRING	Исходная строка.
Выходы	TRIME	STRING	Строка без пробелов в начале и конце.

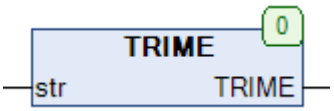


Рис. 13.143. Внешний вид функции **TRIME** на языке CFC

Функция **TRIME** удаляет пробелы в начале и конце строки **str**, после чего возвращает обработанную строку.

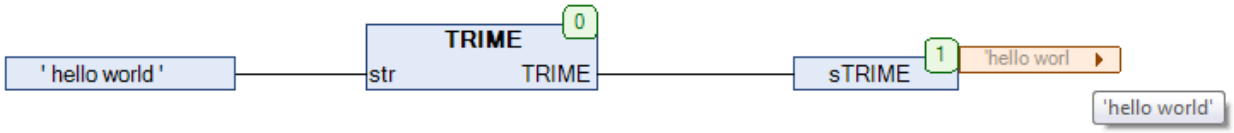


Рис. 13.144. Пример работы с функцией **TRIME** на языке CFC

13.73. UPPERCASE

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	str	STRING	Исходная строка.
Выходы	UPPERCASE	STRING	Строка в верхнем регистре.
Используемые модули	TO_UPPER		

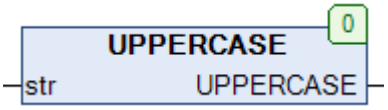


Рис. 13.145. Внешний вид функции **UPPERCASE** на языке CFC

Функция **UPPERCASE** конвертирует исходную строку **str** в соответствующую строку верхнего регистра. Для работы с символами верхней половины таблицы ASCII глобальная переменная EXTENDED_ASCII должна иметь значение **TRUE**.

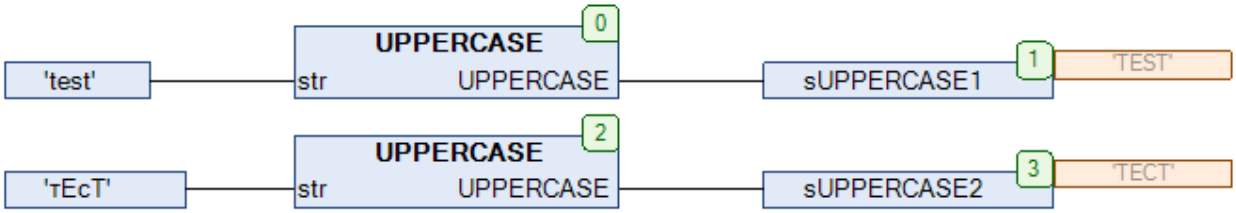


Рис. 13.146. Пример работы с функцией **UPPERCASE** на языке CFC

13.74. WEEKDAY_TO_STRING

Тип модуля: функция	Переменная	Тип	Описание
Входы	WDAY	INT	Номер дня недели.
	LANG	INT	Язык приложения.
	LX	INT	Тип названия (полное/сокращенное).
Выходы	WEEKDAY_TO_STRING	STRING(10)	Название дня недели.

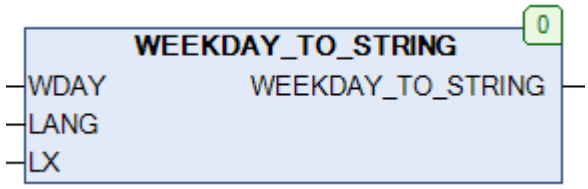


Рис. 13.147. Внешний вид функции **WEEKDAY_TO_STRING** на языке CFC

Функция **WEEKDAY_TO_STRING** возвращает название дня недели с номером **WDAY**. Вход **LANG** определяет язык приложения (см. глобальную переменную LANGUAGE). Вход **LX** определяет тип названия месяца: **0** – полное (Monday), **2** – сокращенное (Mo).

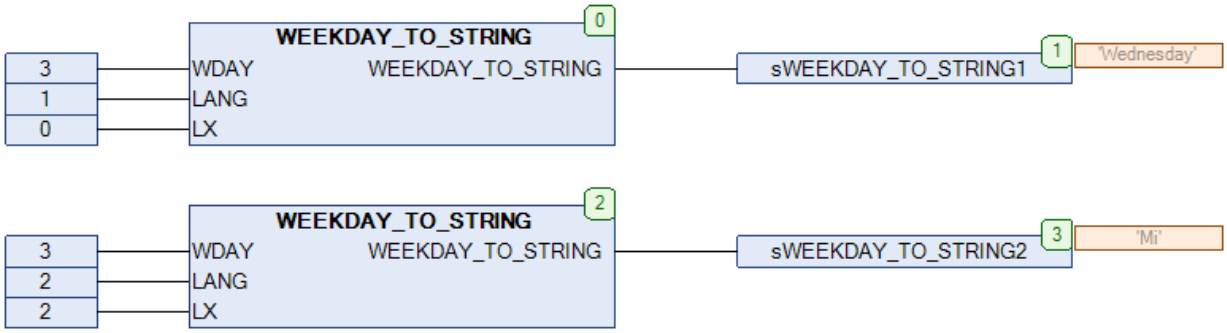


Рис. 13.148. Пример работы с функцией **WEEKDAY_TO_STRING** на языке CFC

14. Модули памяти

14.1. FIFO_16

Тип модуля: ФБ	**Переменная**	**Тип**	**Описание**
Входы	Din	DWORD	Данные, записываемые в буфер.
	E	BOOL	Сигнал управления блоком (вкл./откл.).
	RD	BOOL	Сигнал чтения данных из буфера.
	WD	BOOL	Сигнал записи данных в буфер.
	RST	BOOL	Сигнал очистки буфера.
Выходы	Dout	DWORD	Данные, считанные из буфера.
	EMPTY	BOOL	Флаг «буфер пуст».
	FULL	BOOL	Флаг «буфер полон».
Используемые модули	INC1		

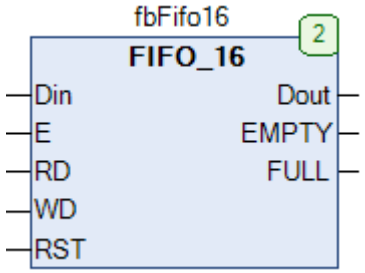


Рис. 14.1. Внешний вид ФБ **FIFO_16** на языке CFC

Функциональный блок **FIFO_16** представляет собой буфер размером 16 значений типа **DWORD**, организованный по принципу **First In – First Out** (FIFO). Если вход **E=TRUE,** то блок находится в работе. Если **WD=TRUE**, то каждый цикл ПЛК значение входа **Din** записывается в буфер. Если **RD=TRUE**, то каждый цикл ПЛК значения из буфера подаются на выход **Dout** (в том порядке, в котором они были записаны в буфер). Чтобы записать/считать один элемент, необходимо подать на вход **RD/WD** единичный импульс. Выход **EMPTY** принимает значение **TRUE**, если буфер пуст (например, при попытке прочитать четвертое значение из буфера, в который было записано только 3 значения, этот флаг станет активным). Выход **FULL** принимает значение **TRUE**, если буфер заполнен. По переднему фронту на входе **RST** происходит очистка буфера.

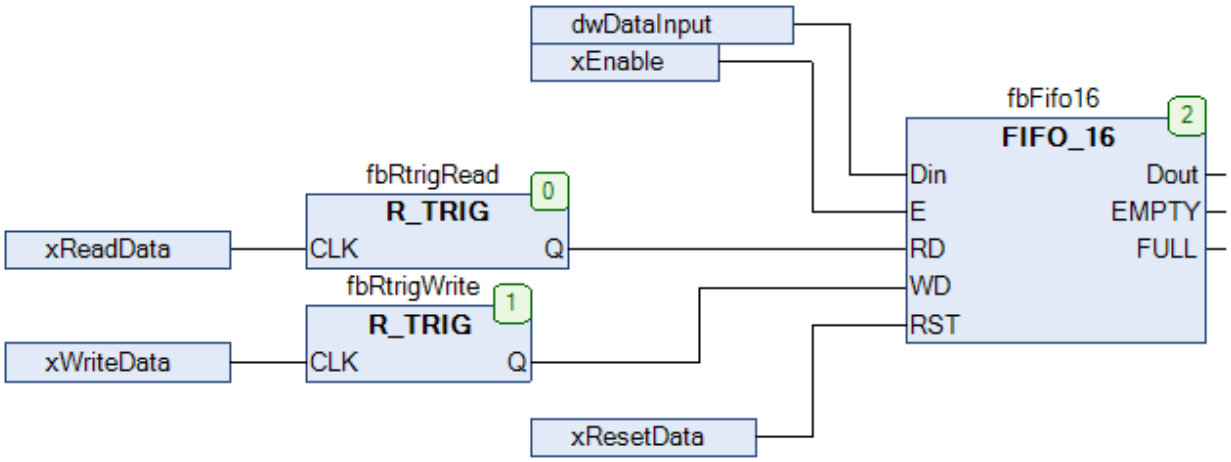


Рис. 14.2. Пример работы с ФБ **FIFO_16** на языке CFC

14.2. FIFO_32

Тип модуля: ФБ	**Переменная**	**Тип**	**Описание**
Входы	Din	DWORD	Данные, записываемые в буфер.
	E	BOOL	Сигнал управления блоком (вкл./откл.).
	RD	BOOL	Сигнал чтения данных из буфера.
	WD	BOOL	Сигнал записи данных в буфер.
	RST	BOOL	Сигнал очистки буфера.
Выходы	Dout	DWORD	Данные, считанные из буфера.
	EMPTY	BOOL	Флаг «буфер пуст».
	FULL	BOOL	Флаг «буфер полон».
Используемые модули	INC1		

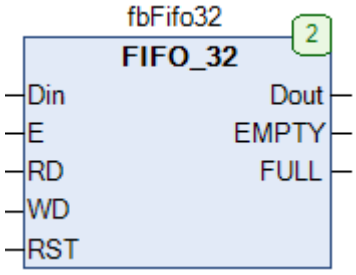


Рис. 14.3. Внешний вид ФБ **FIFO_32** на языке CFC

Функциональный блок **FIFO_32** представляет собой буфер размером 32 значений типа **DWORD**. Принцип работы полностью соответствует ФБ FIFO_16. Можно создать буфер любого размера, изменив константу **n** в разделе объявления переменных блока.

14.3. STACK_16

Тип модуля: ФБ	Переменная	Тип	Описание
Входы	Din	DWORD	Данные, записываемые в буфер.
	E	BOOL	Сигнал управления блоком (вкл./откл.).
	RD	BOOL	Сигнал чтения данных из буфера.
	WD	BOOL	Сигнал записи данных в буфер.
	RST	BOOL	Сигнал очистки буфера.
Выходы	Dout	DWORD	Данные, считанные из буфера.
	EMPTY	BOOL	Флаг «буфер пуст».
	FULL	BOOL	Флаг «буфер полон».

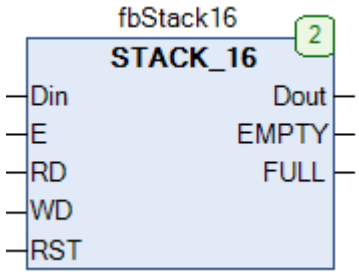


Рис. 14.4. Внешний вид ФБ **STACK_16** на языке CFC

Функциональный блок **STACK_16** представляет собой буфер размером 16 значений типа **DWORD**, организованный по принципу **Last In – First Out** (LIFO). Если вход **E=TRUE**, то блок находится в работе. Если **WD=TRUE**, то каждый цикл ПЛК значение входа Din записывается в буфер. Если **RD=TRUE**, то каждый цикл ПЛК значения из буфера подаются на выход **Dout** (в порядке, обратном порядку их записи в буфер). Чтобы записать/считать один элемент, необходимо подать на вход **RD/WD** единичный импульс. Выход **EMPTY** принимает значение **TRUE**, если буфер пуст (например, при попытке прочитать четвертое значение из буфера, в который было записано только 3 значения, этот флаг станет активным). Выход **FULL** принимает значение **TRUE**, если буфер заполнен. По переднему фронту на входе **RST** происходит очистка буфера.

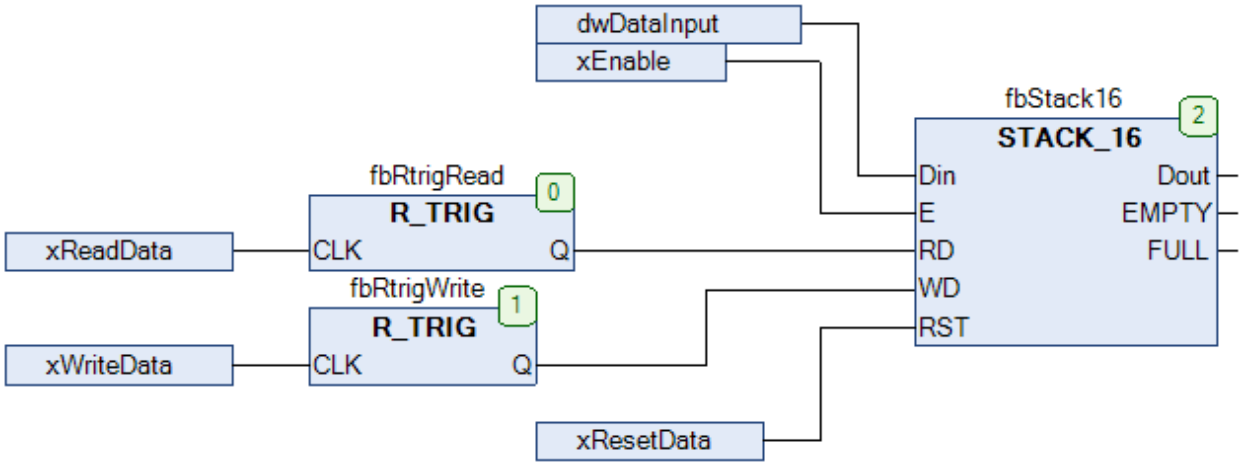


Рис. 14.5. Пример работы с **ФБ STACK_16** на языке CFC

14.4. STACK_32

Тип модуля: ФБ	**Переменная**	**Тип**	**Описание**
Входы	Din	DWORD	Данные, записываемые в буфер.
	E	BOOL	Сигнал управления блоком (вкл./откл.).
	RD	BOOL	Сигнал чтения данных из буфера.
	WD	BOOL	Сигнал записи данных в буфер.
	RST	BOOL	Сигнал очистки буфера.
Выходы	Dout	DWORD	Данные, считанные из буфера.
	EMPTY	BOOL	Флаг «буфер пуст».
	FULL	BOOL	Флаг «буфер полон».

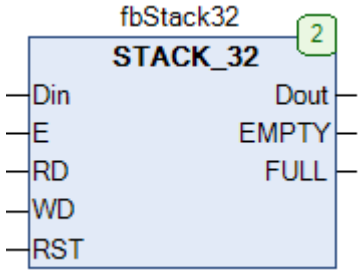


Рис. 14.6. Внешний вид ФБ **STACK_32** на языке CFC

Функциональный блок **STACK_32** представляет собой буфер размером 32 значений типа **DWORD**. Принцип работы полностью соответствует ФБ STACK_16. Можно создать буфер любого размера, изменив константу **n** в разделе объявления переменных блока.

15. Генераторы импульсов

15.1. A_TRIG

Тип модуля: ФБ	**Переменная**	**Тип**	**Описание**
Входы	IN	REAL	Контролируемое значение.
	RES	REAL	Допустимое изменение.
Выходы	Q	BOOL	Импульс превышения допустимого значения.
	D	REAL	Разность между текущим и предыдущим недопустимым значением.

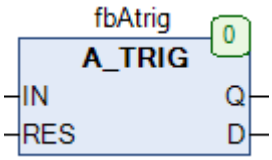


Рис. 15.1. Внешний вид ФБ **A_TRIG** на языке CFC

Функциональный блок **A_TRIG** контролирует входное значение **IN** и при изменении его относительно последнего сохраненного значения на величину, превышающую **RES**, генерирует единичный импульс на выходе **Q** и сохраняет текущее значение **IN**. Разность между последним сохраненным значением **IN** и текущим значением **IN** подается на выход **D**. В момент импульса на выходе **Q** значение **D** обнуляется.

Пример работы блока:

Номер цикла	**IN**	**RES**	**Q**	**D**
1	**0**	5	FALSE	0
2	2		FALSE	2 (2-0)
3	**10**		TRUE (импульс на цикл) [(10-2)>5]	0
4	7		FALSE	-3 (7-10)
5	**4**		TRUE (импульс на цикл) [(10-4)>5]	0
6	0		FALSE	-4 (0-4)

15.2. B_TRIG

Тип модуля: ФБ	**Переменная**	**Тип**	**Описание**
Входы	CLK	BOOL	Контролируемый сигнал.
Выходы	Q	BOOL	Выход триггера.

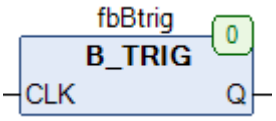


Рис. 15.2. Внешний вид **ФБ B_TRIG** на языке CFC

Функциональный блок **B_TRIG** генерирует единичный импульс на выходе **Q** при изменении входного сигнала **CLK**. В отличие от блоков **R_TRIG** и **F_TRIG** из библиотеки Standard, которые детектируют только импульсы переднего/заднего фронта соответственно, данный блок детектирует импульсы обоих фронтов.

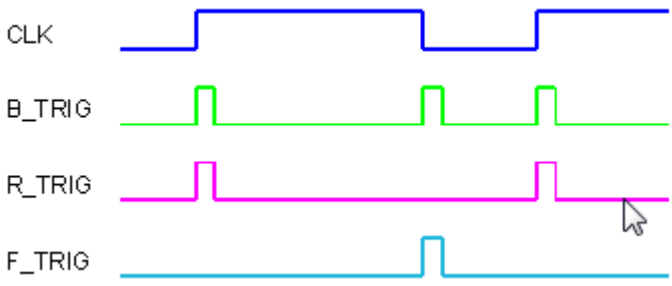


Рис. 15.3. Трассировка работы ФБ **B_TRIG**

15.3. CLICK_CNT

Тип модуля: ФБ	**Переменная**	**Тип**	**Описание**
Входы	IN	BOOL	Контролируемый сигнал.
	N	INT	Ожидаемое число импульсов.
	TC	TIME	Период времени.
Выходы	Q	BOOL	Выход счетчика.

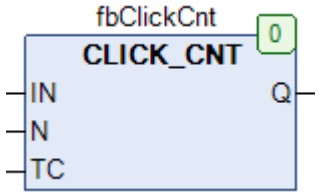


Рис. 15.4. Внешний вид ФБ **CLICK_CNT** на языке CFC

Функциональный блок **CLICK_INT** генерирует единичный импульс на выходе **Q**, если логический сигнал **IN** в течение заданного времени **TC** получает ровно **N** импульсов по заднему фронту. Генерация единичного импульса на выходе осуществляется по истечении времени **TC**. Работа блока (отсчет времени) начинается по импульсу переднего фронта на входе **IN**. Если **N=0**, то единичный импульс на выходе генерируется в том случае, если входной логический сигнал **IN** в течение времени **TC** не получает импульсов по заднему фронту.

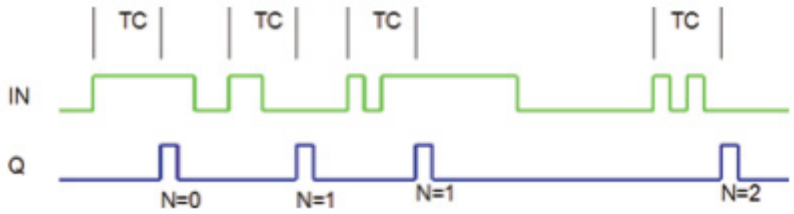


Рис. 15.5. Трассировка работы ФБ **CLICK_CNT**

15.4. CLICK_DEC

Тип модуля: ФБ	Переменная	Тип	Описание
Входы	IN	BOOL	Контролируемый сигнал.
	TC	TIME	Период.
Выходы	Q0	BOOL	Флаг «импульсов не было».
	Q1	BOOL	Флаг «был один импульс».
	Q2	BOOL	Флаг «было два импульса».
	Q3	BOOL	Флаг «было три импульса».

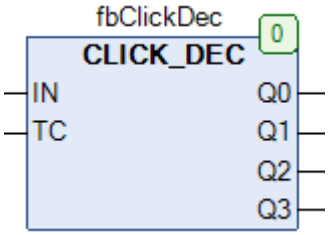


Рис. 15.6. Внешний вид ФБ **CLICK_DEC** на языке CFC

Функциональный блок **CLICK_DEC** генерирует единичный импульс на одном из выходов блока, если логический сигнал **IN** в течение заданного времени **TC** получает соответствующее количество импульсов по заднему фронту (см. таблицу). Генерация единичного импульса на выходе осуществляется по истечении времени **TC.** Работа блока (отсчет времени) начинается по импульсу переднего фронта на входе **IN**.

Число импульсов по заднему фронту за время TC	Единичный импульс на выходе:
0	Q0
1	Q1
2	Q2
3	Q3
>3	-

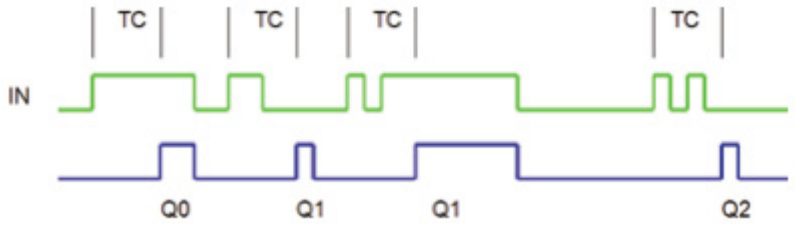


Рис. 15.7. Трассировка работы ФБ **CLICK_DEC**

15.5. CLK_DIV

Тип модуля: ФБ	**Переменная**	**Тип**	**Описание**
Входы	clk	BOOL	Исходный сигнал.
	rst	BOOL	Сброс выходов.
Выходы	Q0…Q7	BOOL	Модулированный сигнал с частотой $\frac{1}{2}, \frac{1}{4} \ldots \frac{1}{256}$ от исходного.

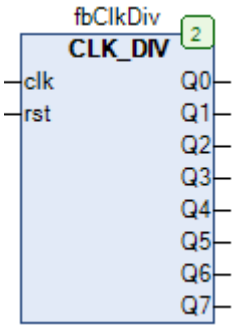


Рис. 15.8. Внешний вид ФБ **CLK_DIV** на языке CFC

Функциональный блок **CLK_DIV** модулирует единичные импульсы, поступающие на вход **clk**, уменьшая их частоту в $\frac{1}{2}, \frac{1}{4} \ldots \frac{1}{256}$ раз и подает обработанные сигналы на выходы **Q0…Q7**. Вход **rst** используется для обнуления выходов блока. Обратите внимание, что на вход **clk** должны подаваться только единичные импульсы; если исходный сигнал не является таковым, то можно обработать его с помощью ФБ TP_X.

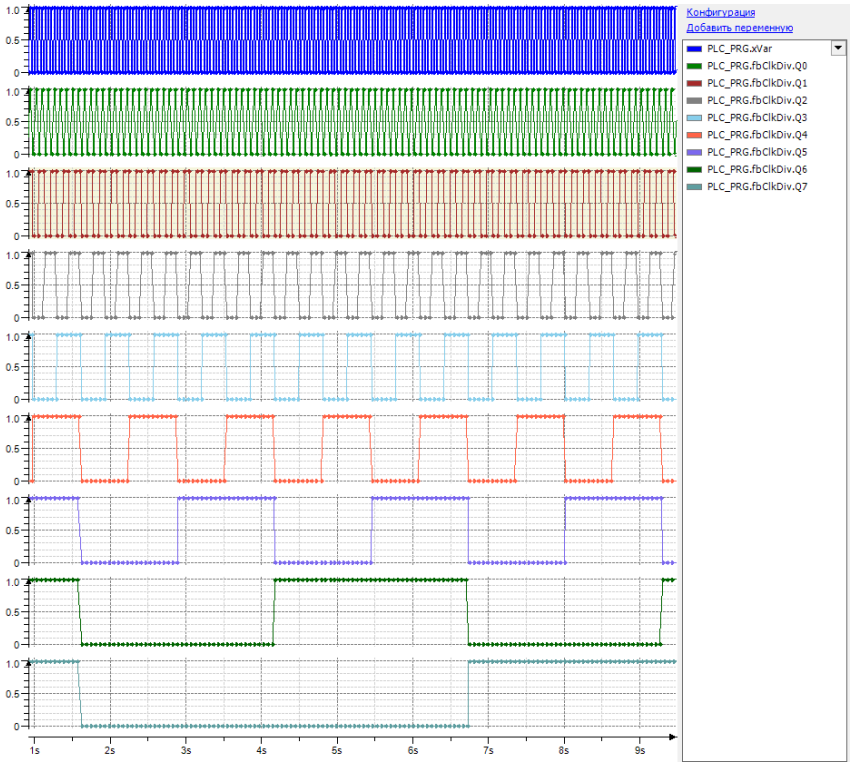


Рис. 15.9. Трассировка работы ФБ **CLK_DIV**

15.6. CLK_N

Тип модуля: ФБ	**Переменная**	**Тип**	**Описание**
Входы	N	INT	Период между импульсами.
Выходы	Q	BOOL	Выходной сигнал.
Используемые модули	T_PLC_MS		

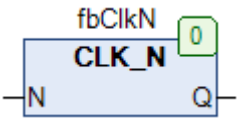


Рис. 15.10. Внешний вид ФБ **CLK_N** на языке CFC

Функциональный блок **CLK_N** генерирует единичные импульсы с периодом 2^N мс. На рис. 15.10 приведена трассировка работы блока в случае:

- N=3 (период между импульсами – 8 мс);
- N=4 (период между импульсами – 16 мс);
- N=5 (период между импульсами – 32 мс);
- N=6 (период между импульсами – 64 мс).

Время цикла ПЛК – 1 мс. По очевидным причинам период между импульсами не может быть меньше реального цикла ПЛК.

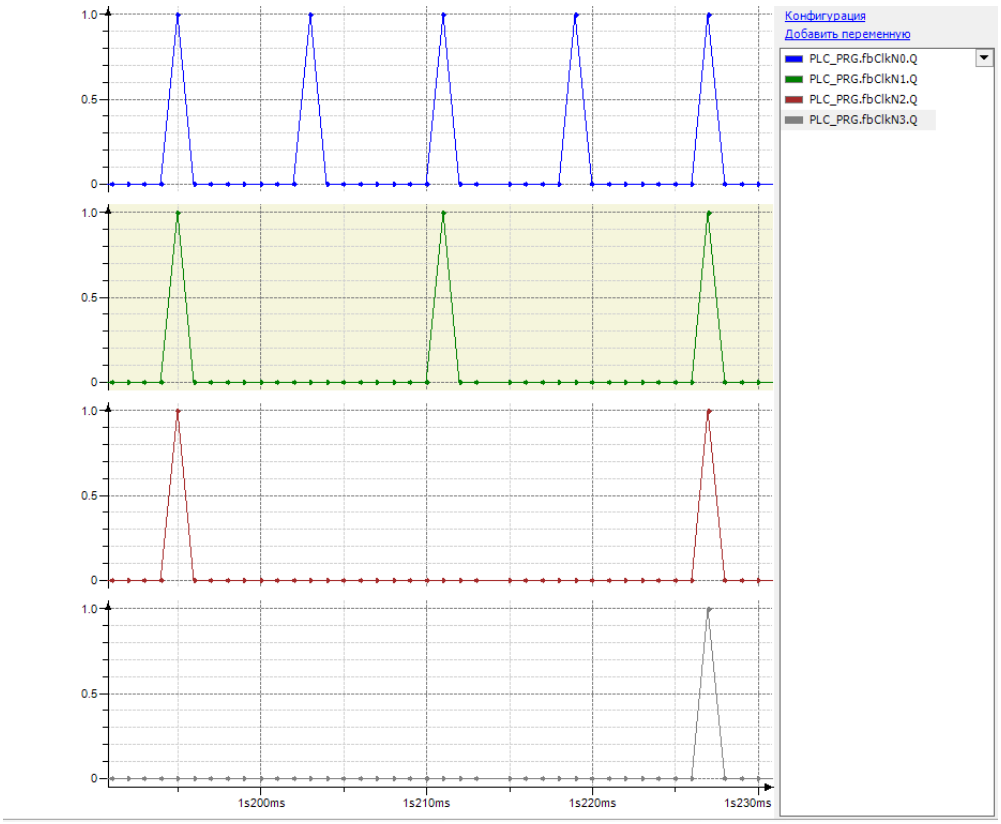


Рис. 15.11. Трассировка работы ФБ **CLK_N**

15.7. CLK_PRG

Тип модуля: ФБ	**Переменная**	**Тип**	**Описание**
Входы	PT	TIME	Период между импульсами.
Выходы	Q	BOOL	Выходной сигнал.
Используемые модули	T_PLC_MS		

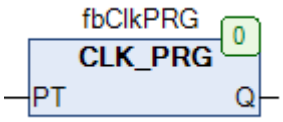


Рис. 15.12. Внешний вид ФБ **CLK_PRG** на языке CFC

Функциональный блок **CLK_PRG** генерирует единичные импульсы с периодом **PT**. По очевидным причинам период между импульсами не может быть меньше реального цикла ПЛК.

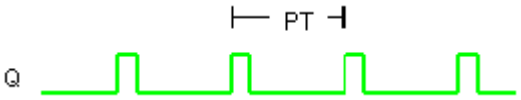


Рис. 15.13. Трассировка работы ФБ **CLK_PRG**

15.8. CLK_PULSE

Тип модуля: ФБ	Переменная	Тип	Описание
Входы	PT	TIME	Период между импульсами.
	N	INT	Кол-во импульсов за рабочий цикл.
	rst	BOOL	Сигнал запуска блока.
Выходы	Q	BOOL	Выходной сигнал.
	cnt	INT	Кол-во сгенерированных импульсов.
	run	BOOL	Флаг «блок в работе».
Используемые модули	T_PLC_MS		

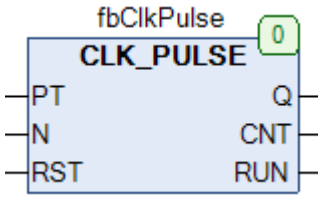


Рис. 15.14. Внешний вид ФБ **CLK_PULSE** на языке CFC

Функциональный блок **CLK_PULSE** генерирует **N** единичных импульсов с заданным периодом **PT**. Импульс по переднему фронту на входе **rst** запускает блок в работу. На выходе **cnt** отображается текущее число сгенерированных импульсов. Выход **run** имеет значение **TRUE**, пока работа блока не завершена. После завершение работы блока выход **run** принимает значение **FALSE**. По очевидным причинам период между импульсами не может быть меньше реального цикла ПЛК. Ниже приведена трассировка ФБ для **PT=T#1s** и **N=5**:

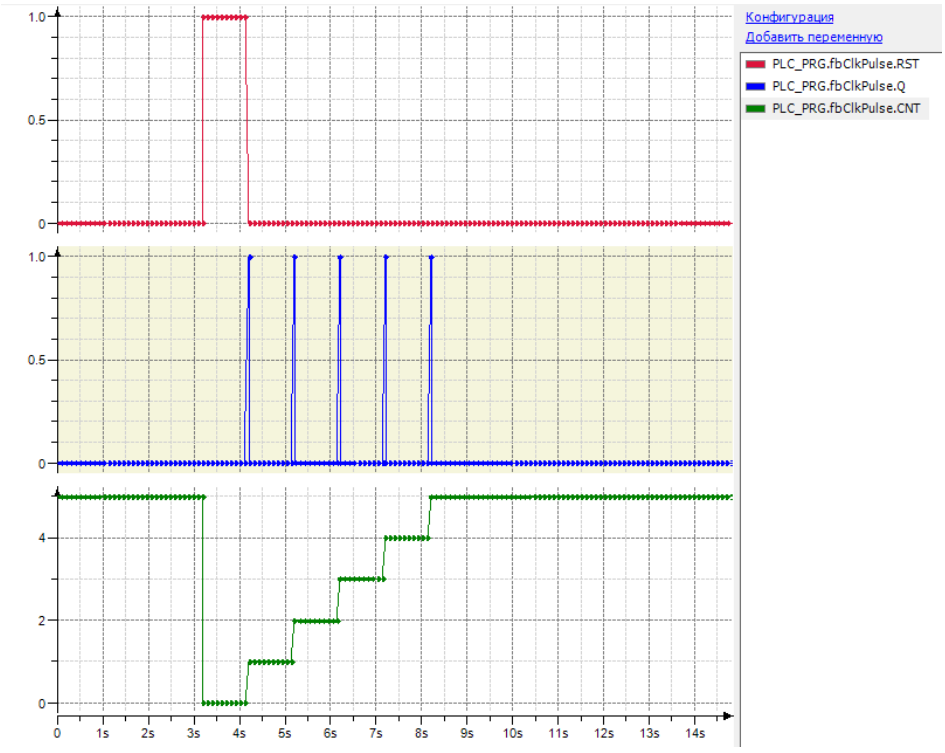


Рис. 15.15. Трассировка работы ФБ **CLK_PULSE**

15.9. CYCLE_4

Тип модуля: ФБ	**Переменная**	**Тип**	**Описание**
Входы	E	BOOL	Вход управления блоком.
	T0	TIME	Время в состоянии 0.
	T1	TIME	Время в состоянии 1.
	T2	TIME	Время в состоянии 2.
	T3	TIME	Время в состоянии 3.
	S0	BOOL	Режим работы блока.
	SX	INT	Номер состояния сброса.
	SL	BOOL	Сброс в состояние SX.
Выходы	STATE	INT	Номер текущего состояния.
Используемые модули	T_PLC_MS		

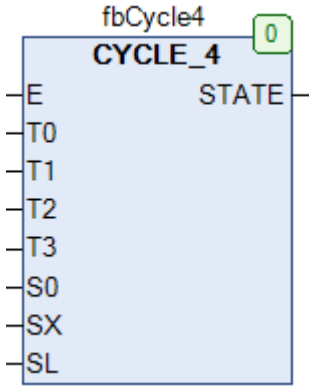


Рис. 15.16. Внешний вид ФБ **CYCLE_4** на языке CFC

Функциональный блок **CYCLE_4** является переключателем для четырех состояний (0-3). Если вход **E=TRUE**, то блок находится в работе, и на выход **STATE** подаются номера состояний (0-3), причем время, проводимое в каждом из состояний, задается переменными **T0…T3**. Вход **S0** определяет режим работы блока:

- Если **S0=TRUE**, то переключение состояний происходит циклически, пока **E=TRUE**.
- Если **S0=FALSE**, то происходит однократное переключение всех состояний, и по достижению состояния 3 блок прекращает работу.

По переднему фронту на входе **SL** происходит переход в состояние с номером **SX**, после чего автоматически начинается очередная смена состояний.

Если **E=FALSE,** то блок прекращает работу и переходит в состояние 0.

На рис. 15.17 приведена трассировка работы блока для S0=TRUE, T0=1 мс, T1=2 мс, T3=4мс, T3=2 мс:

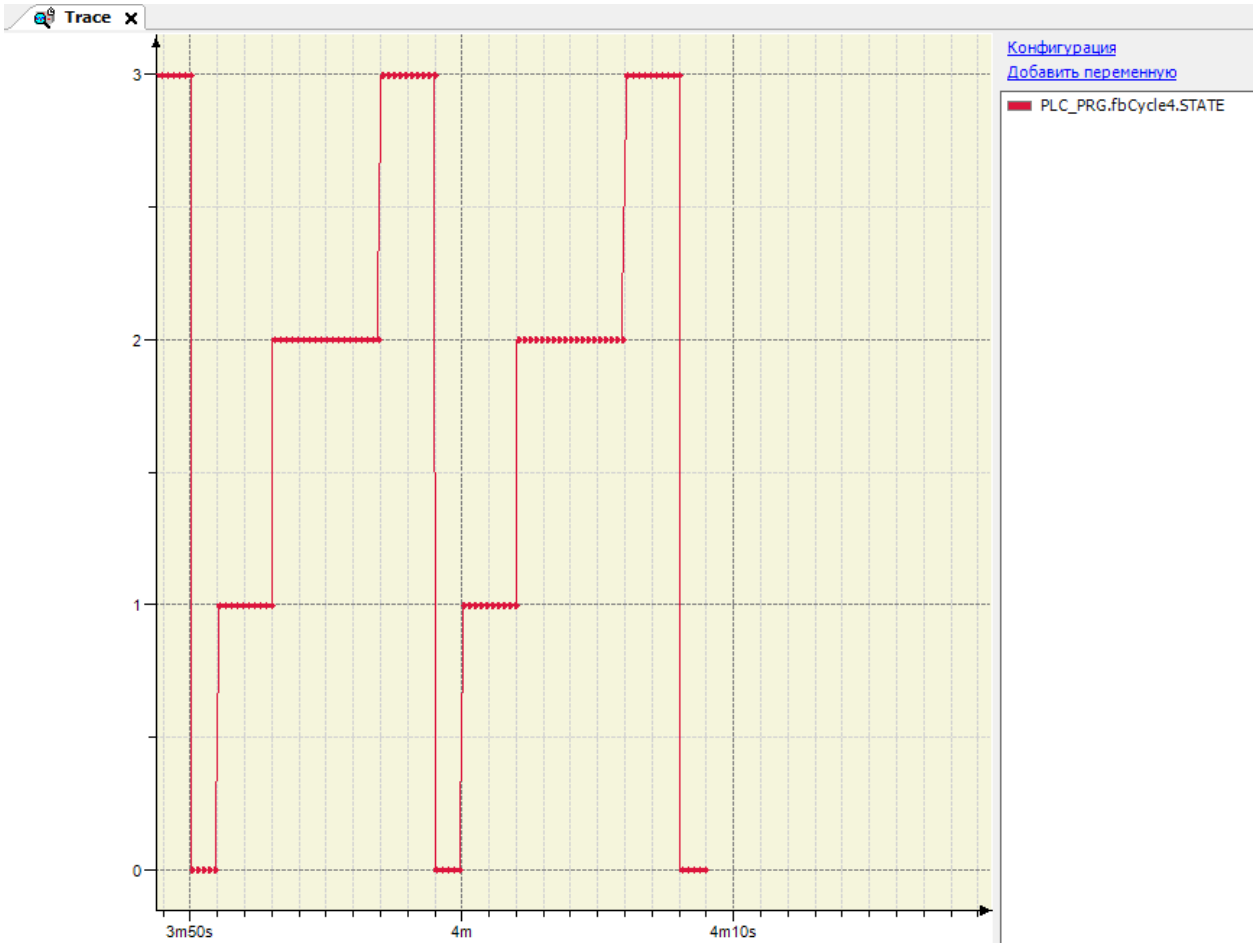


Рис. 15.17. Трассировка работы ФБ **CYCLE_4**

15.10. D_TRIG

Тип модуля: ФБ	**Переменная**	**Тип**	**Описание**
Входы	PT	DWORD	Контролируемое значение.
Выходы	Q	BOOL	Флаг «значение изменилось».
	X	DWORD	Разность между текущим и предыдущим значением.

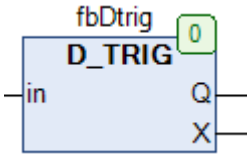


Рис. 15.18. Внешний вид ФБ **D_TRIG** на языке CFC

Функциональный блок **CLK_PRG** генерирует единичный импульс при изменении входного значения **in**. На выход **X** подается разность между значениями **in** текущего и предыдущего цикла. На рис. 15.19 приведен пример работы с блоком. Начальное значение **in** – **0**, далее оно изменяется до **12**, после чего – до **6**.

Поскольку выход **X** имеет тип **DWORD**, а значение **in** может уменьшиться по сравнению с предыдущим циклом (т.е. разность будет отрицательной), то необходимо преобразовать его с помощью стандартного оператора **DWORD_TO_DINT**.

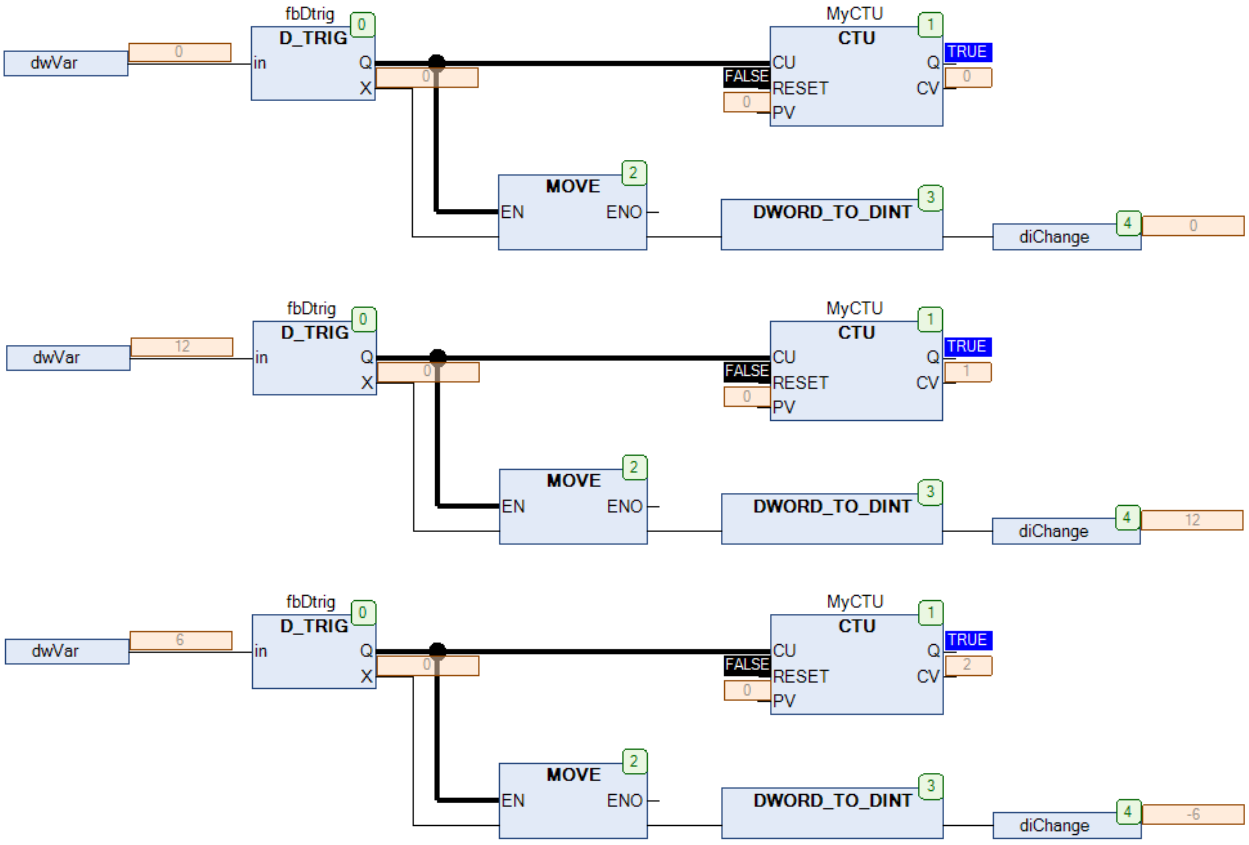


Рис. 15.19. Пример работы с ФБ **D_TRIG** на языке CFC

15.11. GEN_BIT

Тип модуля: ФБ	Переменная	Тип	Описание
Входы	in0…in3	DWORD	Генерируемые последовательности бит.
	clk	BOOL	Вход управления блоком.
	steps	INT	Кол-во используемых бит последовательностей.
	rep	INT	Число повторений последовательностей.
	rst	BOOL	Вход сброса блока.
Выходы	Q0…Q3	BOOL	Выходы генератора.
	cnt	INT	Номер текущего бита последовательностей.
	run	BOOL	Флаг «блок в работе».

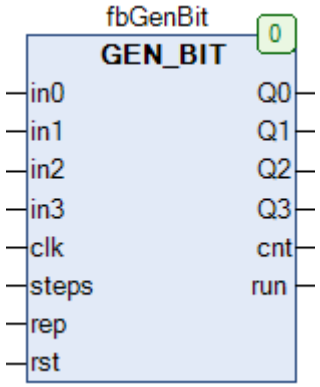


Рис. 15.20. Внешний вид ФБ **GEN_BIT** на языке CFC

Функциональный блок **GEN_BIT** используется для генерации битовых последовательностей на выходах **Q0…Q3** по заданным паттернам **in0…in3**. Если вход **clk=TRUE**, то блок находится в работе, и на выходы **Q0…Q3** каждый цикл ПЛК последовательно подаются значения бит входов **in0…in3** (от младшего бита **DWORD** к старшему). Вход **steps** определяет число используемых бит из паттернов **in0…in3**. Вход **rep** определяет число повторов битовых последовательностей на выходе; если **rep=0**, то последовательность повторяется циклически. На выход **cnt** поступает номер битов паттернов, в данный момент поданных на выход. Выход **run** имеет значение **TRUE**, пока работа блока не завершена. Импульс по переднему фронту на входе **rst** обнуляет все выходы блока.

На рис. 15.21 приведен пример работы с блоком и его трассировка для ПЛК с циклом= 1 с.

В t=11 с на вход **xEnable** было подано значение **TRUE**, после чего блок два раза подряд (**rep=2**) сгенерировал последовательность размером 8 бит (**steps=8**), созданную на основе паттерна **in0** (2#0000_1111).

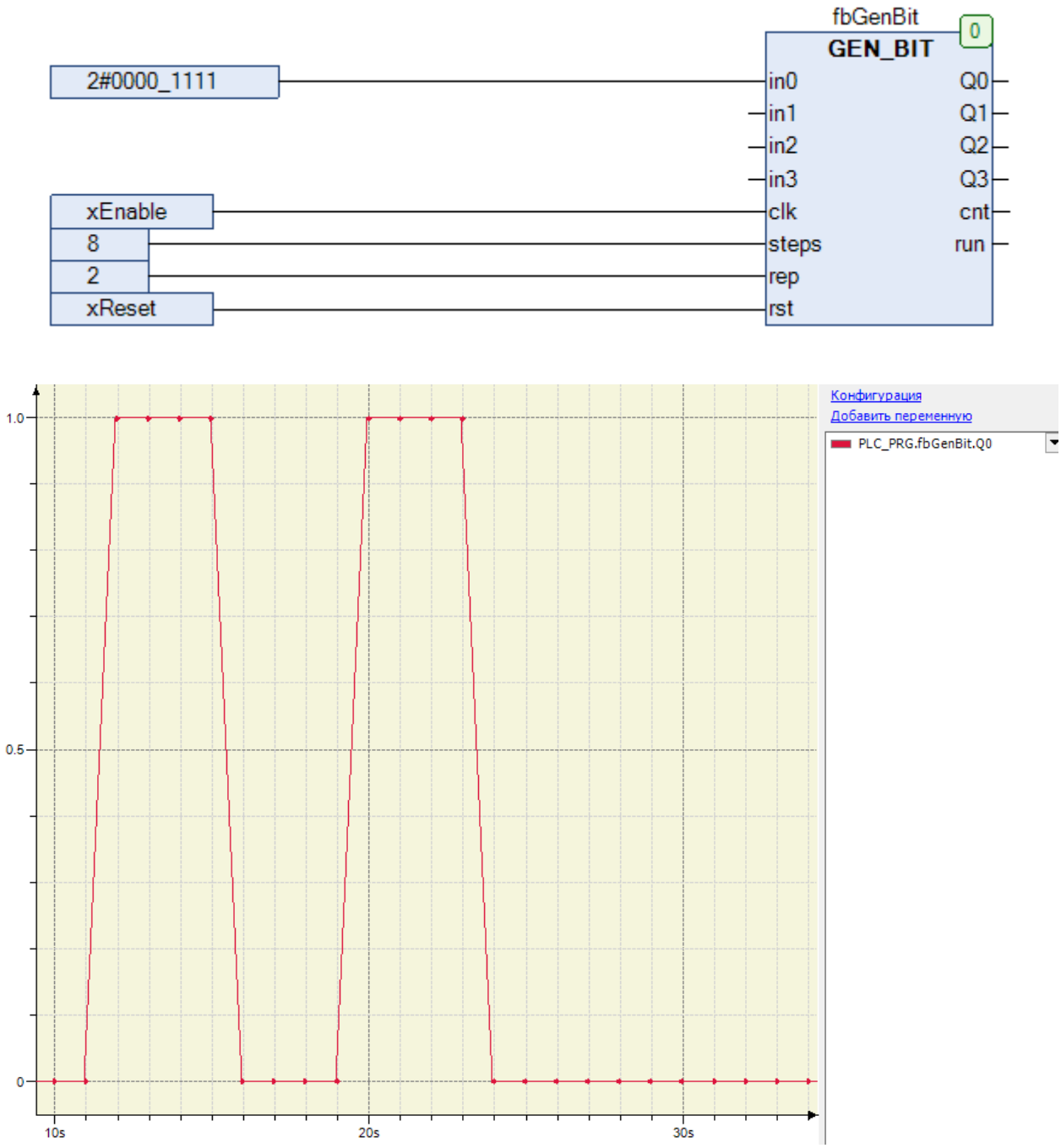


Рис. 15.21. Пример работы с ФБ **GEN_BIT** на языке CFC

15.12. GEN_SQ

Тип модуля: ФБ	**Переменная**	**Тип**	**Описание**
Входы	PT	TIME	Период между импульсами.
Выходы	Q	BOOL	Выход генератора.
Используемые модули	T_PLC_MS		

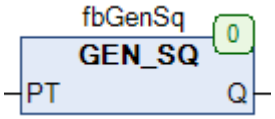


Рис. 15.22. Внешний вид ФБ **GEN_SQ** на языке CFC

Функциональный блок **GEN_SQ** с заданным периодом **PT** генерирует импульсы длиной **0.5 PT** (соответственно, пауза между импульсами также составляет 0.5 PT).

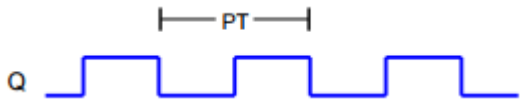


Рис. 15.23. Трассировка работы ФБ **GEN_SQ**

15.13. SCHEDULER

Тип модуля: ФБ	**Переменная**	**Тип**	**Описание**
Входы	E0…E3	BOOL	Вход управления каналом генератора.
Выходы	Q0…Q3	BOOL	Выход канала генератора.
Параметры	T0…T3	TIME	Время между импульсами.
Используемые модули	T_PLC_MS		

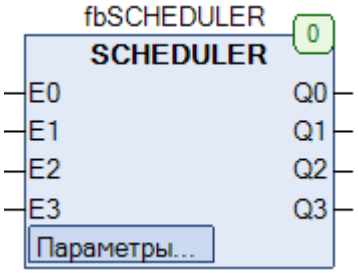


Рис. 15.24. Внешний вид ФБ **SCHEDULER** на языке CFC

Функциональный блок **SCHEDULER** представляй собой четырехканальный генератор единичных импульсов. Если один из входов **E** принимает значение **TRUE**, то на соответствующем выходе **Q** начинают генерироваться единичные импульсы с периодом **T**. В каждом цикле ПЛК импульс генерируется только в одном из каналов.

На рис. 15.25 приведена трассировка работы блока для T0=1 мс, T1=2 мс, T2=3 мс, T3=4 мс.

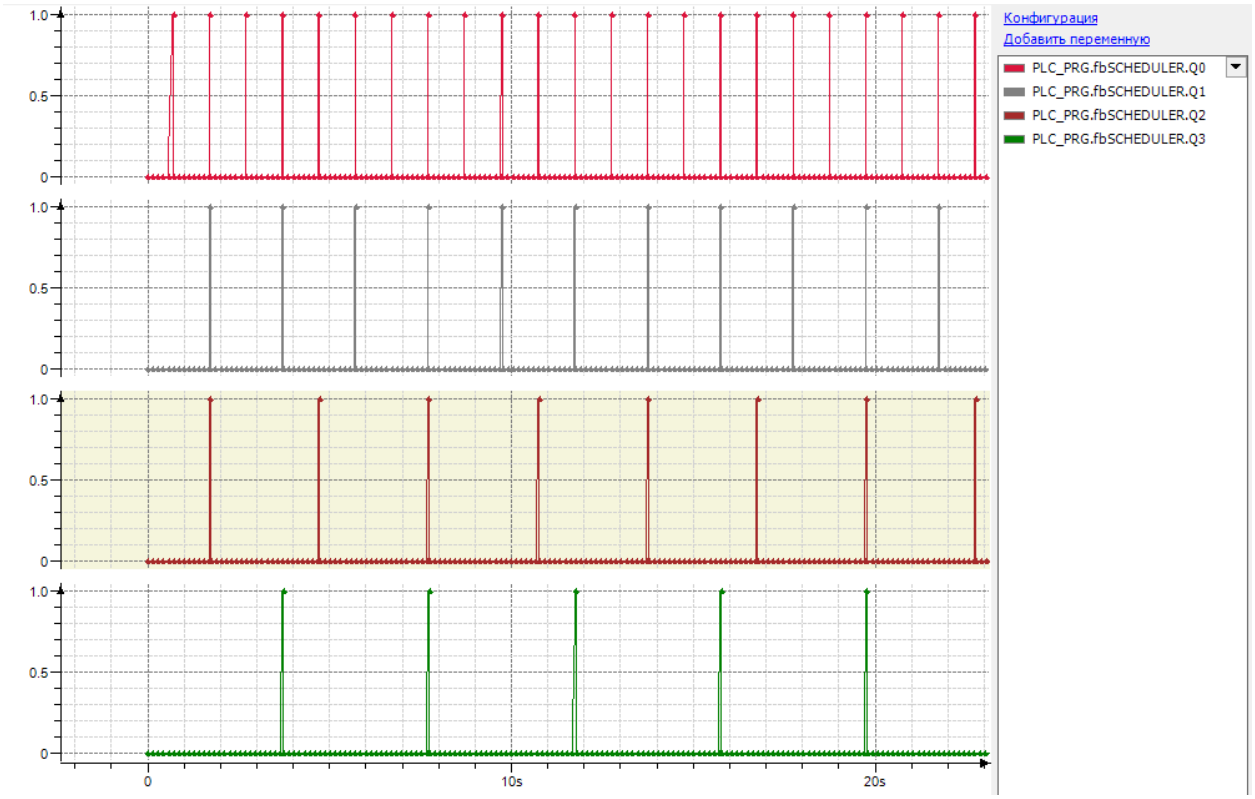


Рис. 15.25. Трассировка работы ФБ **SCHEDULER**

15.14. SCHEDULER_2

Тип модуля: ФБ	**Переменная**	**Тип**	**Описание**
Входы	E0…E3	BOOL	Вход управления каналом генератора.
Выходы	Q0…Q3	BOOL	Выход канала генератора.
Параметры	C0…C3	UINT	Кол-во циклов между импульсами.
	O0…O3	UINT	Кол-во циклов начальной задержки.
Используемые модули	T_PLC_MS		

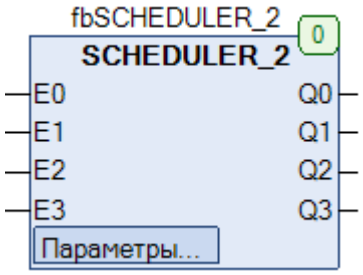


Рис. 15.26. Внешний вид ФБ **SCHEDULER_2** на языке CFC

Функциональный блок **SCHEDULER_2** представляй собой четырехканальный генератор единичных импульсов. Если один из входов **E** принимает значение **TRUE**, то на соответствующем выходе **Q** начинают генерироваться единичные импульсы, пауза между которыми составляет **C** циклов ПЛК. Параметр **O** позволяет настроить задержку (в циклах ПЛК) перед первым импульсом.

На рис. 15.27 приведена трассировка работы блока для ПЛК с циклом 100 мс со следующими значениями параметров: C0=10, C1=20, C2=30, C3=40, O0=O1=5, O2=O3=20. Все каналы генератора были запущены в работу одновременно.

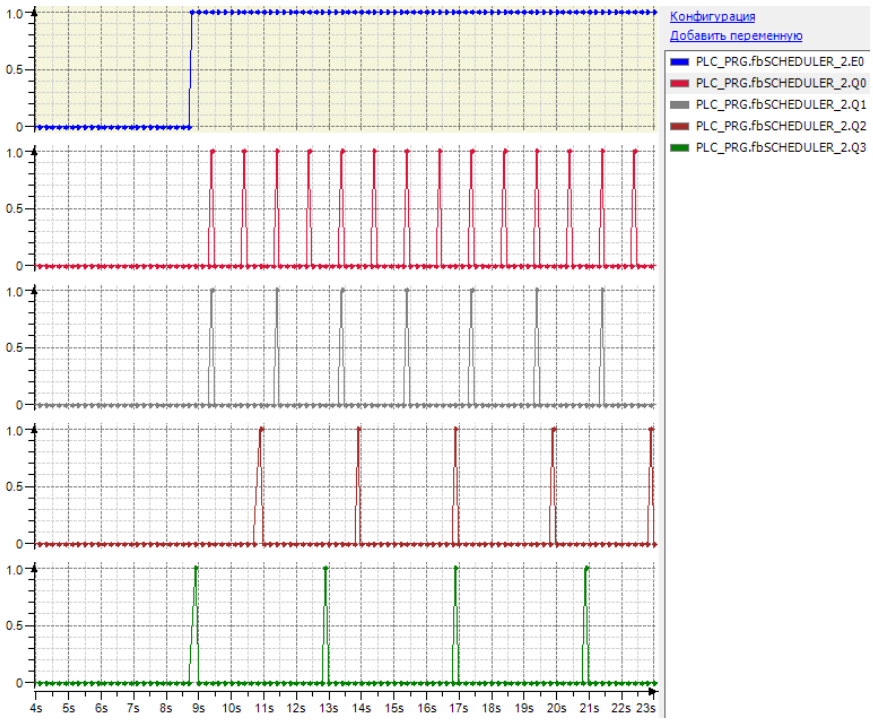


Рис. 15.27. Трассировка работы ФБ **SCHEDULER_2**

15.15. SEQUENCE_4

Тип модуля: ФБ	Переменная	Тип	Описание
Входы	in0…in3	BOOL	Контролируемый сигнал.
	start	BOOL	Вход управления блоком.
	rst	BOOL	Вход сброса блока.
	wait0…wait3	TIME	Время ожидания сигнала.
	delay0…delay3	TIME	Задержка.
Выходы	Q0…Q3	BOOL	Выход секвенсора.
	QX	BOOL	Флаг «один из выходов активен».
	run	BOOL	Флаг «блок в работе».
	step	INT	Номер обрабатываемого канала.
	status	BYTE	Код ошибки/сообщения.
Параметры	stop_on_error	BOOL	Режим обработки ошибок.
Используемые модули	T_PLC_MS		

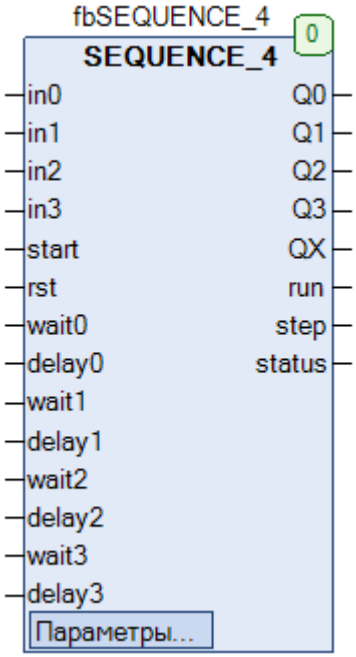


Рис. 15.28. Внешний вид ФБ **SEQUENCE_4** на языке CFC

Функциональный блок **SEQUENCE_4** представляет собой четырехбитный секвенсор. Блок запускается в работу импульсом по переднему фронту на входе **start** и течение времени **wait0** ожидает сигнала **TRUE** на входе **in0**. Когда **in0** принимает значение **TRUE**, выход **Q0** устанавливается в **TRUE** и блок делает задержку на время **delay0**, после чего переходит к ожиданию сигнала **TRUE** на входе **in1** в течение времени **wait1** и т.д. После задержки **delay3** блок прекращает работу. Вход **rst** используется для сброса выходных переменных блока (по переднему фронту). Выход **QX** принимает значение **TRUE**, если хотя бы один из выходов **Q0…Q3** имеет значение **TRUE**. Выход **run** принимает значение **TRUE**, если блок находится в работе. Выход **step** сигнализирует о номере текущего обрабатываемого входа (**-1** – блок не в работе, **0** – in0, …, **3** – in3).

Выход **status** возвращает код ошибки:

Код ошибки	Описание
1	Вход in0 не стал активным в течение времени wait0.
2	Вход in1 не стал активным в течение времени wait1.
3	Вход in2 не стал активным в течение времени wait2.
4	Вход in3 не стал активным в течение времени wait3.
110	Блок не запущен.
111	Блок в работе.

Параметр **stop_on_error** определяет поведение блока при возникновении ошибок с кодом 1-4: **FALSE** – блок продолжает работу, **TRUE** – блок прекращает работу.

На рис. 15.29 приведена трассировка работы блока в случае, когда **delay0=delay2=10 мс**, **delay1=delay3=20 мс**. На входы **in0…in3** не заведены переменные – и по умолчанию они имеют значения **TRUE**. Из-за этого времена **wait0…wait3** не обрабатываются блоком (т.к. сигнал изначально=**TRUE**, то время ожидания=**0**). Иными словами, в данном случае выходы **Q0…Q3** последовательно принимают значение **TRUE** на время **delay0…delay3**.

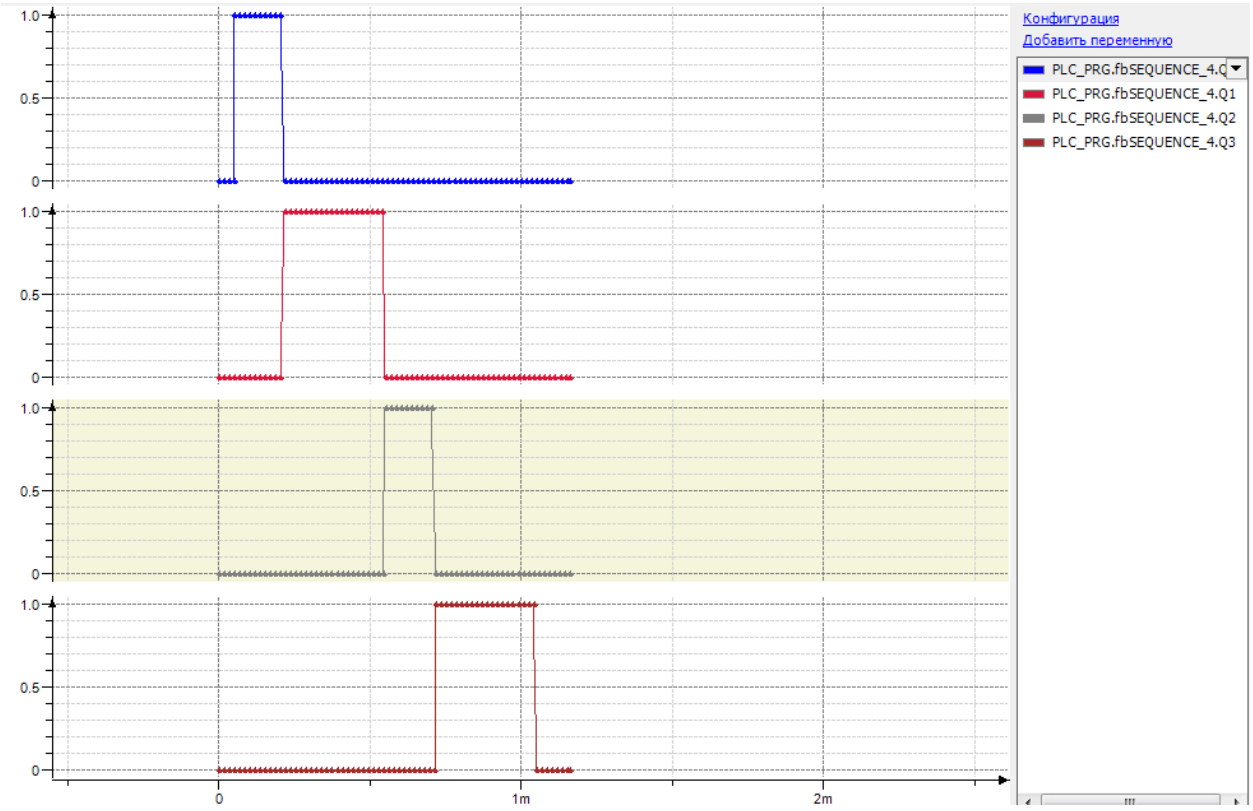


Рис. 15.29. Трассировка работы ФБ **SEQUENCE_4**

15.16. SEQUENCE_64

Тип модуля: ФБ	Переменная	Тип	Описание
Входы	START	BOOL	Вход управления блоком.
	SMAX	INT	Кол-во состояний последовательности.
	PROG	ARRAY [0..63] OF TIME	Периоды состояний последовательности.
	RST	BOOL	Вход сброса блока.
Выходы	STATE	INT	Выход секвенсора.
	TRIG	BOOL	Сигнал смены состояний.
Используемые модули	T_PLC_MS, INC2		

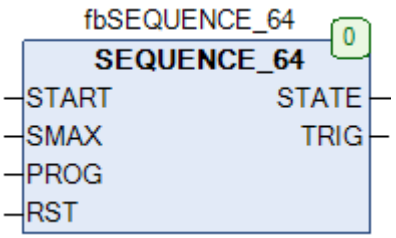


Рис. 15.30. Внешний вид ФБ **SEQUENCE_64** на языке CFC

Функциональный блок **SEQUENCE_64** представляет собой 64-х битный секвенсор. Пока блок не запущен, выход **STATE = -1**. Блок запускается в работу импульсом по переднему фронту на входе **START**, после чего выход **STATE** имеет значение **0** в течение времени **PROG[0],** затем **1** в течение времени **PROG[1]** и т.д. вплоть до **STATE=(SMAX-1)** в течение времени **PROG[SMAX-1],** после чего **STATE** снова принимает значение **-1** и блок прекращает работу. Вход **RST** используется для принудительного прекращения работы блока. На выходе **TRIG** генерируется единичный импульс при каждом изменении выхода **STATE**.

15.17. SEQUENCE_8

Тип модуля: ФБ	Переменная	Тип	Описание
Входы	in0…in7	BOOL	Контролируемый сигнал.
	Start	BOOL	Вход управления блоком.
	rst	BOOL	Вход сброса блока.
	wait0…wait7	TIME	Время ожидания сигнала.
	delay0…delay7	TIME	Задержка.
Выходы	Q0…Q7	BOOL	Выход секвенсора.
	QX	BOOL	Флаг «один из выходов активен».
	run	BOOL	Флаг «блок в работе».
	step	INT	Номер обрабатываемого канала.
	status	BYTE	Код ошибки/сообщения.
Параметры	stop_on_error	BOOL	Режим обработки ошибок.
Используемые модули	T_PLC_MS		

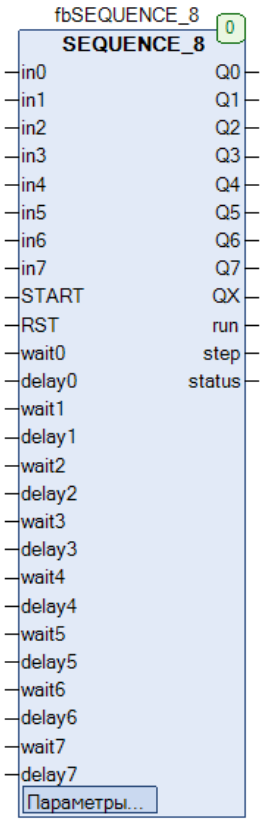


Рис. 15.31. Внешний вид ФБ SEQUENCE_8 на языке CFC

Функциональный блок **SEQUENCE_8** представляет собой восьмибитный секвенсор. Принцип работы полностью соответствует SEQUENCE_4.

15.18. TMAX

Тип модуля: ФБ	Переменная	Тип	Описание
Входы	IN	BOOL	Вход таймера.
	PT	TIME	Максимальное время работы.
Выходы	Q	BOOL	Выход таймера.
	Z	BOOL	Флаг «завершение работы».
Используемые модули	T_PLC_MS		

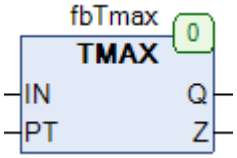


Рис. 15.32. Внешний вид ФБ **TMAX** на языке CFC

Функциональный блок **TMAX** представляет собой таймер с ограничением на максимальное время включения. Пока вход **IN** имеет значение **TRUE**, таймер находится в работе и выход **Q** имеет значение **TRUE**. По истечении времени **PT** выход **Q** принимает значение **FALSE**, а на выходе **Z** генерируется единичный импульс. Если вход **IN** принимает значение **FALSE** до истечения времени **PT**, то таймер прекращает работу и обнуляет выходы.

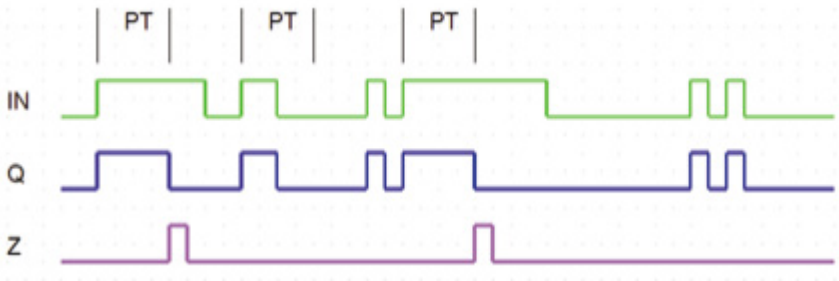


Рис. 15.33. Трассировка работы ФБ **TMAX**

15.19. TMIN

Тип модуля: ФБ	Переменная	Тип	Описание
Входы	IN	BOOL	Вход таймера.
	PT	TIME	Минимальное время работы.
Выходы	Q	BOOL	Выход таймера.

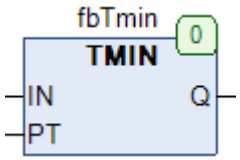


Рис. 15.34. Внешний вид ФБ **TMIN** на языке CFC

Функциональный блок **TMIN** представляет собой таймер с ограничением на минимальное время включения. Пока вход **IN** имеет значение **TRUE**, таймер находится в работе и выход **Q** имеет значение **TRUE**. Если вход **IN** принимает значение **FALSE** до истечения времени **PT**, то по истечении времени **PT** выход **Q** принимает значение **FALSE**. Если вход **IN** принимает значение **FALSE** после истечения времени **PT**, то выход **Q** сразу принимает значение **FALSE**.

Рис. 15.35. Трассировка работы ФБ **TMIN**

15.20. TOF_1

Тип модуля: ФБ	Переменная	Тип	Описание
Входы	IN	BOOL	Вход таймера.
	PT	TIME	Время работы.
	RST	BOOL	Сброс таймера.
Выходы	Q	BOOL	Выход таймера.
Используемые модули	T_PLC_MS		

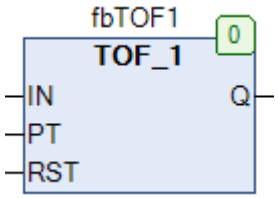


Рис. 15.36. Внешний вид ФБ **TOF_1** на языке CFC

Функциональный блок **TOF1** представляет собой таймер с задержкой отключения. Блок запускается в работу по переднему фронту на входе **IN**, после чего выход **Q** принимает значение **TRUE**. По истечении времени **PT** выход **Q** принимает значение **FALSE**. По переднему фронту на входе **RST** блок прекращает работу и обнуляет выход.

Как можно заметить, в отличие от таймера **TOF** из библиотеки **Standard**, данный ФБ запускается по фронту (а не по уровню) и имеет дополнительный вход для сброса.

Рис. 15.37. Трассировка работы ФБ **TOF_1**

15.21. TONOF

Тип модуля: ФБ	Переменная	Тип	Описание
Входы	IN	BOOL	Вход таймера.
	T_ON	TIME	Задержка включения.
	T_OFF	TIME	Задержка отключения.
Выходы	Q	BOOL	Выход таймера.

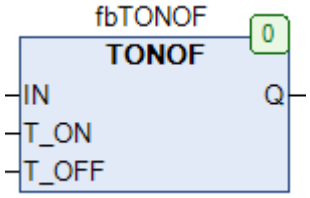


Рис. 15.38. Внешний вид ФБ **TONOF** на языке CFC

Функциональный блок **TONOF** представляет собой таймер с настраиваемыми задержками включения и отключения. Если вход **IN** принимает значение **TRUE**, то блок начинает работу. Спустя время **T_ON** выход **Q** принимает значение **TRUE**. Если вход **IN** принимает значение **FALSE**, то спустя время **T_OFF** выход **Q** принимает значение **FALSE**.

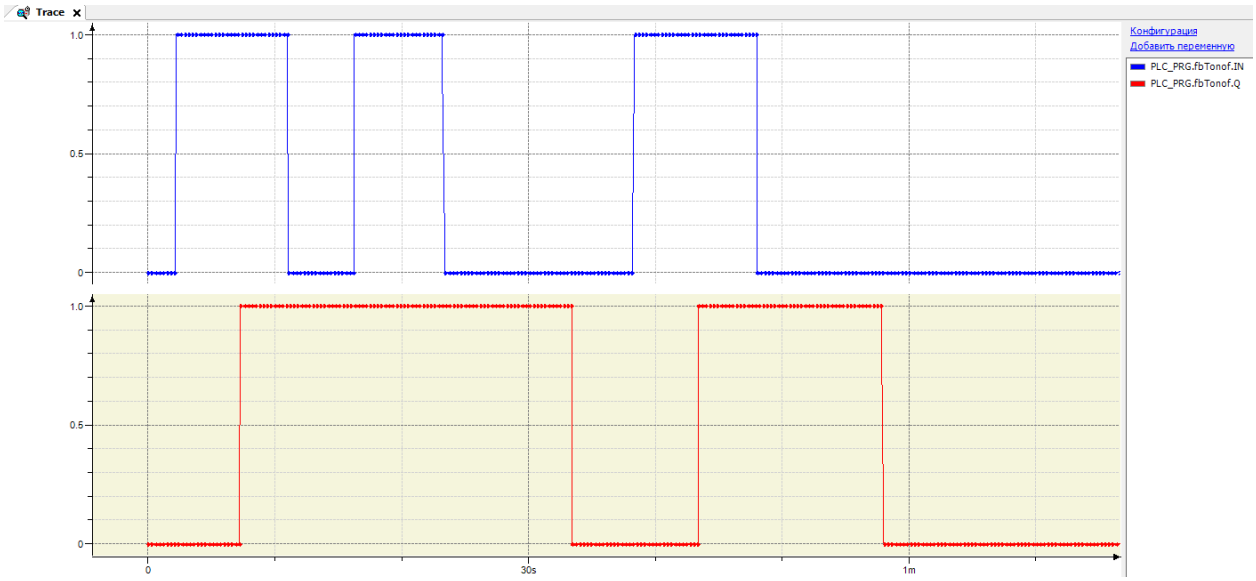


Рис. 15.39. Трассировка работы ФБ **TONOF** (T_ON=T#5s, T_OFF=T#10s)

15.22. TP_1

Тип модуля: ФБ	Переменная	Тип	Описание
Входы	IN	BOOL	Вход таймера.
	PT	TIME	Время импульса.
	RST	BOOL	Сброс таймера.
Выходы	Q	BOOL	Выход таймера.
Используемые модули	T_PLC_MS		

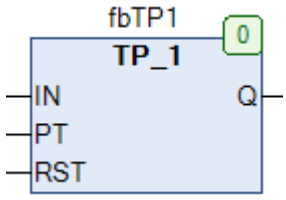


Рис. 15.40. Внешний вид ФБ **TP_1** на языке CFC

Функциональный блок **TP_1** представляет собой таймер. Блок запускается в работу по переднему фронту на входе **IN**, после чего выход **Q** принимает значение **TRUE**. Каждый импульс по переднему фронту на входе **IN** обнуляет счетчик таймера. По истечении времени **PT** выход **Q** принимает значение **FALSE**. По переднему фронту на входе **RST** блок прекращает работу и обнуляет выход.

Как можно заметить, в отличие от таймера **TP** из библиотеки **Standard**, данный ФБ перезапускается после каждого импульса на входе **IN** (в свою очередь **TP** детектирует первый импульс и отрабатывает в независимости от состояния **IN**, после чего может детектировать следующий импульс).

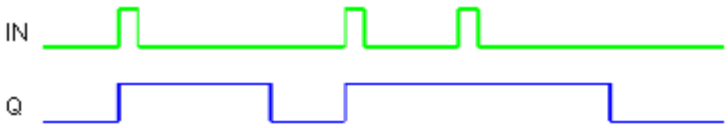


Рис. 15.41. Трассировка работы ФБ **TP_1**

15.23. TP_1D

Тип модуля: ФБ	Переменная	Тип	Описание
Входы	IN	BOOL	Вход таймера.
	PT1	TIME	Время импульса.
	PTD1	TIME	Время задержки.
	RST	BOOL	Сброс таймера.
Выходы	Q	BOOL	Выход таймера.
	W	BOOL	Флаг «задержка».
Используемые модули	T_PLC_MS		

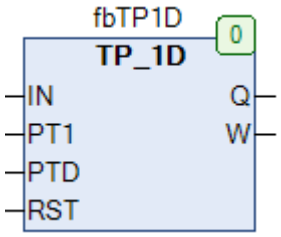


Рис. 15.42. Внешний вид ФБ **TP_1D** на языке CFC

Функциональный блок **TP_1D** представляет собой таймер с задержкой. Блок запускается в работу по переднему фронту на входе **IN**, после чего выход **Q** принимает значение **TRUE**. Каждый импульс по переднему фронту на входе **IN** обнуляет счетчик таймера. По истечении времени **PT1** выход **Q** принимает значение **FALSE**. После этого выход **W** принимает значение **TRUE** и таймер делает задержку на время **PTD1**, во время которой не детектирует импульсы на входе **IN**. По переднему фронту на входе **RST** блок прекращает работу и обнуляет выход.

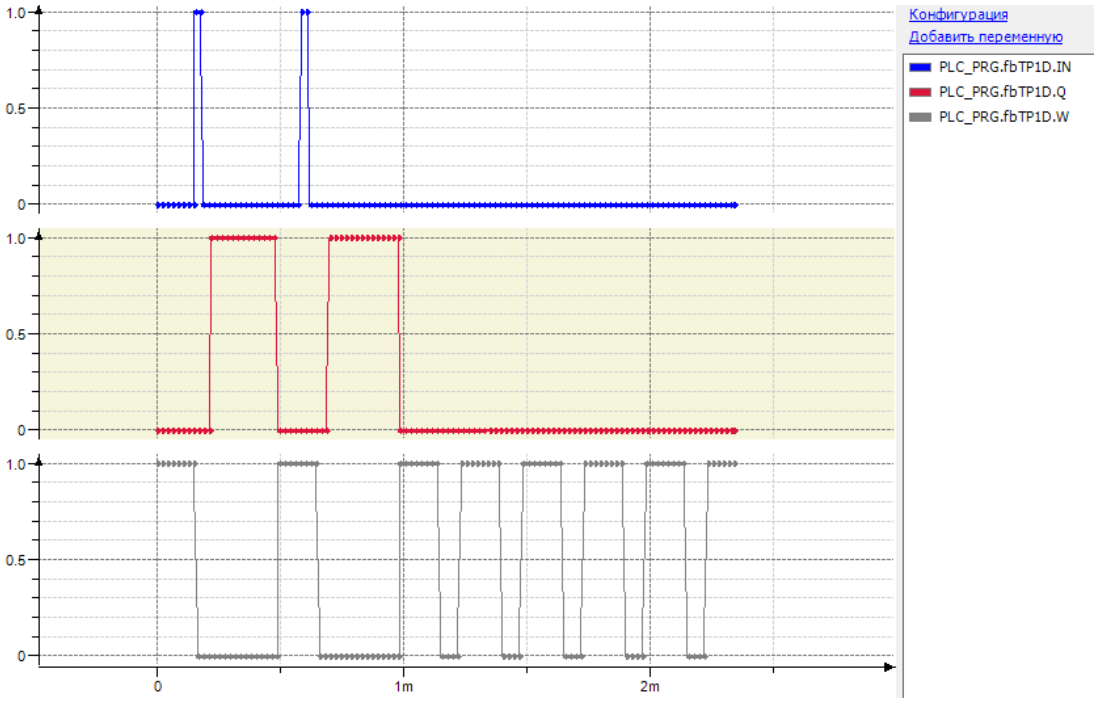


Рис. 15.43. Трассировка работы ФБ **TP_1D**

15.22. TP_X

Тип модуля: ФБ	Переменная	Тип	Описание
Входы	IN	BOOL	Вход таймера.
	PT	TIME	Время импульса.
Выходы	Q	BOOL	Выход таймера.
	ET	TIME	Текущее время таймера.
Используемые модули	T_PLC_MS		

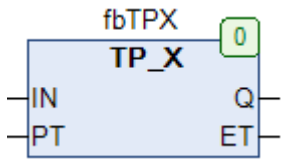


Рис. 15.44. Внешний вид ФБ **TP_X** на языке CFC

Функциональный блок **TP_X** представляет собой таймер. Блок запускается в работу по переднему фронту на входе **IN**, после чего выход **Q** принимает значение **TRUE**. Каждый импульс по переднему фронту на входе **IN** обнуляет счетчик таймера. По истечении времени **PT** выход **Q** принимает значение **FALSE**. На выходе **ET** отображается текущее время таймера.

Как можно заметить, в отличие от таймера **TP** из библиотеки **Standard**, данный ФБ перезапускается после каждого импульса на входе **IN** (в свою очередь **TP** детектирует первый импульс и отрабатывает в независимости от состояния **IN**, после чего может детектировать следующий импульс).

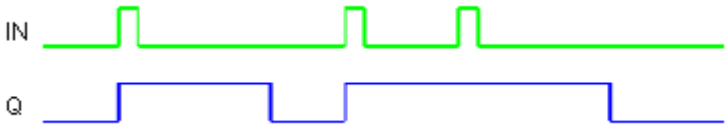


Рис. 15.45. Трассировка работы ФБ **TP_X**

16. Логические модули

16.1. BCDC_TO_INT

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	IN	BYTE	Двухзначное число в формате BCD.
Выходы	BCDC_TO_INT	INT	Двухзначное число в формате DEC.

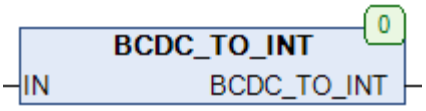


Рис. 16.1. Внешний вид функции **BCDC_TO_INT** на языке CFC

Функция **BCDC_TO_INT** конвертирует двухзначное число из двоично-десятичного формата в десятичный. См. также обратную функцию INT_TO_BCDC.

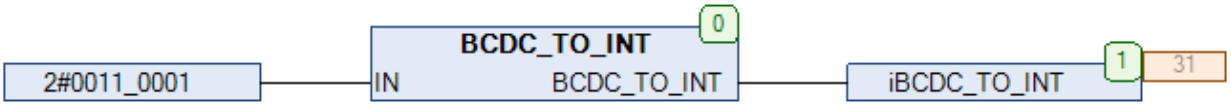


Рис. 16.2. Пример работы с функцией **BCDC_TO_INT** на языке CFC

16.2. BIT_COUNT

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	IN	DWORD	Строка бит.
Выходы	BIT_COUNT	INT	Число бит со значением TRUE.

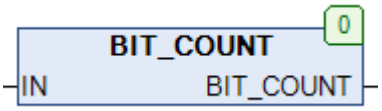


Рис. 16.3. Внешний вид функции **BIN_COUNT** на языке CFC

Функция **BIT_COUNT** возвращает число бит со значением **TRUE** в переменной **IN** типа **DWORD**.

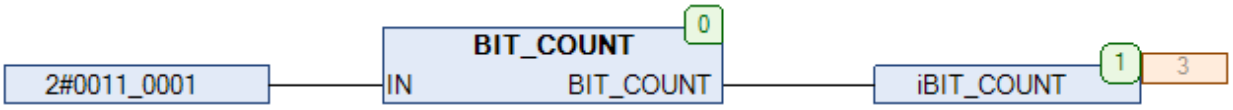


Рис. 16.4. Пример работы с функцией **BIT_COUNT** на языке CFC

16.3. BIT_LOAD_B

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	IN	BYTE	Строка бит.
	VAL	BOOL	Значение бита.
	POS	INT	Номер бита (0…7).
Выходы	BIT_LOAD_B	BYTE	Строка бит с записанным битом.

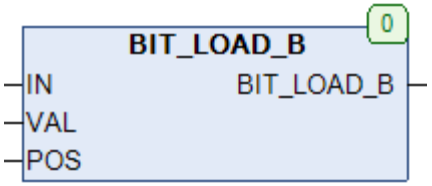


Рис. 16.5. Внешний вид функции **BIT_LOAD_B** на языке CFC

Функция **BIT_LOAD_B** устанавливает бит номер **POS** переменной **IN** типа **BYTE** в значение **VAL**.

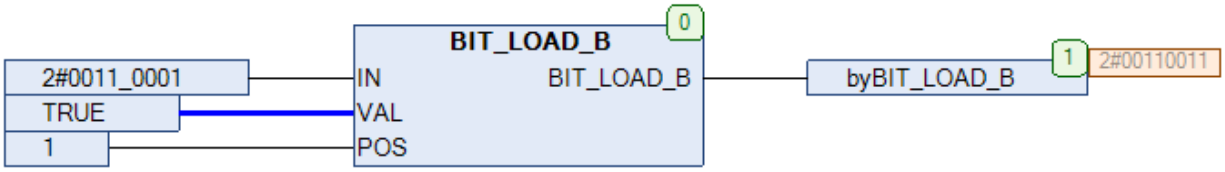


Рис. 16.6. Пример работы с функцией **BIT_LOAD_B** на языке CFC

16.4. BIT_LOAD_B2

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	I	BYTE	Строка бит.
	D	BOOL	Значение бита.
	P	INT	Номер начального бита для записи (0…7).
	N	INT	Число записываемых бит.
Выходы	BIT_LOAD_B2	BYTE	Строка бит с записанными битами.

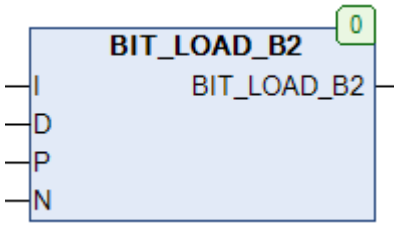


Рис. 16.7. Внешний вид функции **BIT_LOAD_B2** на языке CFC

Функция **BIT_LOAD_B2** устанавливает **N** бит переменной **I** типа **BYTE**, начиная с **P**-го, в значение **D**.

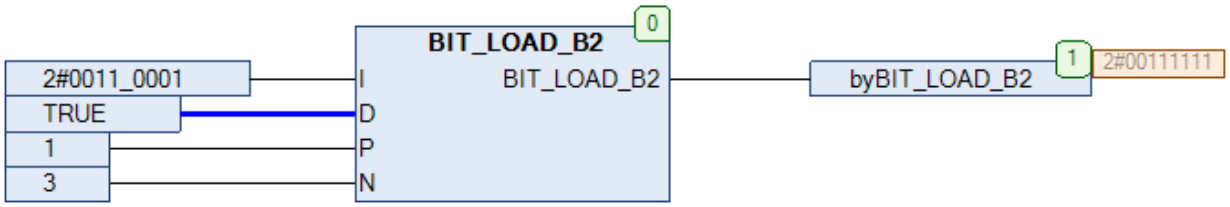


Рис. 16.8. Пример работы с функцией **BIT_LOAD_B2** на языке CFC

16.5. BIT_LOAD_DW

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	IN	DWORD	Строка бит.
	VAL	BOOL	Значение бита.
	POS	INT	Номер бита (0…31).
Выходы	BIT_LOAD_DW	DWORD	Строка бит с записанным битом.

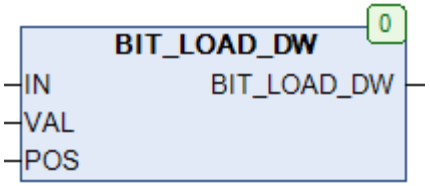


Рис. 16.9. Внешний вид функции **BIT_LOAD_DW** на языке CFC

Функция **BIT_LOAD_DW** устанавливает бит номер **POS** переменной **IN** типа **DWORD** в значение **VAL**.

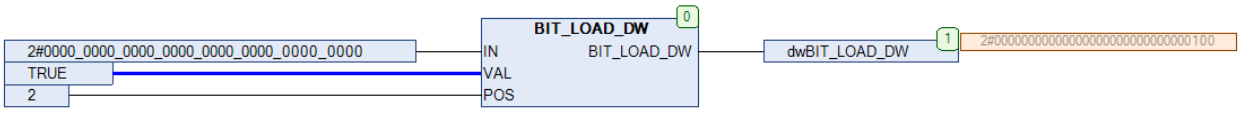


Рис. 16.10. Пример работы с функцией **BIT_LOAD_DW** на языке CFC

16.6. BIT_LOAD_DW2

Тип модуля: функция	Переменная	Тип	Описание
Входы	I	DWORD	Строка бит.
	D	BOOL	Значение бита.
	P	INT	Номер начального бита для записи (0…31).
	N	INT	Число записываемых бит.
Выходы	BIT_LOAD_DW2	DWORD	Строка бит с записанными битами.

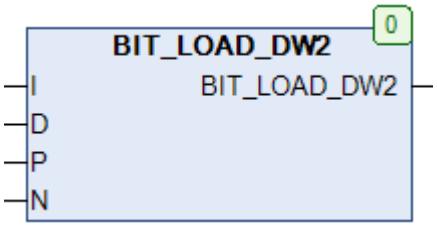


Рис. 16.11. Внешний вид функции **BIT_LOAD_DW2** на языке CFC

Функция **BIT_LOAD_B2** устанавливает **N** бит переменной **I** типа **DWORD**, начиная с **P**-го, в значение **D**.

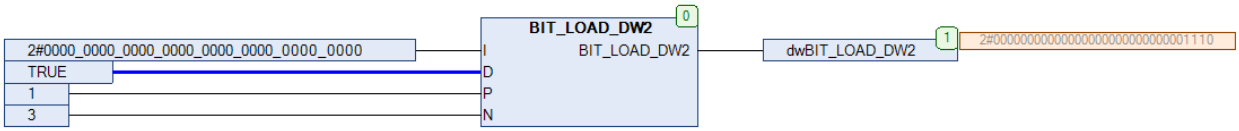


Рис. 16.12. Пример работы с функцией **BIT_LOAD_DW2** на языке CFC

16.7. BIT_LOAD_W

Тип модуля: функция	Переменная	Тип	Описание
Входы	IN	WORD	Строка бит.
	VAL	BOOL	Значение бита.
	POS	INT	Номер бита (0…15).
Выходы	BIT_LOAD_W	WORD	Строка бит с записанным битом.

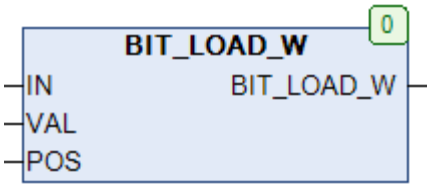


Рис. 16.13. Внешний вид функции **BIT_LOAD_W** на языке CFC

Функция **BIT_LOAD_W** устанавливает бит номер **POS** переменной **IN** типа **WORD** в значение **VAL**.

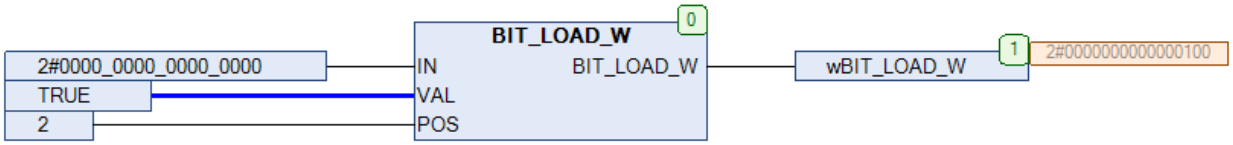


Рис. 16.14. Пример работы с функцией **BIT_LOAD_W** на языке CFC

16.8. BIT_LOAD_W2

Тип модуля: функция	Переменная	Тип	Описание
Входы	I	WORD	Строка бит.
	D	BOOL	Значение бита.
	P	INT	Номер начального бита для записи (0…15).
	N	INT	Число записываемых бит.
Выходы	BIT_LOAD_W2	WORD	Строка бит с записанными битами.

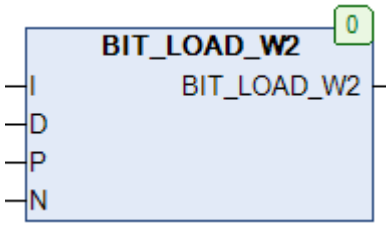


Рис. 16.15. Внешний вид функции **BIT_LOAD_W2** на языке CFC

Функция **BIT_LOAD_W2** устанавливает **N** бит переменной **I** типа **WORD**, начиная с **P**-го, в значение **D**.

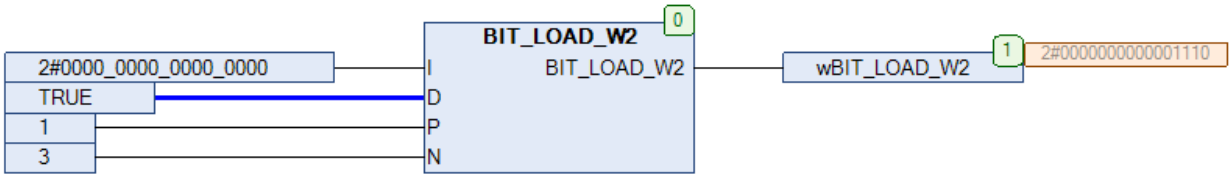


Рис. 16.16. Пример работы с функцией **BIT_LOAD_W2** на языке CFC

16.9. BIT_OF_DWORD

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	in	DWORD	Строка бит.
	N	INT	Номер бита (0…31).
Выходы	BIT_OF_DWORD	BOOL	Значение бита.

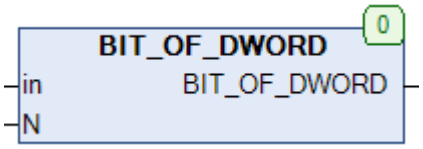


Рис. 16.17. Внешний вид функции **BIT_OF_DWORD** на языке CFC

Функция **BIT_OF_DWORD** возвращает значение бита номер **N** переменной **in**.

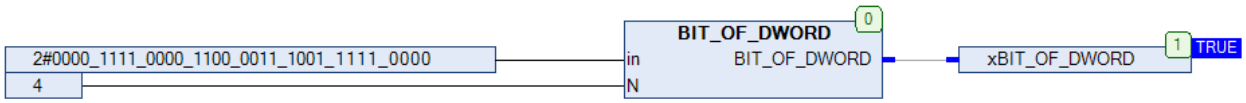


Рис. 16.18. Пример работы с функцией **BIT_OF_DWORD** на языке CFC

16.10. BIT_TOGGLE_B

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	in	BYTE	Строка бит.
	POS	INT	Номер бита (0…7).
Выходы	BIT_TOGGLE_B	BYTE	Строка бит с инвертированным битом.

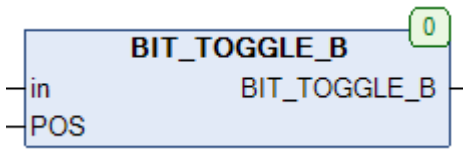


Рис. 16.19. Внешний вид функции **BIT_TOGGLE_B** на языке CFC

Функция **BIT_TOGGLE_B** инвертирует бит **N** переменной **in**.

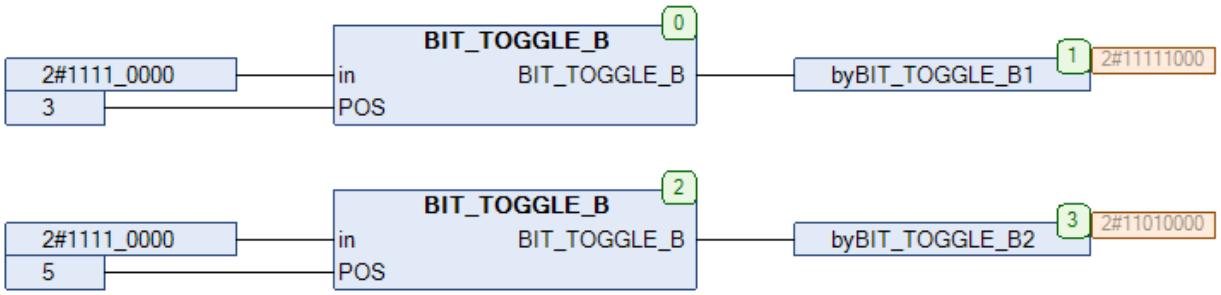


Рис. 16.20. Пример работы с функцией **BIT_TOGGLE_B** на языке CFC

16.11. BIT_TOGGLE_DW

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	in	DWORD	Строка бит.
	POS	INT	Номер бита (0…31).
Выходы	BIT_TOGGLE_DW	DWORD	Строка бит с инвертированным битом.

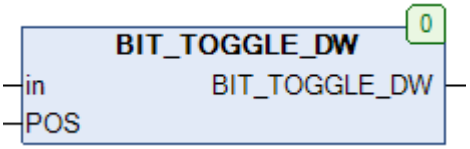


Рис. 16.21. Внешний вид функции **BIT_TOGGLE_DW** на языке CFC

Функция **BIT_TOGGLE_DW** инвертирует бит **N** переменной **in**.

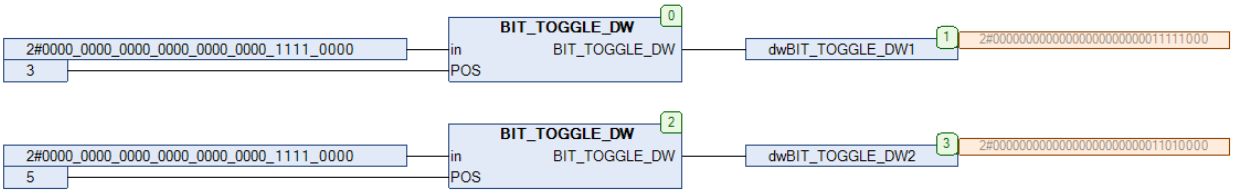


Рис. 16.22. Пример работы с функцией **BIT_TOGGLE_DW** на языке CFC

16.12. BIT_TOGGLE_W

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	in	WORD	Строка бит.
	POS	INT	Номер бита (0…15).
Выходы	BIT_TOGGLE_W	WORD	Строка бит с инвертированным битом.

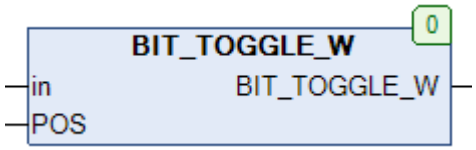


Рис. 16.23. Внешний вид функции **BIT_TOGGLE_W** на языке CFC

Функция **BIT_TOGGLE_W** инвертирует бит **N** переменной **in**.

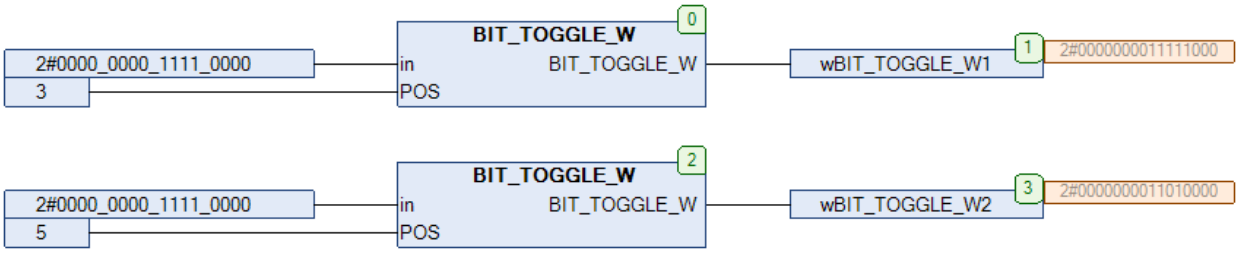


Рис. 16.24. Пример работы с функцией **BIT_TOGGLE_W** на языке CFC

16.13. BYTE_OF_BIT

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	B0…B7	BOOL	Отдельные биты (B0 – младший).
Выходы	BYTE_OF_BIT	BYTE	Байт, собранный из отдельных бит.

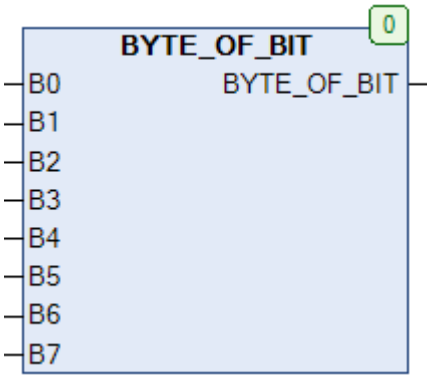


Рис. 16.25. Внешний вид функции **BYTE_OF_BIT** на языке CFC

Функция **BYTE_OF_BIT** собирает 8 **BOOL** переменных в переменную типа **BYTE**.

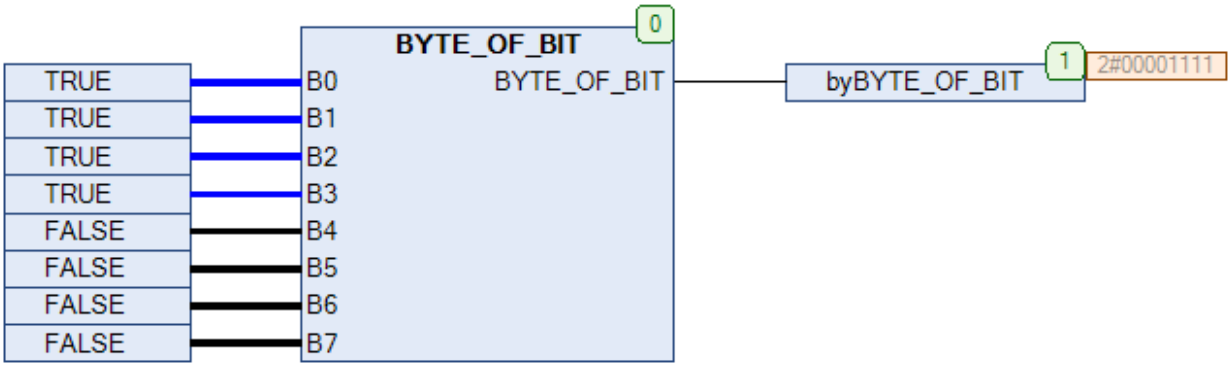


Рис. 16.26. Пример работы с функцией **BYTE_OF_BIT** на языке CFC

16.14. BYTE_OF_DWORD

Тип модуля: функция	Переменная	Тип	Описание
Входы	in	DWORD	Исходная переменная.
	N	BYTE	Номер считываемого байта (0…3).
Выходы	BYTE_OF_DWORD	BYTE	Выделенный байт.

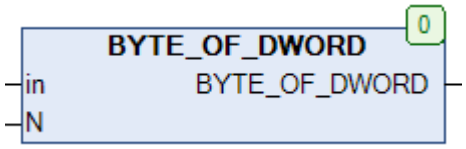


Рис. 16.27. Внешний вид функции **BYTE_OF_DWORD** на языке CFC

Функция **BYTE_OF_DWORD** выделяет из переменной **in** типа **DWORD** байт с номер **N**. См. также обратную функцию DWORD_OF_BYTE.

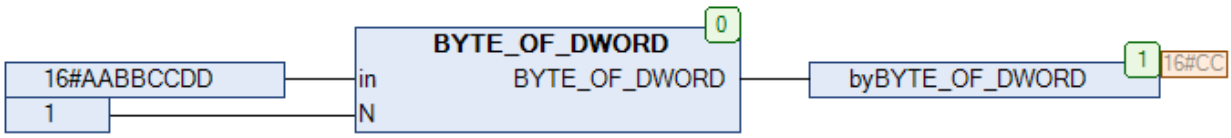


Рис. 16.28. Пример работы с функцией **BYTE_OF_DWORD** на языке CFC

16.15. BYTE_TO_BITS

Тип модуля: ФБ	**Переменная**	**Тип**	**Описание**
Входы	in	BYTE	Исходная переменная.
Выходы	B0…B7	BOOL	Выделенные биты (B0 – младший).

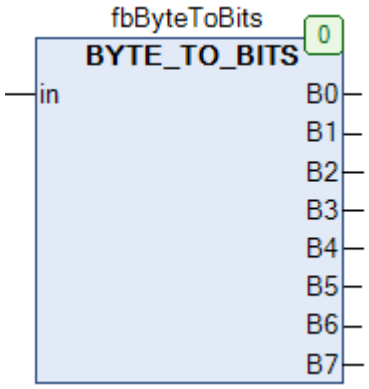


Рис. 16.29. Внешний вид ФБ **BYTE_TO_BITS** на языке CFC

ФБ **BYTE_TO_BITS** извлекает отдельные биты из переменной **in** типа **BYTE**.

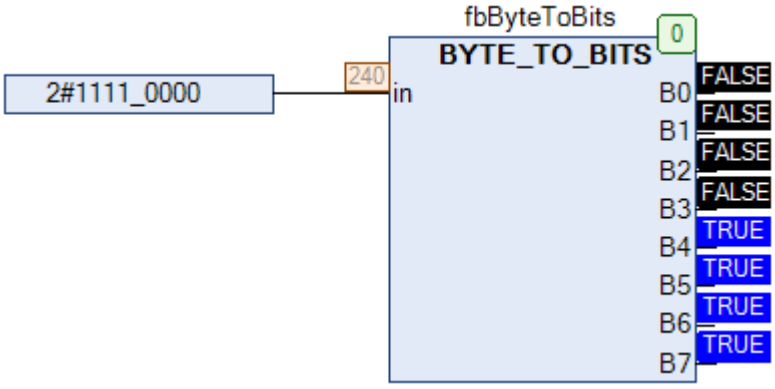


Рис. 16.30. Пример работы с ФБ **BYTE_TO_BITS** на языке CFC

16.16. BYTE_TO_GRAY

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	in	BYTE	Число в двоичном коде.
Выходы	BYTE_TO_GRAY	BYTE	Число в коде Грея.

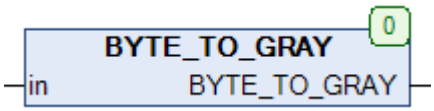


Рис. 16.31. Внешний вид функции **BYTE_TO_GRAY** на языке CFC

Функция **BYTE_TO_GRAY** преобразует число в двоичном коде **in** в число в коде Грея. См. также обратную функцию GRAY_TO_BYTE.

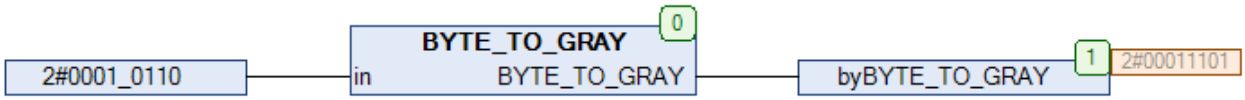


Рис. 16.32. Пример работы с функцией **BYTE_TO_GRAY** на языке CFC

16.17. CHK_REAL

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	X	REAL	Проверяемое число с плавающей точкой.
Выходы	CHK_REAL	BYTE	Код состояния числа.

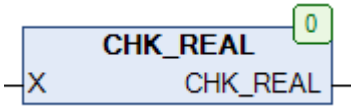


Рис. 16.33. Внешний вид функции **CHK_REAL** на языке CFC

Функция **CHK_REAL** проверяет состояние числа с плавающей точкой **X** и возвращает код состояния согласно стандарту IEEE 754-2008:

- 16#00 – число с плавающей точкой;
- 16#20 – +∞;
- 16#40 – −∞;
- 16#80 – NaN.

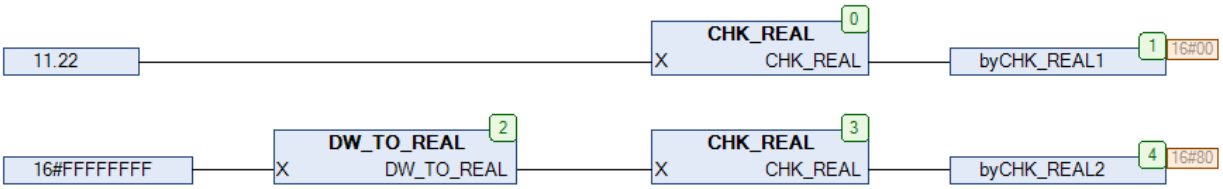


Рис. 16.34. Пример работы с функцией **CHK_REAL** на языке CFC

16.18. CHECK_PARITY

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	in	DWORD	Проверяемая переменная.
	p	BOOL	Бит четности.
Выходы	CHECK_PARITY	BOOL	Результат проверки.

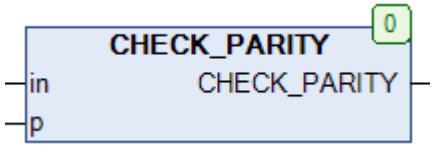


Рис. 16.35. Внешний вид функции **CHECK_PARITY** на языке CFC

Функция **CHECK_PARITY** проверяет переменную **in** типа **DWORD** с битом четности **p** на четность и возвращает **TRUE** в случае, есть общее число битов со значением **TRUE** (включая значение бита четности) является четным числом. См. также функцию PARITY.

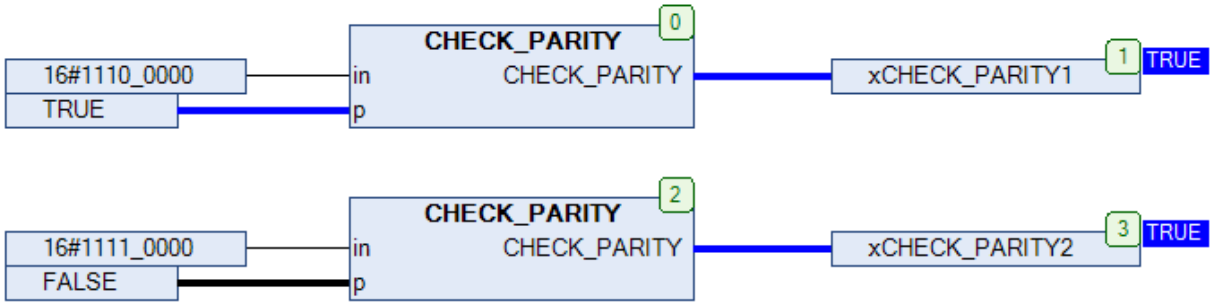


Рис. 16.36. Пример работы с функцией **CHECK_PARITY** на языке CFC

16.19. CRC_CHECK

Функция **CRG_CHECK** была удалена из библиотеки, так как ее функционал реализован в функции CRC_GEN. Обычно ФБ **CRG_GEN** используется для генерации контрольной суммы для выбранного пакета данных. Если после этого сгенерировать контрольную сумму для пакета данных с приложенной корректной контрольной суммой, то в результате получится **0** – таким образом можно проверить корректность контрольной суммы пакета.

16.20. CRC_GEN

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	PT	POINTER TO ARRAY [0..32000] OF BYTE	Пакет данных, для которого будет сгенерирована контрольная сумма.
	SIZE	INT	Размер пакета данных (в байтах).
	PL	INT	Степень порождающего полинома.
	PN	DWORD	Порождающий полином.
	INIT	DWORD	Стартовые данные.
	REV_IN	BOOL	Направление вычислений.
	REV_OUT	BOOL	Инверсия порядка битов.
	XOR_OUT	DWORD	Значение для XOR.
Выходы	CRG_GEN	DWORD	Сгенерированная контрольная сумма.
Используемые модули	REVERSE, REFLECT		

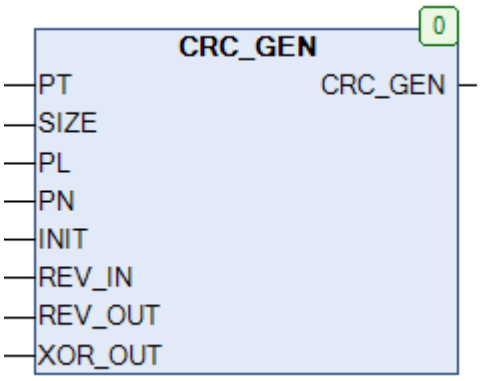


Рис. 16.37. Внешний вид функции **CRC_GEN** на языке CFC

Функция **CRC_GEN** генерирует контрольную сумму (по алгоритму CRC) для пакета данных размером **SIZE** байт, расположенных по указателю **PT**. В большинстве случаев при вызове функции используются операторы **ADR()** и **SIZEOF()** – первый возвращает указатель на заданную переменную (массив байт), а второй – ее размер в байтах.

Входная переменная **PN** определяет производящий полином для функции, **PL** – его степень. Максимально допустимая степень полинома, используемого в функции – **32**. Вход **INIT** определяет стартовые данные для генерации (обычно **0x0000** или **0xFFFF**). Вход **REV_IN** определяет порядок вычислений: **FALSE** – со старшего значащего бита, **TRUE** – с младшего. Вход **REV_OUT** определяет порядок бит при входе на элемент **XOR**: **FALSE** – прямой, **TRUE** – инвертированный. Вход **XOR_OUT** определяет число, которое используется при операции **XOR**.

Функция способна работать с пакетами данных с длиной **не менее 4 байта**.

Ниже приведены значения входов функции для некоторых алгоритмов **CRC**. Более подробная информация может быть найдена в сети. Приведем несколько полезных ссылок:

- http://reveng.sourceforge.net/crc-catalogue/ (каталог алгоритмов CRC)
- http://zorc.breitbandkatze.de/crc.html (онлайн-генератор контрольных сумм)
- http://www.tahapaksu.com/crc/ (еще один онлайн-генератор, удобен при работе с Modbus)

Таблица аргументов функции **CRC_GEN** для некоторых алгоритмов **CRC**

CRC	PL (DEC)	PN (HEX)	INIT (HEX)	REV_IN	REV_OUT	XOR_OUT (HEX)
CRC-3/ROHC	3	0x3	0x7	TRUE	TRUE	0x0
CRC-4/ITU	4	0x3	0x0	TRUE	TRUE	0x0
CRC5/EPC	5	0x9	0x9	FALSE	FALSE	0x0
CRC-5/ITU	5	0x15	0x0	TRUE	TRUE	0x0
CRC-5/USB	5	0x5	0x1F	TRUE	TRUE	0x1F
CRC-6/DARC	6	0x19	0x0	TRUE	FALSE	0x0
CRC-6/ITU	6	0x3	0x0	TRUE	TRUE	0x0
CRC-7	7	0x9	0x0	FALSE	FALSE	0x0
CRC-7/ROHC	7	0x4F	0x7F	TRUE	TRUE	0x0
CRC-8	8	0x7	0x0	FALSE	FALSE	0x0
CRC-8/DARC	8	0x39	0x0	TRUE	TRUE	0x0
CRC-8/I-CODE	8	0x1D	0xFD	FALSE	FALSE	0x0
CRC-8/ITU	8	0x7	0x0	FALSE	FALSE	0x55
CRC-8/MAXIM	8	0x31	0x0	TRUE	TRUE	0x0
CRC-8/ROHC	8	0x7	0xFF	TRUE	TRUE	0x0
CRC-8/WCDNA	8	0x9B	0x0	TRUE	TRUE	0x0
CRC-10	10	0x233	0x0	FALSE	FALSE	0x0
CRC-11	11	0x385	0x1A	FALSE	FALSE	0x0
CRC-12/3GPP	12	0x80F	0x0	FALSE	TRUE	0x0
CRC-12/DECT	12	0x80F	0x0	FALSE	FALSE	0x0
CRC-14/DARC	14	0x805	0x0	TRUE	TRUE	0x0
CRC-15	15	0x4599	0x0	FALSE	FALSE	0x0
CRC-16/LHA	16	0x8005	0x0	TRUE	TRUE	0x0
CRC-16/CCITT-AUG	16	0x1021	0x1D0F	FALSE	FALSE	0x0
CRC-16/BUYPASS	16	0x8005	0x0	FALSE	FALSE	0x0
CRC-16/CCITT-FALSE	16	0x1021	0xFFFF	FALSE	FALSE	0x0
CRC-16/DDS	16	0x8005	0x800D	FALSE	FALSE	0x0
CRC-16/DECT-R	16	0x589	0x0	FALSE	FALSE	0x1

CRC	PL (DEC)	PN (HEX)	INIT (HEX)	REV_IN	REV_OUT	XOR_OUT (HEX)
CRC-16/DECT-X	16	0x589	0x0	FALSE	FALSE	0x0
CRC-16/DNP	16	0x3D65	0x0	TRUE	TRUE	0xFFFF
CRC-16/EN13757	16	0x3D65	0x0	FALSE	FALSE	0xFFFF
CRC-16/GENIBUS	16	0x1021	0xFFFF	FALSE	FALSE	0xFFFF
CRC-16/MAXIM	16	0x8005	0x0	TRUE	TRUE	0xFFFF
CRC-16/MCRF4XX	16	0x1021	0xFFFF	TRUE	TRUE	0x0
CRC-16/RIELLO	16	0x1021	0xB2AA	TRUE	TRUE	0x0
CRC-16/T10-DIF	16	0x8BB7	0x0	FALSE	FALSE	0x0
CRC-16/TELEDISK	16	0xA097	0x0	FALSE	FALSE	0x0
CRC-16/USB	16	0x8005	0xFFFF	TRUE	TRUE	0xFFFF
CRC-16/CCITT-TRUE	16	0x1021	0x0	TRUE	TRUE	0x0
CRC-16/MODBUS	16	0x8005	0x0	TRUE	TRUE	0x0
CRC-16/X-25	16	0x1021	0xFFFF	TRUE	TRUE	0xFFFF
CRC-16/XMODEM	16	0x1021	0x0	FALSE	FALSE	0x0
CRC-24/OPENPGP	24	0x864CFB	0xB704CE	FALSE	FALSE	0x0
CRC-24/FLEXRAY-A	24	0x5D6DCB	0xFEDCBA	FALSE	FALSE	0x0
CRC-24/FLEXRAY-B	24	0x5D6DCB	0xABCDEF	FALSE	FALSE	0x0
CRC-32/PKZIP	32	0x04C11DB7	0xFFFFFFFF	TRUE	TRUE	0xFFFFFFFF
CRC-32/BZIP2	32	0x04C11DB7	0xFFFFFFFF	FALSE	FALSE	0xFFFFFFFF
CRC-32/CASTAGNOLI	32	0x1EDC6F41	0xFFFFFFFF	TRUE	TRUE	0xFFFFFFFF
CRC-32/D	32	0xA833982B	0xFFFFFFFF	TRUE	TRUE	0xFFFFFFFF
CRC-32/MPEG2	32	0x04C11DB7	0xFFFFFFFF	FALSE	FALSE	0x0
CRC-32/POSIX	32	0x04C11DB7	0x0	FALSE	FALSE	0xFFFFFFFF
CRC-32/Q	32	0x814141AB	0x0	FALSE	FALSE	0x0
CRC-32/JAM	32	0x04C11DB7	0xFFFFFFFF	TRUE	TRUE	0x0
CRC-32/XFER	32	0xAF	0x0	FALSE	FALSE	0x0

Ниже приведен пример использования функции **CRC_GEN** для расчета контрольной суммы пакета данных протокола Modbus. ***Обратите внимание***, что рассчитанная контрольная сумма представляет собой переменную типа **DWORD**, младший байт которой соответствует старшему байту контрольной суммы, прикладываемой к пакету данных.

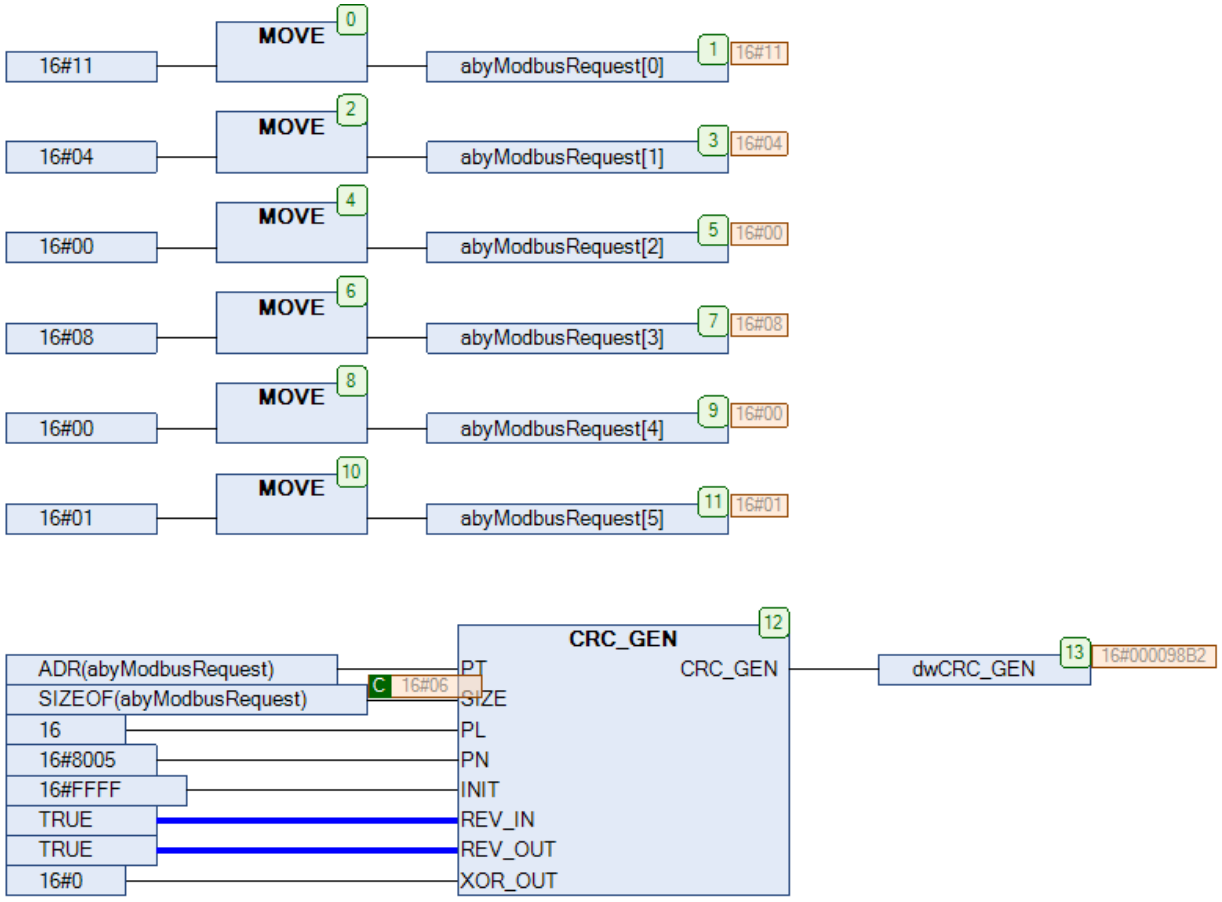


Рис. 16.38. Пример работы с функцией **CRC_GEN** на языке CFC

Проверка полученной контрольной суммы с помощью онлайн-парсера Modbus от разработчиков RapidScada:

Data Package (Application Data Unit):

```
11 04 00 08 00 01 B2 98
```

Parse

Part of Data Package	Description	Value
11	Slave address	0x11 (17)
04	Function code	0x04 (4) - Read Input Registers
00 08	Starting address	0x0009 (9)
00 01	Quantity	0x0001 (1)
B2 98	CRC	0xB298 (45720)

16.21. DEC_2

Тип модуля: ФБ	Переменная	Тип	Описание
Входы	D	BOOL	Вход декодера.
	A	BOOL	Бит разрешения.
Выходы	Q0	BOOL	Выход 0.
	Q1	BOOL	Выход 1.

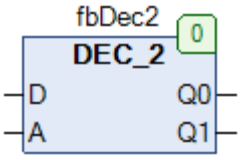


Рис. 16.39. Внешний вид ФБ **DEC_2** на языке CFC

Функциональный блок **DEC_2** представляет собой двухбитный декодер. Принцип работы блока:

- если A=FALSE, то Q0:=D;
- если A=TRUE, то Q1:=D.

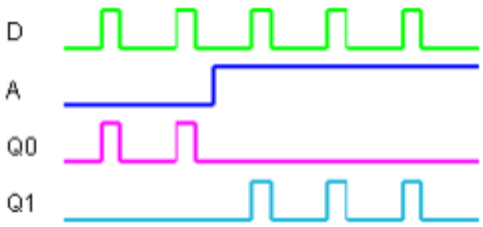


Рис. 16.40. Трассировка работы ФБ **DEC_2**

16.22. DEC_4

Тип модуля: ФБ	Переменная	Тип	Описание
Входы	D	BOOL	Вход декодера.
	A0	BOOL	Бит разрешения 0.
	A1	BOOL	Бит разрешения 1.
Выходы	Q0	BOOL	Выход 0.
	Q1	BOOL	Выход 1.
	Q2	BOOL	Выход 2.
	Q3	BOOL	Выход 3.

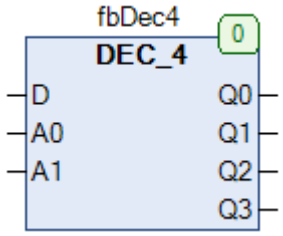


Рис. 16.41. Внешний вид ФБ **DEC_4** на языке CFC

Функциональный блок **DEC_4** представляет собой четырехбитный декодер. Принцип работы блока:

- если A0=FALSE и A1=FALSE, то Q0:=D;
- если A0=TRUE и A1=FALSE , то Q1:=D;
- если A0=FALSE и A1=TRUE, то Q2:=D;
- если A0=TRUE и A1=TRUE , то Q3:=D;

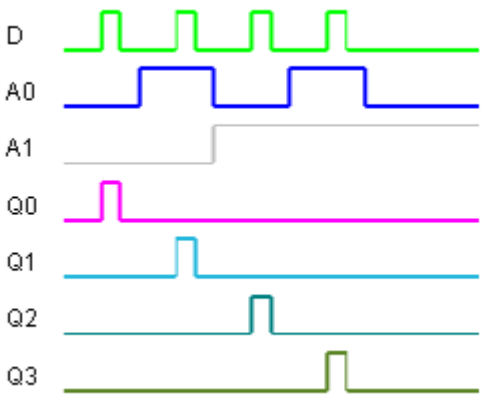


Рис. 16.42. Трассировка работы ФБ **DEC_4**

16.23. DEC_8

Тип модуля: ФБ	Переменная	Тип	Описание
Входы	D	BOOL	Вход декодера.
	A0	BOOL	Бит разрешения 0.
	A1	BOOL	Бит разрешения 1.
	A2	BOOL	Бит разрешения 2.
Выходы	Q0	BOOL	Выход 0.
	Q1	BOOL	Выход 1.
	Q2	BOOL	Выход 2.
	Q3	BOOL	Выход 3.
	Q4	BOOL	Выход 4.
	Q5	BOOL	Выход 5.
	Q6	BOOL	Выход 6.
	Q7	BOOL	Выход 7.

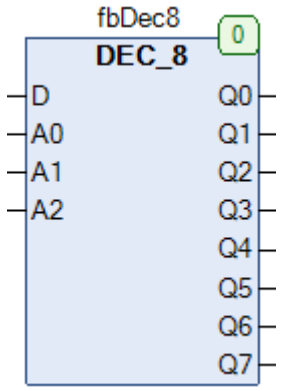


Рис. 16.43. Внешний вид ФБ **DEC_8** на языке CFC

Функциональный блок **DEC_8** представляет собой восьмибитный декодер. Принцип работы блока:

- если A0=FALSE и A1=FALSE и A2=FALSE, то Q0:=D;
- если A0=TRUE и A1=FALSE и A2=FALSE, то Q1:=D;
- если A0=FALSE и A1=TRUE и A2=FALSE, то Q2:=D;
- если A0=TRUE и A1=TRUE и A2=FALSE, то Q3:=D;
- если A0=FALSE и A1=FALSE и A2=TRUE, то Q4:=D;
- если A0=TRUE и A1=FALSE и A2=TRUE, то Q5:=D;
- если A0=FALSE и A1=TRUE и A2=TRUE, то Q6:=D;
- если A0=TRUE и A1=TRUE и A2=TRUE, то Q7:=D;

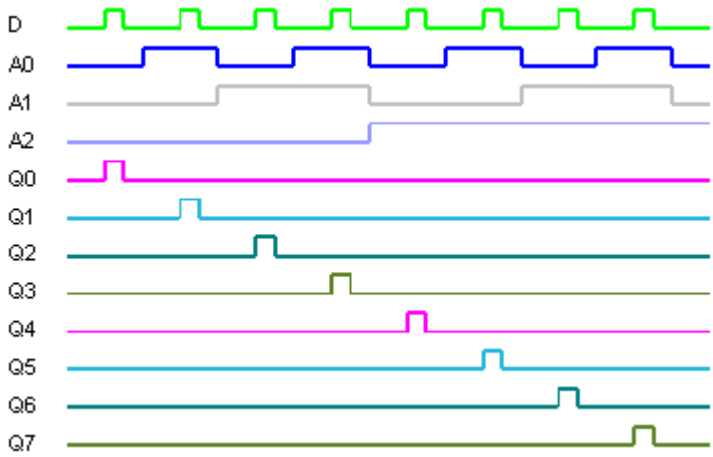


Рис. 16.44. Трассировка работы ФБ **DEC_8**

16.24. DW_TO_REAL

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	X	DWORD	Входное значение.
Выходы	DW_TO_REAL	REAL	Представление входного значения как REAL.

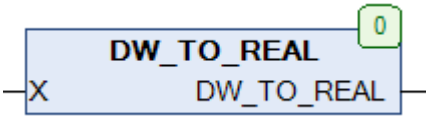


Рис. 16.45. Внешний вид функции **DW_TO_REAL** на языке CFC

Функция **DW_TO_REAL** интерпретирует входную переменную **X** типа **DWORD** как значение с плавающей точкой, представленное в формате IEEE 754. В отличие от стандартного оператора конверсии **DWORD_TO_REAL**, который преобразует десятичное представление переменной типа **DWORD** в соответствующее число с плавающей точкой (например, 5--->5.0), данная функция не меняет исходное значение, а только представляет его в другом формате. В частности, это необходимо при передаче переменных типа **REAL** по протоколу Modbus, который не стандартизирует передаваемые типы данных.

См. также обратную функцию REAL_TO_DW.

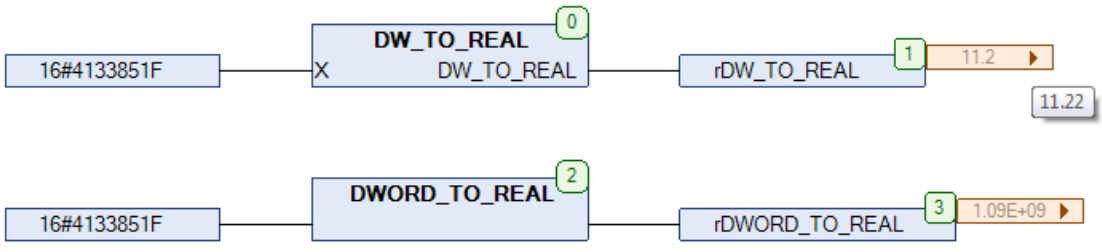


Рис. 16.46. Пример работы с функцией **DW_TO_REAL** на языке CFC

16.25. DWORD_OF_BYTE

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	B3…B0	BYTE	Исходные байты (B0 – младший).
Выходы	DWORD_OF_BYTE	DWORD	Собранное значение.

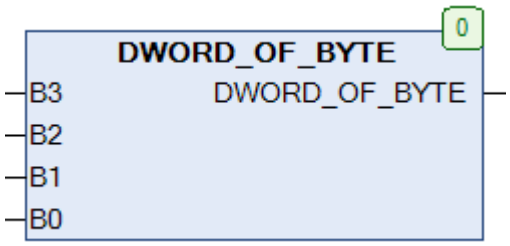


Рис. 16.47. Внешний вид функции **DWORD_OF_BYTE** на языке CFC

Функция **DWORD_OF_BYTE** возвращает значение типа **DWORD,** собранное из отдельных байтов **B3…B0**, где **B0** является младшим байтом.

См. также обратную функцию BYTE_OF_DWORD.

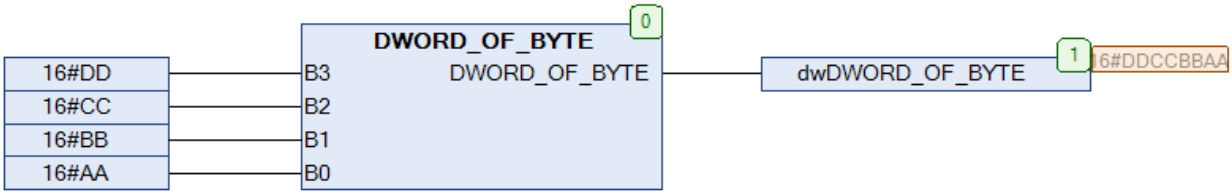


Рис. 16.48. Пример работы с функцией **DWORD_OF_BYTE** на языке CFC

16.26. DWORD_OF_WORD

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	W1…W0	WORD	Исходные слова (W0 – младшее).
Выходы	DWORD_OF_WORD	DWORD	Собранное значение.

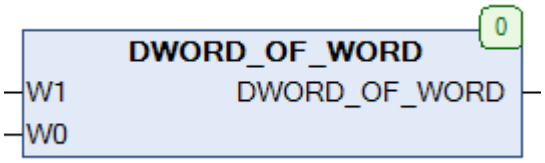


Рис. 16.49. Внешний вид функции **DWORD_OF_WORD** на языке CFC

Функция **DWORD_OF_WORD** возвращает значение типа **DWORD**, собранное из отдельных слов **W1** и **W0**, где **W0** является младшим словом. См. также обратную функцию WORD_OF_DWORD.

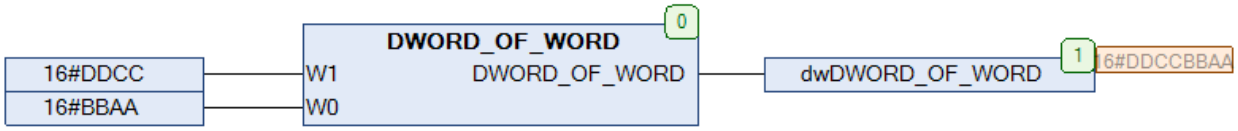


Рис. 16.50. Пример работы с функцией **DWORD_OF_WORD** на языке CFC

16.27. GRAY_TO_BYTE

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	IN	BYTE	Число в коде Грея.
Выходы	GRAY_TO_BYTE	BYTE	Число в двоичном коде.

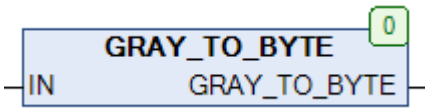


Рис. 16.51. Внешний вид функции **GRAY_TO_BYTE** на языке CFC

Функция **GRAY_TO_BYTE** преобразует число в коде Грея **IN** в число в двоичном коде. См. также обратную функцию BYTE_TO_GRAY.

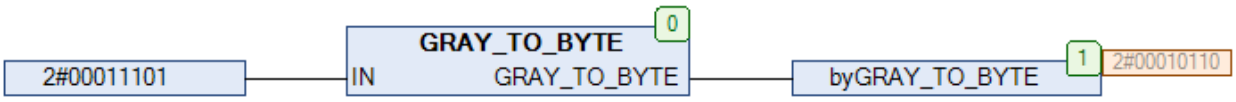


Рис. 16.52. Пример работы с функцией **GRAY_TO_BYTE** на языке CFC

16.28. INT_TO_BCDC

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	IN	INT	Двухзначное число в формате DEC.
Выходы	INT_TO_BCDC	BYTE	Двухзначное число в формате BCD.

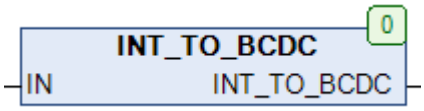


Рис. 16.53. Внешний вид функции **INT_TO_BCDC** на языке CFC

Функция **INT_TO_BCDC** конвертирует двухзначное число из десятичного формата в двоично-десятичный. См. также обратную функцию BCDC_TO_INT.

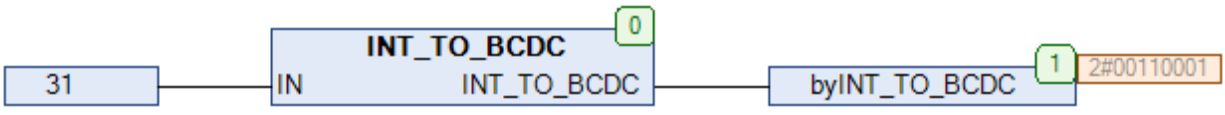


Рис. 16.54. Пример работы с функцией **INT_TO_BCDC** на языке CFC

16.29. MATRIX

Тип модуля: ФБ	Переменная	Тип	Описание
Входы	x1…x5	BOOL	Сигналы строк.
Выходы	code	BYTE	Код нажатия.
	TP	BOOL	Флаг «произошло нажатие».
	y1…y4	BOOL	Сигналы опроса столбцов.
Параметры	Release	BOOL	Обработка отпускания кнопки.

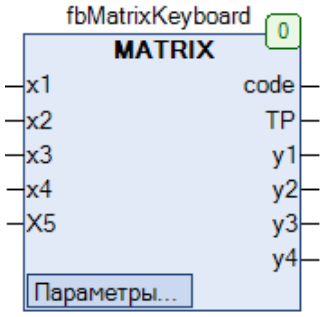


Рис. 16.55. Внешний вид ФБ **MATRIX** на языке CFC

Функциональный блок **MATRIX** используется для опроса матричной клавиатуры размером **5x4**. На входы **x1…5** заводятся сигналы от строк. ФБ последовательно генерирует импульсы на выходах **y1…y4**, которые должны использоваться для опроса соответствующих столбцов. Если один из входов имеет значение **TRUE**, то на выход **code** поступает код нажатой кнопки, содержащий номер ее строки и столбца, а на выходе **TP** генерируется единичный импульс. Если параметр **Release** имеет значение **TRUE**, то отрабатывается также отпускание кнопки.

См. также ФБ PIN_CODE, предназначенный для проверки нажатия заданной кнопки.

Расшифровка битов кода		
Бит 7	**TRUE** – кнопка нажата, **FALSE** – кнопка отпущена.	
Бит 6	Номер столбца (бит 2).	От 0 до 3.
Бит 5	Номер столбца (бит 1).	
Бит 4	Номер столбца (бит 0).	
Бит 3	Всегда **FALSE**.	
Бит 2	Номер строки (бит 2).	От 1 до 5.
Бит 1	Номер строки (бит 1).	
Бит 0	Номер строки (бит 0).	

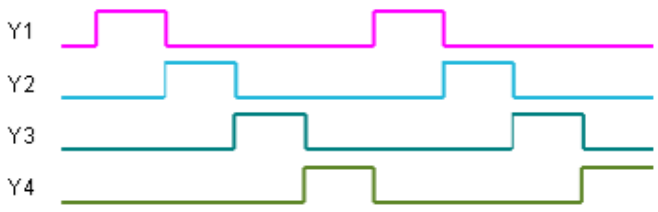


Рис. 16.56. Диаграмма опроса столбцов

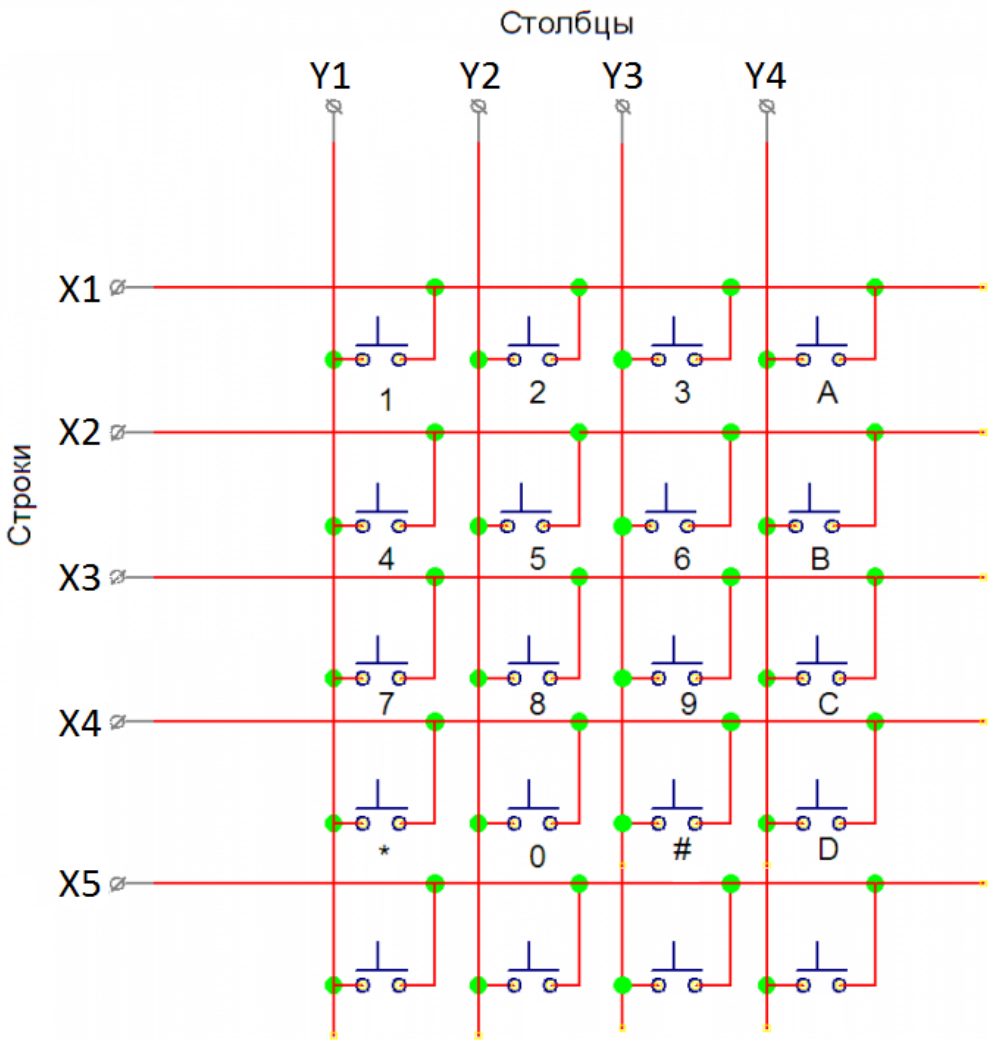


Рис. 16.57. Принципиальная схема матричной клавиатуры

16.30. MUX_2

Тип модуля: функция	Переменная	Тип	Описание
Входы	D0	BOOL	Информационный вход 0.
	D1	BOOL	Информационный вход 1.
	A0	BOOL	Адресный вход.
Выходы	MUX_2	BOOL	Выход мультиплексора.

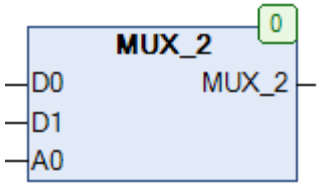


Рис. 16.58. Внешний вид функции **MUX_2** на языке CFC

Функция **MUX_2** реализует двухбитный мультиплексор. Принцип работы функции:

- если A0=FALSE, то MUX_2:=D0;
- если A0=TRUE, то MUX_2:=D1.

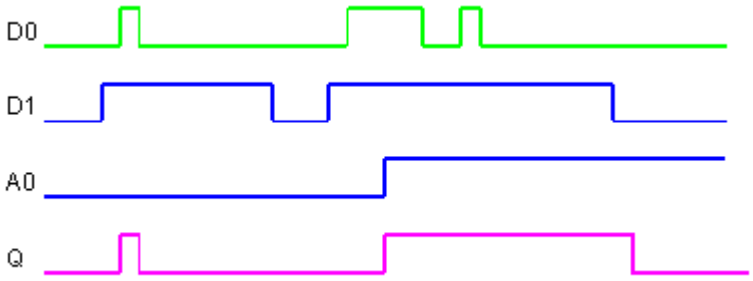


Рис. 16.59. Трассировка работы функции **MUX_2**

16.31. MUX_4

Тип модуля: функция	Переменная	Тип	Описание
Входы	D0	BOOL	Информационный вход 0.
	D1	BOOL	Информационный вход 1.
	D2	BOOL	Информационный вход 2.
	D3	BOOL	Информационный вход 3.
	A0	BOOL	Адресный вход 0.
	A1	BOOL	Адресный вход 1.
Выходы	MUX_4	BOOL	Выход мультиплексора.

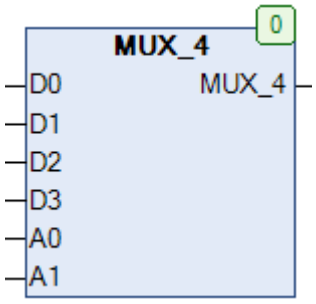


Рис. 16.60. Внешний вид функции **MUX_4** на языке CFC

Функция **MUX_4** реализует четырехбитный мультиплексор. Принцип работы функции:

- если A0=FALSE и A1=FALSE, то MUX_4:=D0;
- если A0=TRUE и A1=FALSE, то MUX_4:=D1;
- если A0=FALSE и A1=TRUE, то MUX_4:=D2;
- если A0=TRUE и A1=TRUE, то MUX_4:=D3.

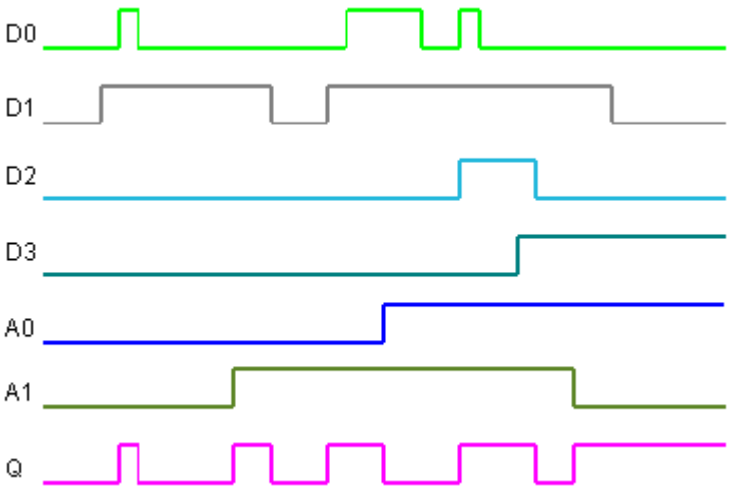


Рис. 16.61. Трассировка работы функции **MUX_4**

16.32. PARITY

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	in	BYTE	Исходный байт.
Выходы	PARITY	BOOL	Бит четности.

Рис. 16.62. Внешний вид функции **PARITY** на языке CFC

Функция **PARITY** вычисляет бит четности для байта **in**. Соответственно, если число битов в состоянии **TRUE** в байте является нечетным, то бит четности = **TRUE**. В противном случае бит четности = **FALSE**. См. также функцию CHECK_PARITY.

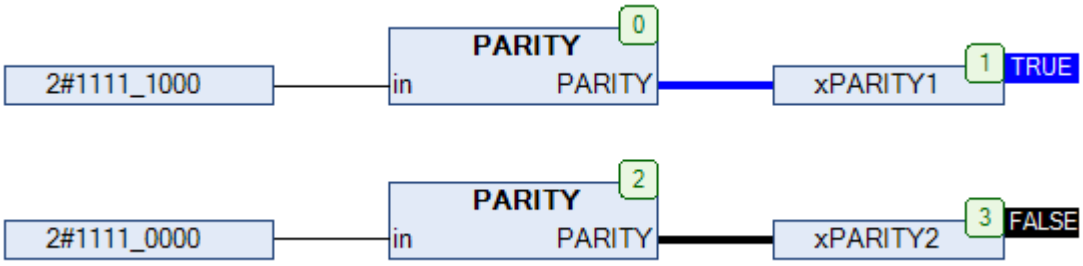


Рис. 16.63. Пример работы с функцией **PARITY** на языке CFC

16.33. PIN_CODE

Тип модуля: ФБ	Переменная	Тип	Описание
Входы	CB	BYTE	Код символа.
	E	BOOL	Сигнал чтения символа.
Выходы	TP	BOOL	Флаг «последовательность введена».
Параметры	PIN	STRING(8)	Контрольная последовательность
Используемые модули	CODE		

Рис. 16.64. Внешний вид ФБ **PIN_CODE** на языке CFC

Функциональный блок **PIN_CODE** используется для декодирования последовательности нажатий матричной клавиатуры, реализованной в ФБ MATRIX. Если вход **E** имеет значение **TRUE**, то код символа **CB** сравнивается с символом контрольной последовательности **PIN**. Если коды совпадают, то внутренний счетчик блока инкрементируется, и при следующем запуске новый символ **CB** будет сравниваться со следующим символом последовательности и т.д. Как только коды не совпадают, счетчик сбрасывается, и следующий введенный код опять будет сравниваться с первым символом последовательности. Если последовательность введенных кодов **CB** соответствует контрольной последовательности **PIN**, то на выходе **TP** генерируется единичный импульс.

Ниже приведен пример совместного использования ФБ MATRIX и **PIN_CODE**. ***Обратите внимание***, что формат кодов контрольной последовательности должен соответствовать формату кодов ФБ **MATRIX** (см. в соответствующем пункте). В данном примере контрольная последовательность – “$81$82$83”, что соответствует кодам кнопок, расположенным в нулевом столбце в первой, второй и третьей строках соответственно.

16#81 = 2#1000_0001

Синим выделен бит нажатия, красным – биты номера столбца нажатой кнопки, зеленым – биты номера строки нажатой кнопки на матричной клавиатуре.

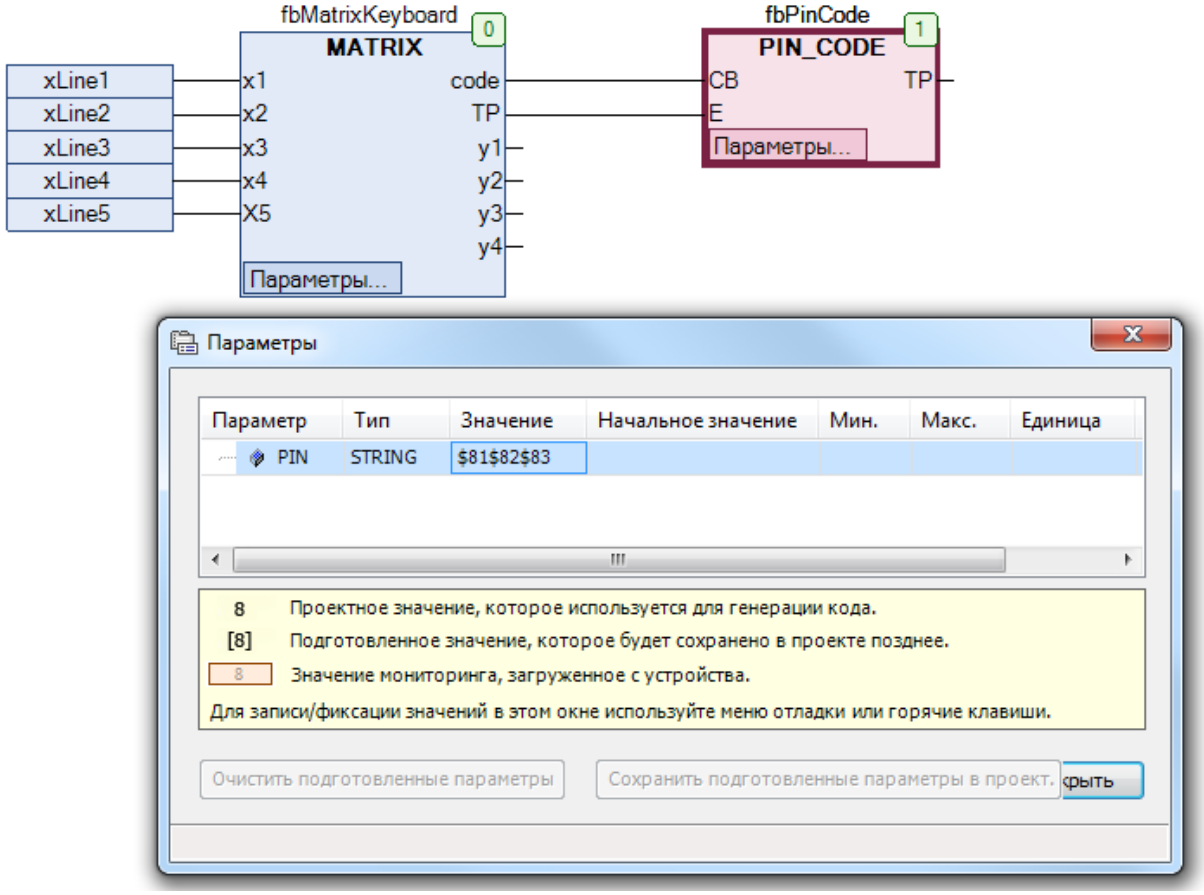


Рис. 16.65. Пример работы с ФБ **PIN_CODE** на языке CFC

16.34. REAL_TO_DW

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	X	REAL	Входное значение.
Выходы	REAL_TO_DW	DWORD	Представление входного значения как DWORD.

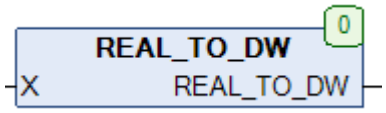


Рис. 16.66. Внешний вид функции **REAL_TO_DW** на языке CFC

Функция **REAL_TO_DW** интерпретирует входную переменную **X** типа **REAL** как значение в шестнадцатеричной системе. В отличие от стандартного оператора конверсии **REAL_TO_DWORD**, который округляет значение с плавающей точкой до ближайшего меньшего целого (например, 5.7--->5), данная функция не меняет исходное значение, а только представляет его в другом формате. В частности, это необходимо при передаче переменных типа **REAL** по протоколу Modbus, который не стандартизирует передаваемые типы данных.

См. также обратную функцию DW_TO_REAL.

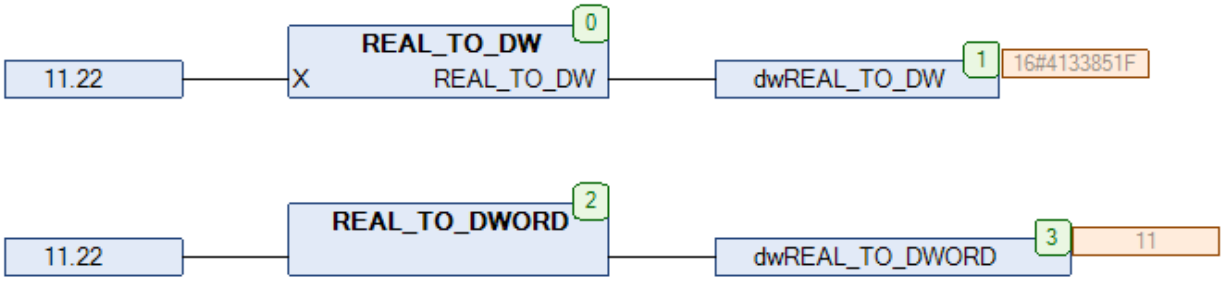


Рис. 16.67. Пример работы с функцией **REAL_TO_DW** на языке CFC

16.35. REFLECT

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	D	DWORD	Строка бит.
	L	INT	Длина отражаемой части.
Выходы	REFLECT	DWORD	Строка бит с обратным порядком бит.

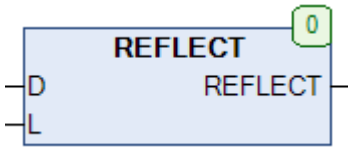


Рис. 16.68. Внешний вид функции **REFLECT** на языке CFC

Функция **REFLECT** меняет порядок **L** младших бит переменной **D** на обратный.

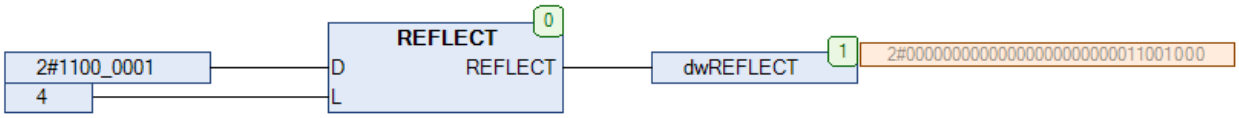


Рис. 16.69. Пример работы с функцией **REFLECT** на языке CFC

16.36. REVERSE

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	IN	BYTE	Строка бит.
Выходы	REVERSE	BYTE	Строка бит с обратным порядком бит.

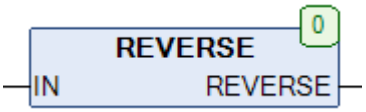


Рис. 16.70. Внешний вид функции **REVERSE** на языке CFC

Функция **REVERSE** меняет порядок бит переменной **IN** на обратный.

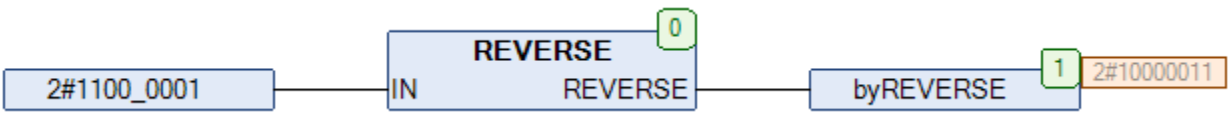


Рис. 16.71. Пример работы с функцией **REVERSE** на языке CFC

16.37. SHL1

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	IN	DWORD	Строка бит.
	N	INT	Кол-во сдвигаемых бит.
Выходы	SHL1	DWORD	Строка бит, сдвинутая влево.

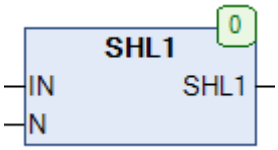


Рис. 16.72. Внешний вид функции **SHL1** на языке CFC

Функция **SHL1** выполняет побитный сдвиг переменной **IN** влево на **N** бит с дополнением единицами справа (в отличие от стандартного оператора **SHL**, который производит дополнение нулями).

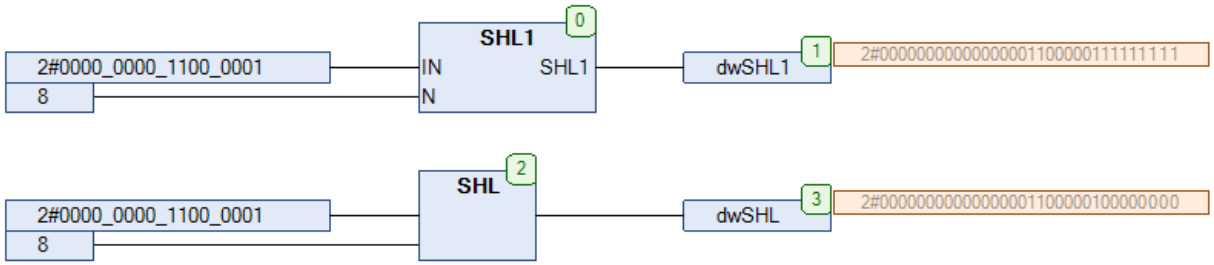


Рис. 16.73. Пример работы с функцией **SHL1** на языке CFC

16.38. SHR1

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	IN	DWORD	Строка бит.
	N	INT	Кол-во сдвигаемых бит.
Выходы	SHR1	DWORD	Строка бит, сдвинутая вправо.

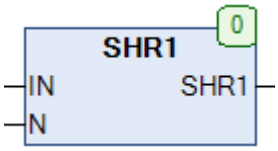


Рис. 16.72. Внешний вид функции **SHR1** на языке CFC

Функция **SHR1** выполняет побитный сдвиг переменной **IN** вправо на **N** бит с дополнением единицами слева (в отличие от стандартного оператора **SHL**, который производит дополнение нулями).

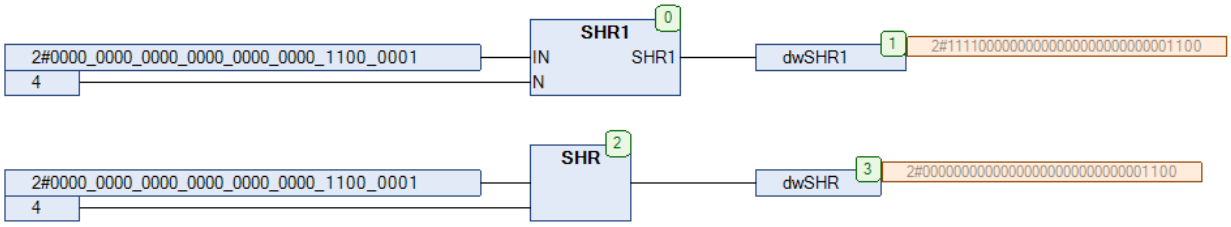


Рис. 16.73. Пример работы с функцией **SHR1** на языке CFC

16.39. SWAP_BYTE

Тип модуля: функция	Переменная	Тип	Описание
Входы	IN	WORD	Исходная переменная.
Выходы	SWAP_BYTE	WORD	Переменная с переставленными байтами.

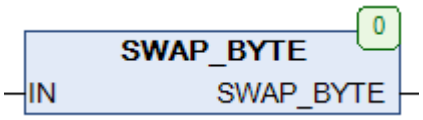


Рис. 16.74. Внешний вид функции **SWAP_BYTE** на языке CFC

Функция *SWAP_BYTE* меняет порядок байт переменной **IN**.

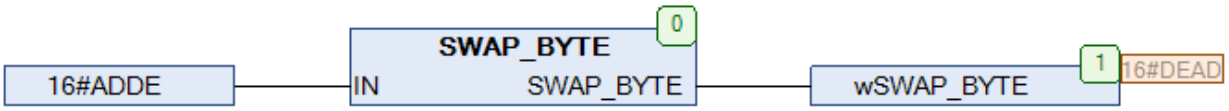


Рис. 16.75. Пример работы с функцией **SWAP_BYTE** на языке CFC

16.40. SWAP_BYTE2

Тип модуля: функция	Переменная	Тип	Описание
Входы	IN	DWORD	Исходная переменная.
Выходы	SWAP_BYTE2	DWORD	Переменная с обратным порядком байт.

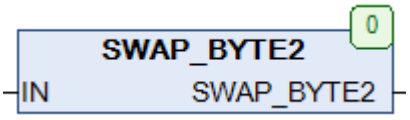


Рис. 16.76. Внешний вид функции **SWAP_BYTE2** на языке CFC

Функция **SWAP_BYTE2** меняет порядок байт переменной **IN** на обратный.

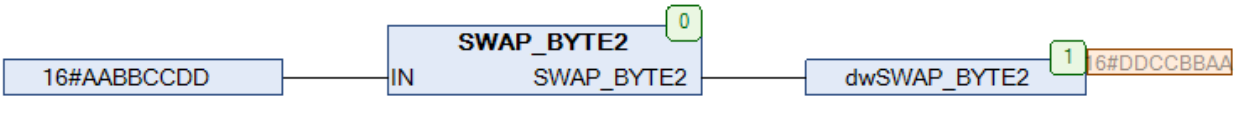


Рис. 16.77. Пример работы с функцией **SWAP_BYTE2** на языке CFC

16.41. WORD_OF_BYTE

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	B1…B0	BYTE	Исходные байты (B0 – младший).
Выходы	WORD_OF_BYTE	WORD	Собранное значение.

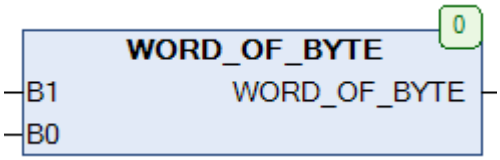


Рис. 16.78. Внешний вид функции **WORD_OF_BYTE** на языке CFC

Функция **WORD_OF_BYTE** возвращает значение типа **WORD**, собранное из отдельных байтов **B1…B0**, где **B0** является младшим байтом.

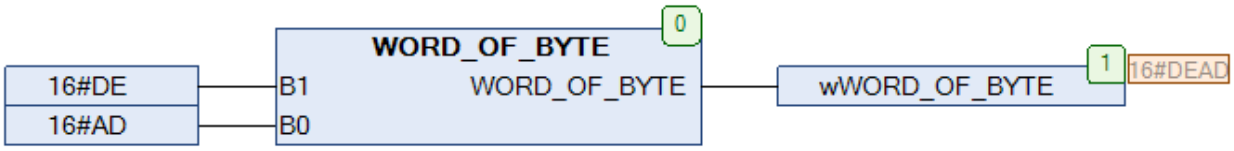


Рис. 16.79. Пример работы с функцией **WORD_OF_BYTE** на языке CFC

16.42. WORD_OF_DWORD

Тип модуля: функция	Переменная	Тип	Описание
Входы	in	DWORD	Исходная переменная.
	N	BYTE	Номер считываемого слова (0…1).
Выходы	WORD_OF_DWORD	WORD	Выделенное слово.

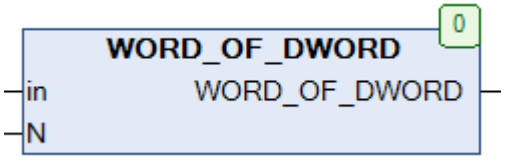


Рис. 16.80. Внешний вид функции **WORD_OF_DWORD** на языке CFC

Функция **WORD_OF_DWORD** выделяет из переменной **in** типа **DWORD** слово с номером **N**. См. также обратную функцию DWORD_OF_WORD.

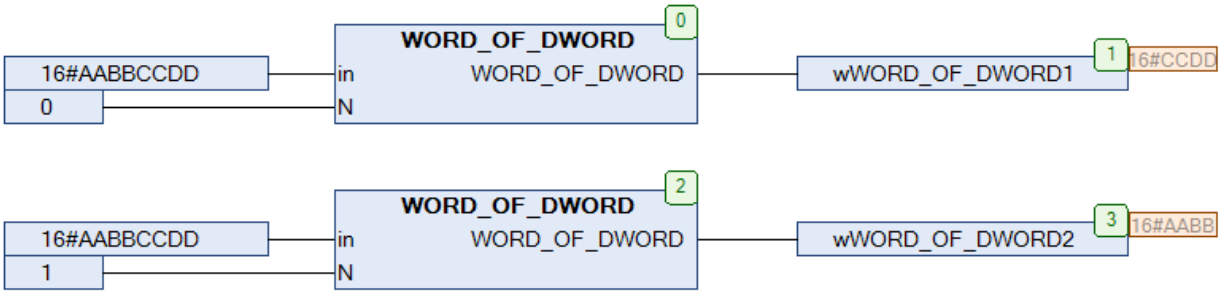


Рис. 16.81. Пример работы с функцией WORD_OF_DWORD на языке CFC

17. Триггеры, элементы хранения и регистры сдвига

17.1. COUNT_BR

Тип модуля: ФБ	**Переменная**	**Тип**	**Описание**
Входы	SET	BOOL	Сигнал установки начального значения.
	in	BYTE	Начальное значение.
	UP	BOOL	Импульс «добавить»
	DN	BOOL	Импульс «отнять».
	STEP	BYTE	Шаг счетчика.
	MX	BYTE	Максимальное значение счетчика.
	RST	BOOL	Сигнал сброса блока.
Выходы	CNT	BYTE	Текущее значение счетчика.
Используемые модули	INC		

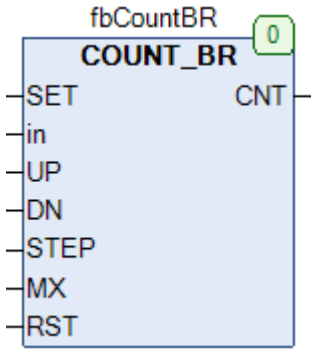


Рис. 17.1. Внешний вид ФБ **COUNT_BR** на языке CFC

Функциональный блок **COUNT_BR** представляет собой счетчик импульсов. По переднему фронту входа **SET** на выход счетчика **CNT** устанавливается начальное значение **in**. По переднему фронту входов **UP/DN** происходит увеличение/уменьшение счетчика на величину **STEP**. Вход **MX** определяет максимальное значение счетчика. При достижении этого значения счетчик переходит через ноль. Иными словами, при значениях in=0, STEP=100 и MX=255 во время счета в положительном направлении счетчик будет принимать следующие значения: 0, 100, 200, 44, 144, 244, 88 и т.д. По переднему фронту входа **RST** происходит сброс счетчика в **0** в независимости от состояния остальных входов.

На рис. 17.2 приведен пример работы с счетчиком. По переднему фронту переменной **xTrig** происходит срабатывание счетчика. Переменная **xMode** определяет направление счета: **TRUE** – положительное, **FALSE** – отрицательное.

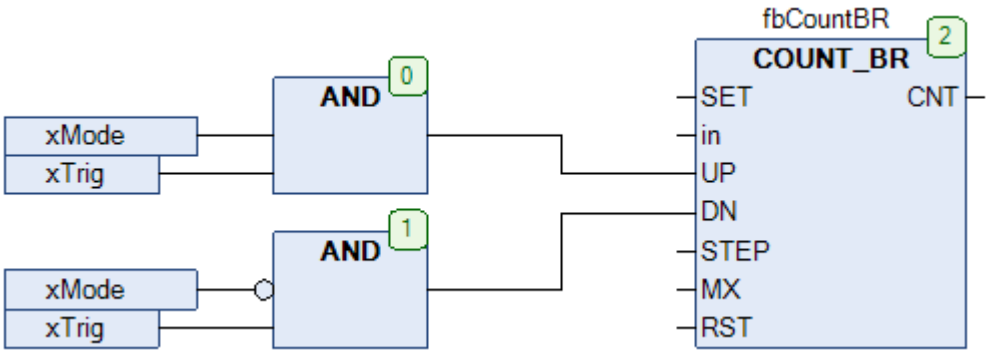


Рис. 17.2. Пример работы с ФБ **COUNT_BR** на языке CFC

17.2. COUNT_DR

Тип модуля: ФБ	**Переменная**	**Тип**	**Описание**
Входы	SET	BOOL	Сигнал установки начального значения.
	in	DWORD	Начальное значение.
	UP	BOOL	Импульс «добавить»
	DN	BOOL	Импульс «отнять».
	STEP	DWORD	Шаг счетчика.
	MX	DWORD	Максимальное значение счетчика.
	RST	BOOL	Сигнал сброса блока.
Выходы	CNT	DWORD	Текущее значение счетчика.

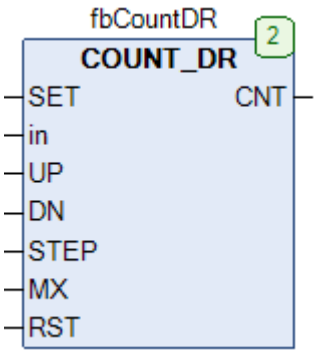


Рис. 17.3. Внешний вид ФБ **COUNT_DR** на языке CFC

Функциональный блок **COUNT_DR** представляет собой счетчик импульсов. Принцип работы полностью соответствует ФБ COUNT_BR, единственным отличием является тип используемых переменных – **DWORD**.

17.3. FF_D2E

Тип модуля: ФБ	Переменная	Тип	Описание
Входы	D0	BOOL	Контролируемое значение 0.
	D1	BOOL	Контролируемое значение 1.
	CLK	BOOL	Сигнал сохранения значений.
	RST	BOOL	Сигнал сброса блока.
Выходы	Q0	BOOL	Сохраненное значение 0.
	Q1	BOOL	Сохраненное значение 1.

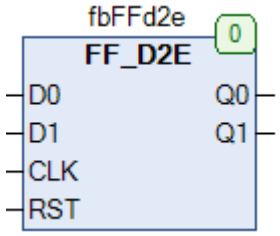


Рис. 17.4. Внешний вид ФБ **FF_D2E** на языке CFC

Функциональный блок **FF_D2E** представляет собой двухбитный D-триггер. По переднему фронту входа **CLK** значения входов **D0** и **D1** присваиваются выходам **Q0** и **Q1**. По переднему фронту входа **RST** значения выходов сбрасываются в **FALSE** в независимости от состояния остальных входов.

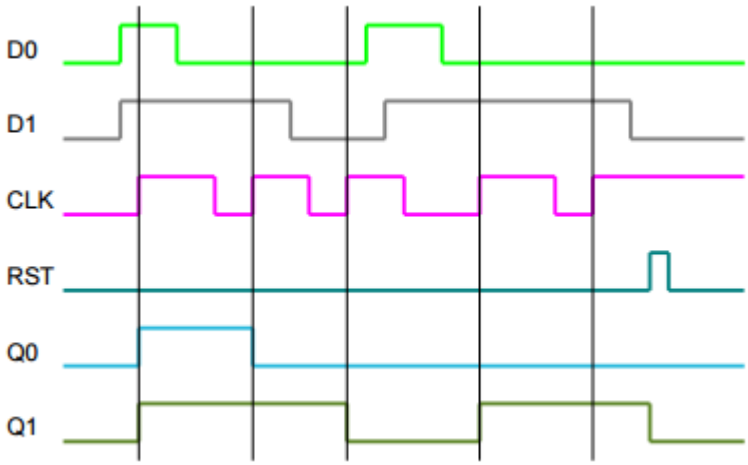


Рис. 17.5. Трассировка работы ФБ **FF_D2E**

17.4. FF_D4E

Тип модуля: ФБ	**Переменная**	**Тип**	**Описание**
Входы	D0	BOOL	Контролируемое значение 0.
	D1	BOOL	Контролируемое значение 1.
	D2	BOOL	Контролируемое значение 2.
	D3	BOOL	Контролируемое значение 3.
	CLK	BOOL	Управляющий сигнал.
	RST	BOOL	Сигнал сброса блока.
Выходы	Q0	BOOL	Сохраненное значение 0.
	Q1	BOOL	Сохраненное значение 1.
	Q2	BOOL	Сохраненное значение 2.
	Q3	BOOL	Сохраненное значение 3.

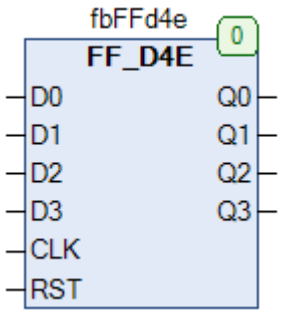


Рис. 17.6. Внешний вид ФБ **FF_D4E** на языке CFC

Функциональный блок **FF_D4E** представляет собой четырехбитный D-триггер. По переднему фронту входа **CLK** значения входов **D0, D1, D2** и **D3** присваиваются выходам **Q0, Q1, Q2** и **Q3** соответственно. По переднему фронту входа **RST** значения выходов сбрасываются в **FALSE** в независимости от состояния остальных входов. См. также описание ФБ FF_D2E.

17.5. FF_DRE

Тип модуля: ФБ	**Переменная**	**Тип**	**Описание**
Входы	SET	BOOL	Сигнал активации выхода.
	D	BOOL	Контролируемое значение.
	CLK	BOOL	Управляющий сигнал.
	RST	BOOL	Сигнал сброса блока.
Выходы	Q	BOOL	Сохраненное значение.

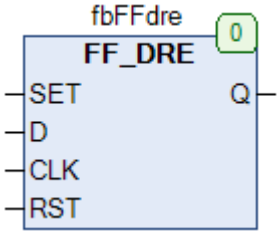


Рис. 17.7. Внешний вид ФБ **FF_DRE** на языке CFC

Функциональный блок **FF_DRE** представляет собой D-триггер с асинхронным управлением выходом. По переднему фронту входа **CLK** значения входа **D** присваивается выходу **Q**. По переднему фронту входа **SET** выход принимает значение **TRUE**. По переднему фронту входа **RST** выход сбрасывается в **FALSE** в независимости от состояния остальных входов. Вход **RST** имеет приоритет над входом **SET**, и при их одновременном срабатывании сигнал на входе **SET** не обрабатывается. Вход **SET** имеет приоритет над входом **CLK**.

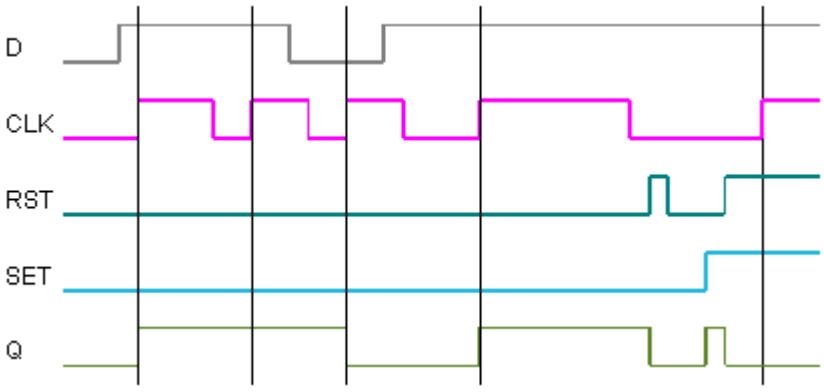


Рис. 17.8. Трассировка работы ФБ **FF_DRE**

17.6. FF_JKE

Тип модуля: ФБ	Переменная	Тип	Описание
Входы	SET	BOOL	Сигнал установки выхода.
	J	BOOL	Вход «Jump».
	CLK	BOOL	Управляющий сигнал.
	K	BOOL	Вход «Kill».
	RST	BOOL	Сигнал сброса блока.
Выходы	Q	BOOL	Выход триггера.

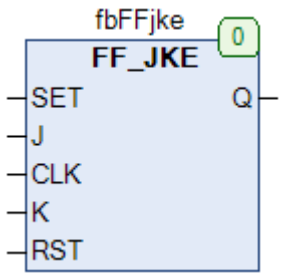


Рис. 17.9. Внешний вид ФБ **FF_JKE** на языке CFC

Функциональный блок **FF_JKE** представляет собой JK-триггер. По переднему фронту входа **SET** выход принимает значение **TRUE**. По переднему фронту входа **RST** выход сбрасывается в **FALSE** в независимости от состояния остальных входов. По переднему фронту входа **CLK** выход Q меняет свое значение в зависимости от состояний входов **J** и **K**:

- если J=TRUE, то Q:=TRUE
- если K=TRUE, то Q:=FALSE
- если J=TRUE и K=TRUE, то Q:=NOT(Q) [значение выхода инвертируется]

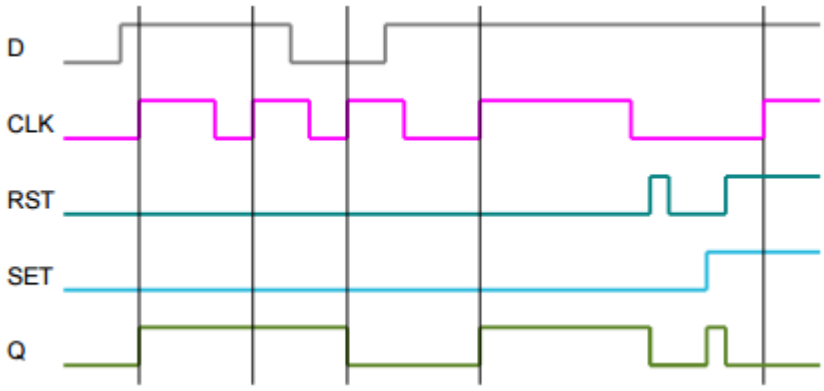


Рис. 17.10. Трассировка работы ФБ **FF_JKE**

Вход **SET** имеет приоритет над входом **CLK**.

17.7. FF_RSE

Тип модуля: ФБ	**Переменная**	**Тип**	**Описание**
Входы	CS	BOOL	Вход «Set».
	CR	BOOL	Вход «Reset».
	RST	BOOL	Сигнал сброса блока.
Выходы	Q	BOOL	Выход триггера.

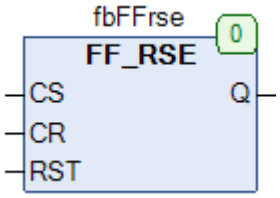


Рис. 17.11. Внешний вид ФБ **FF_RSE** на языке CFC

Функциональный блок **FF_RSE** представляет собой RS-триггер с асинхронным сбросом. По переднему фронту входа **CS** выход **Q** принимает значение **TRUE**. По переднему фронту входа **CR** выход **Q** принимает значение **FALSE**. Вход **CR** имеет приоритет над входом **CS**, и при их одновременном срабатывании сигнал на входе **CS** не обрабатывается. По переднему фронту входа **RST** выход сбрасывается в **FALSE** в независимости от состояния остальных входов.

17.8. LTCH

Тип модуля: ФБ	**Переменная**	**Тип**	**Описание**
Входы	D	BOOL	Контролируемое значение.
	L	BOOL	Режим «сохранение значений».
	RST	BOOL	Сигнал сброса блока.
Выходы	Q	BOOL	Сохраненное значение.

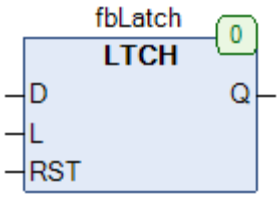


Рис. 17.12. Внешний вид ФБ **LTCH** на языке CFC

Функциональный блок **LTCH** представляет собой одноступенчатый триггер («защелку»). Пока вход **L** имеет значение **TRUE**, значение на выходе **Q** равно значению входа **D**. По заднему фронту входа **L** выход **Q** фиксируется в своем текущем состоянии. По переднему фронту входа **RST** выход сбрасывается в **FALSE** в независимости от состояния остальных входов.

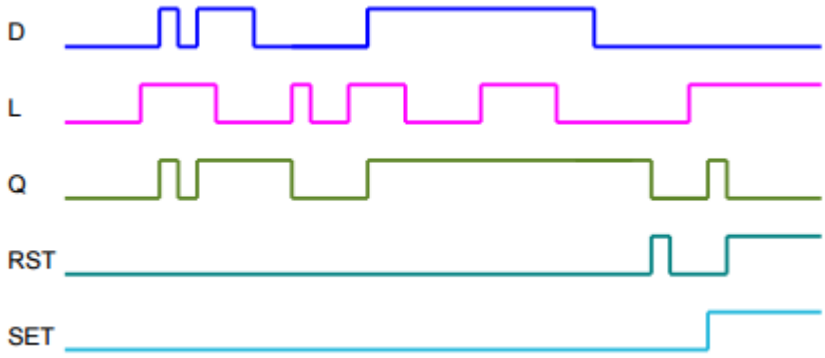


Рис. 17.13. Трассировка работы ФБ **LTCH**

17.9. LTCH_4

Тип модуля: ФБ	**Переменная**	**Тип**	**Описание**
Входы	D0…D3	BOOL	Контролируемые значения.
	L	BOOL	Режим «сохранение значений».
	RST	BOOL	Сигнал сброса блока.
Выходы	Q	BOOL	Сохраненные значения.

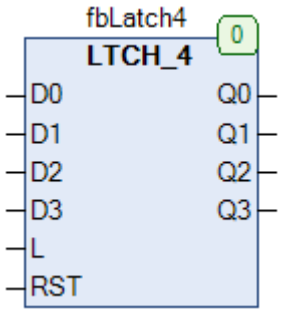


Рис. 17.12. Внешний вид ФБ **LTCH_4** на языке CFC

Функциональный блок **LTCH_4** представляет собой четырехбитный одноступенчатый триггер («защелку»). Пока вход **L** имеет значение **TRUE**, значения на выходах **Q0...Q3** равны значениям входов **D0...D3**. По заднему фронту входа **L** выходы фиксируется в своих текущих состояниях. По переднему фронту входа **RST** выходы сбрасываются в **FALSE** в независимости от состояния остальных входов.

17.10. SELECT_8

Тип модуля: ФБ	**Переменная**	**Тип**	**Описание**
Входы	E	BOOL	Вход управления блоком.
	SET	BOOL	Сигнал установки выхода номер IN.
	IN	BYTE	Номер устанавливаемого выхода.
	UP	BOOL	Включение следующего выхода.
	DN	BOOL	Включение предыдущего выхода.
	RST	BOOL	Сигнал сброса блока.
Выходы	Q0…Q7	BOOL	Выходы блока.
	STATE	INT	Номер активного выхода (0...7).
Используемые модули	INC		

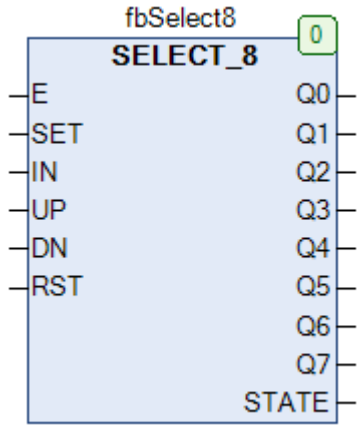


Рис. 17.13. Внешний вид ФБ **SELECT_8** на языке CFC

Функциональный блок **SELECT_8** представляет собой восьмибитный селектор - в каждый момент времени только один из выходов **Q0…Q7** является активным. Пока вход **E** имеет значение **TRUE** блок находится в работе. По переднему фронту на входе **SET** выход с номером **IN** принимает значение **TRUE**, а все остальные выходы сбрасываются в **FALSE**. По переднему фронту на входе **UP** активным становится следующий выход, по переднему фронту на входе **DN** – предыдущий. Соответственно, если активным является **Q7**, то после импульса на входе **UP** активным становится **Q0**; если же активным является **Q0**, то после импульса на входе **DN** активным становится **Q7**. Выход **STATE** содержит номер активного выхода. По переднему фронту входа **RST** выход сбрасывается в **FALSE** в независимости от состояния остальных входов.

17.11. SHR_4E

Тип модуля: ФБ	Переменная	Тип	Описание
Входы	SET	BOOL	Сигнал активации выходов.
	D0	BOOL	Вход данных.
	CLK	BOOL	Управляющий сигнал.
	RST	BOOL	Сигнал сброса блока.
Выходы	Q0…Q3	BOOL	Выходы блока.

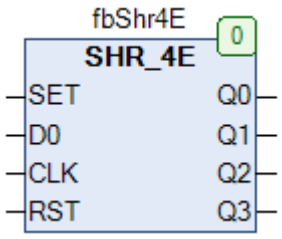


Рис. 17.14. Внешний вид ФБ **SHR_4E** на языке CFC

Функциональный блок **SHR4_4E** представляет собой четырехбитный регистр сдвига. По переднему фронту входа **SET** выходы **Q0…Q3** принимают значение **TRUE**. По переднему фронту входа **RST** выходы **Q0...Q3** сбрасываются в **FALSE** в независимости от состояния остальных входов. По переднему фронту входа **CLK** значения на выходах **Q0…Q3** сдвигаются по следующему алгоритму:

- Q2---- > Q3
- Q1---- > Q2
- Q0---- > Q1
- D0---- > Q0

Вход **SET** имеет приоритет над входом **CLK**.

Обратите внимание, что сдвиг не является циклическим.

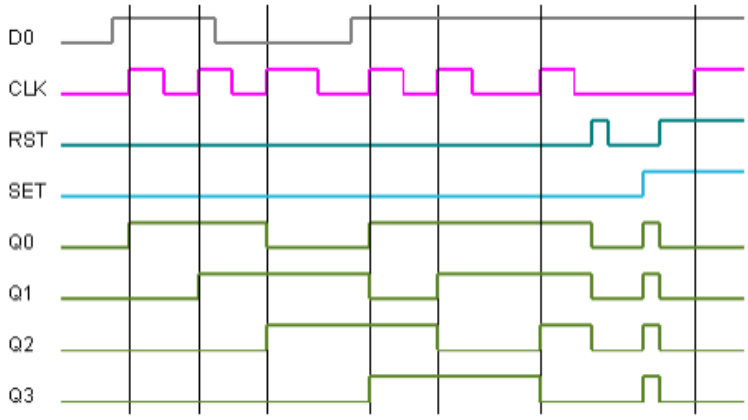


Рис. 17.15. Трассировка работы ФБ **SHR_4E**

17.12. SHR_4UDE

Тип модуля: ФБ	**Переменная**	**Тип**	**Описание**
Входы	SET	BOOL	Сигнал активации выходов.
	D0	BOOL	Вход данных (при счете вверх).
	D3	BOOL	Вход данных (при счете вниз).
	CLK	BOOL	Управляющий сигнал.
	DN	BOOL	Направление счета (TRUE – вниз, FALSE – вверх).
	RST	BOOL	Сигнал сброса блока.
Выходы	Q0…Q3	BOOL	Выходы блока.

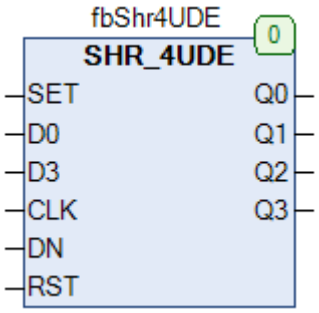


Рис. 17.16. Внешний вид ФБ **SHR_4UDE** на языке CFC

Функциональный блок **SHR4_4UDE** представляет собой четырехбитный регистр сдвига с выбором направления сдвига. По переднему фронту входа **SET** выходы **Q0…Q3** принимают значение **TRUE**. По переднему фронту входа **RST** выходы **Q0...Q3** сбрасываются в **FALSE** в независимости от состояния остальных входов. По переднему фронту входа **CLK** значения на выходах **Q0…Q3** сдвигаются по алгоритму, определяемому состоянием входа **DN**:

- DN=FALSE (сдвиг вверх):
  - Q2---- > Q3
  - Q1---- > Q2
  - Q0---- > Q1
  - D0---- > Q0
- DN=TRUE (сдвиг вниз):
  - Q1---- > Q0
  - Q2---- > Q1
  - Q3---- > Q2
  - D3---- > Q3

Вход **SET** имеет приоритет над входом **CLK**.

Обратите внимание, что сдвиг не является циклическим.

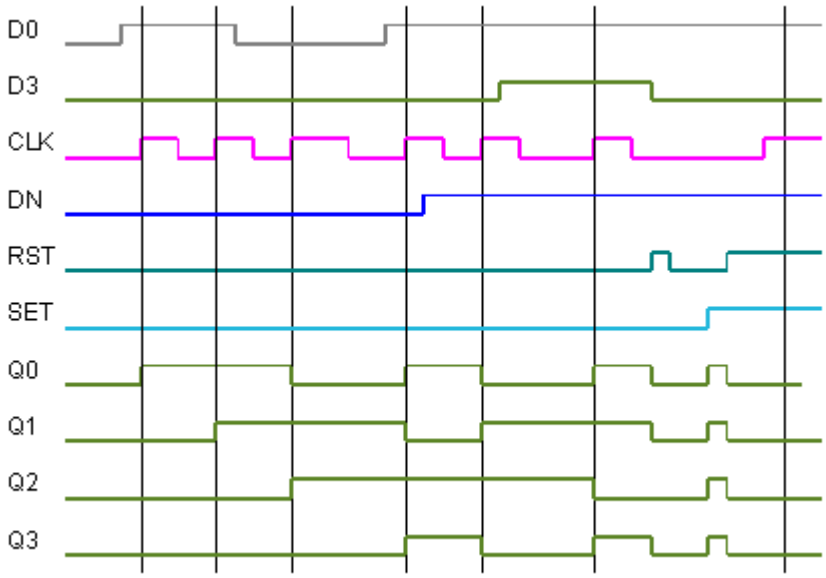


Рис. 17.17. Трассировка работы ФБ **SHR_4UDE**

17.13. SHR_8PLE

Тип модуля: ФБ	**Переменная**	**Тип**	**Описание**
Входы	Din	BOOL	Значение для записи.
	Dload	BYTE	Вход данных.
	CLK	BOOL	Управляющий сигнал.
	UP	BOOL	Направление счета (TRUE – вверх, FALSE – вниз)
	load	BOOL	Режим загрузки данных в блок.
	RST	BOOL	Сигнал сброса блока.
Выходы	DOut	BOOL	Выход блока.

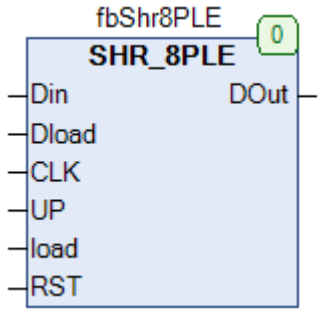


Рис. 17.18. Внешний вид ФБ **SHR_8PLE** на языке CFC

Функциональный блок **SHR4_8PLE** представляет собой восьмибитный регистр сдвига с выбором направления сдвига. Вход **UP** определяет направление сдвига: **TRUE** – вверх (при этом выход **DOut** соответствует старшему биту регистра), **FALSE** – вниз (при этом выход **DOut** соответствует младшему биту регистра). По переднему фронту на входе **CLK** происходит сдвиг регистра в соответствующем направлении, при этом младший (при **UP=TRUE**) или старший (при **UP=FALSE**) бит принимает значение **Din**. Если вход **load** имеет значение **TRUE**, то по переднему фронту на входе **CLK** происходит запись значения **Dload** в регистр.

Обратите внимание, что сдвиг не является циклическим.

17.14. SHR_8UDE

Тип модуля: ФБ	**Переменная**	**Тип**	**Описание**
Входы	SET	BOOL	Сигнал активации выходов.
	D0	BOOL	Вход данных (при счете вверх).
	D7	BOOL	Вход данных (при счете вниз).
	CLK	BOOL	Управляющий сигнал.
	DN	BOOL	Направление счета (TRUE – вниз, FALSE – вверх).
	RST	BOOL	Сигнал сброса блока.
Выходы	Q0…Q7	BOOL	Выходы блока.

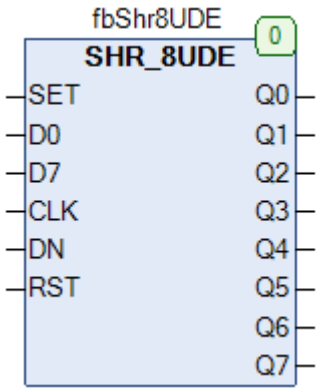


Рис. 17.19. Внешний вид ФБ **SHR_8UDE** на языке CFC

Функциональный блок **SHR4_8UDE** представляет собой восьмибитный регистр сдвига с выбором направления сдвига. Принцип работы блока полностью соответствует ФБ SHR_4UDE.

17.15. STORE_8

Тип модуля: ФБ	**Переменная**	**Тип**	**Описание**
Входы	SET	BOOL	Сигнал активации выходов.
	D0…D7	BOOL	Вход данных (при счете вверх).
	Clr	BOOL	Сигнал сброса выхода с наименьшим номером.
	RST	BOOL	Сигнал сброса блока.
Выходы	Q0…Q7	BOOL	Выходы блока.

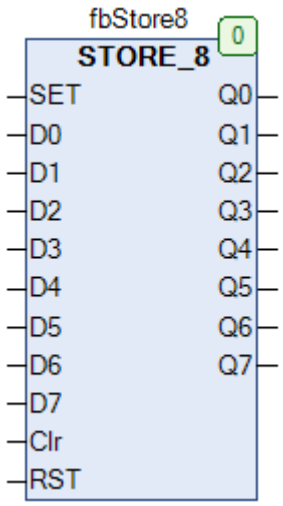


Рис. 17.20. Внешний вид ФБ **STORE_8** на языке CFC

Функциональный блок **STORE_8** представляет собой восьмибитный элемент хранения. По переднему фронту входов **D0…D7** выходы **Q0…Q7** принимают значение **TRUE**. Пока вход **SET** имеет значение **TRUE**, выходы **Q0…Q7** принимают значение **TRUE** в независимости от состояния входов. По переднему фронту входа **RST** выходы **Q0...Q7** сбрасываются в **FALSE**. По переднему фронту входа **Clr** происходит сброс в **FALSE** активного выхода с наименьшим номером (только в том случае, если соответствующий вход имеет значение **FALSE**).

17.16. TOGGLE

Тип модуля: ФБ	**Переменная**	**Тип**	**Описание**
Входы	CLK	BOOL	Управляющий сигнал.
	RST	BOOL	Сигнал сброса блока.
Выходы	Q	BOOL	Выход блока.

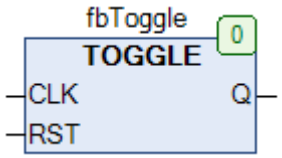


Рис. 17.20. Внешний вид ФБ **TOGGLE** на языке CFC

Функциональный блок **TOGGLE** представляет собой переключатель со сбросом. По переднему фронту входа **CLK** значение выхода **Q** инвертируется. По переднему фронту входа **RST** выход **Q** принимает значение **FALSE** в независимости от состояния входа **CLK**.

18. Генераторы сигналов

18.1. _RMP_B

Тип модуля: ФБ	Переменная	Тип	Описание
Входы	DIR	BOOL	Направление изменения выхода.
	E	BOOL	Сигнал управления блоком.
	TR	TIME	Полное время генерации (от 0 до 255).
Входы-выходы	RMP	BYTE	Выход блока.
Используемые модули	T_PLC_MS, FRMP_B		

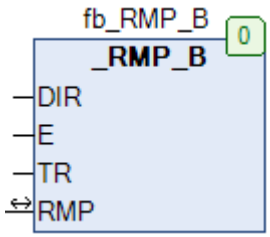


Рис. 18.1. Внешний вид ФБ **_RMP_B** на языке CFC

Функциональный блок **_RMP_B** представляет собой генератор линейной функции. Пока вход **E** имеет значение **TRUE** блок находится в работе и значение переменной, связанной со входом-выходом **RMP**, линейно изменяется от текущего значения до максимального (при **DIR=TRUE**) или минимального (при **DIR=FALSE**). Минимальное и максимальное значения определяются диапазоном типа **BYTE** и соответственно составляют **0** и **255**. Скорость изменения выходного сигнала определяется отношением (**255/TR**). Для повторного запуска блока после окончания его работы достаточно изменить значение **RMP** (например, обнулив его). Если вход **E** принимает значение **FALSE**, то блок прекращает работу, при этом **RMP** сохраняет свое последнее значение.

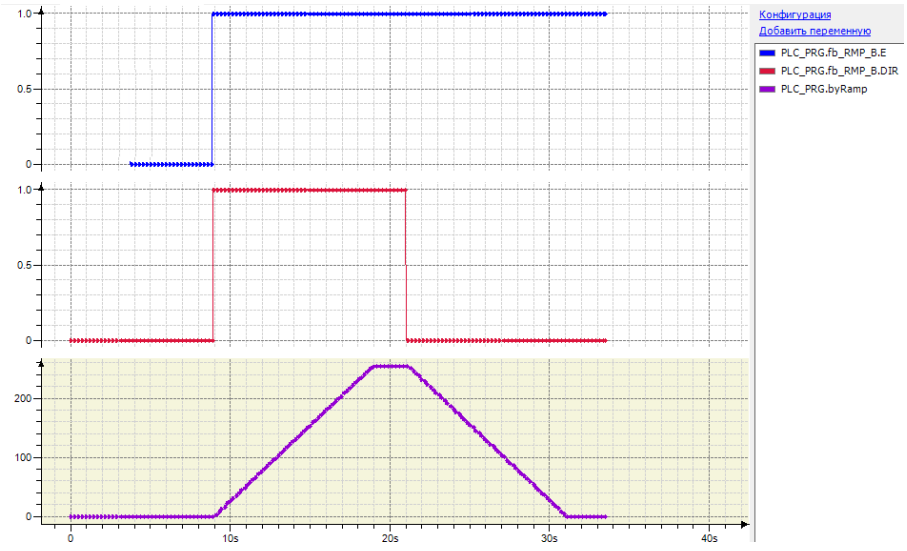


Рис. 18.2. Трассировка работы ФБ **_RMP_B** (TR=T#10s)

18.2. _RMP_NEXT

Тип модуля: ФБ	**Переменная**	**Тип**	**Описание**
Входы	E	BOOL	Сигнал управления блоком.
	IN	BYTE	Контролируемый сигнал.
	TR	TIME	Время нарастания сигнала.
	TF	TIME	Время спада сигнала.
	TL	TIME	Время задержки.
Выходы	DIR	BOOL	Флаг «направление изменения».
	UP	BOOL	Импульс «начало нарастания».
	DN	BOOL	Импульс «начало спада».
Входы-выходы	OUT	BYTE	Выход блока.
Используемые модули	_RMP_B, TREND_DW		

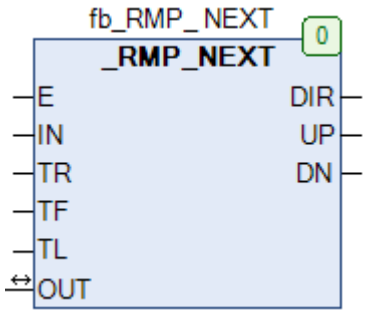


Рис. 18.3. Внешний вид ФБ **_RMP_NEXT** на языке CFC

Функциональный блок **_RMP_NEXT** представляет собой генератор линейной функции. Пока вход **E** имеет значение **TRUE** блок находится в работе и значение переменной, связанной со входом-выходом **OUT**, линейно изменяется от текущего значения до ближайшего значения относительно входа **IN**. Время достижения этого значения определяется переменными **TR** (в случае нарастания сигнала) и **TF** (в случае спада сигнала). Время **TL** определяет задержку изменения выходного сигнала относительно изменения входного. Выход **DIR** определяет текущее изменение выхода (**TRUE** – нарастание, **FALSE** – спад); если изменений не происходит, то **DIR** сохраняет свое последнее значение. Выход **UP** принимает значение **TRUE** на время нарастания выходного сигнала, выход **DN** – на время его спада. В остальные периоды времени выходы имеют значение **FALSE**. Если вход **E** принимает значение **FALSE**, то блок прекращает работу, при этом **OUT** сохраняет свое последнее значение.

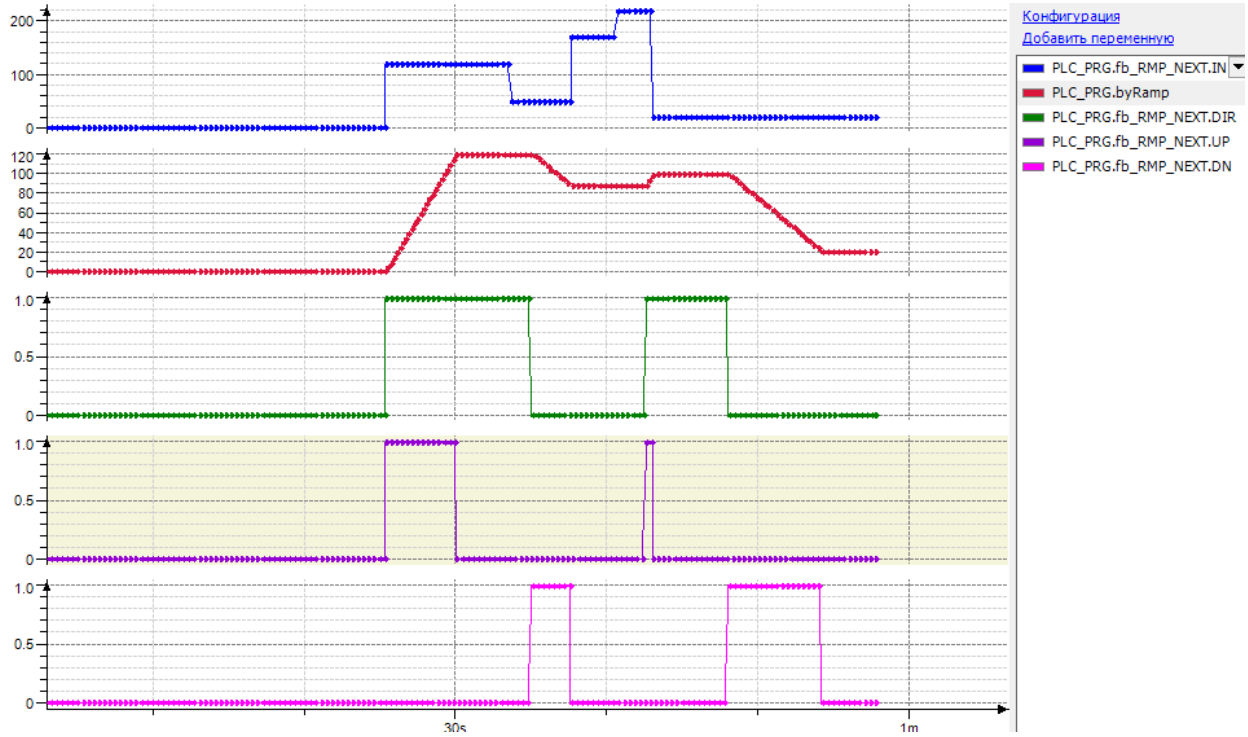


Рис. 18.4. Трассировка работы ФБ **_RMP_NEXT** (TR=T#10s, TF=T#20s, TL=T#5s)

18.3. _RMP_W

Тип модуля: ФБ	**Переменная**	**Тип**	**Описание**
Входы	DIR	BOOL	Направление изменения выхода.
	E	BOOL	Сигнал управления блоком.
	TR	TIME	Полное время генерации (от 0 до 65535).
Входы-выходы	RMP	WORD	Выход блока.
Используемые модули	T_PLC_MS		

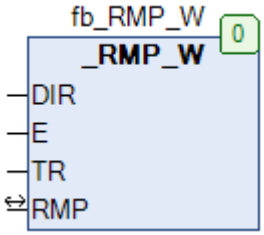


Рис. 18.5. Внешний вид ФБ **_RMP_W** на языке CFC

Функциональный блок **_RMP_W** представляет собой генератор линейной функции. Принцип работы блока полностью соответствует блоку _RMP_B, единственным отличие является тип выхода генератора – у данного блока он имеет тип **WORD**.

18.4. GEN_PULSE

Тип модуля: ФБ	**Переменная**	**Тип**	**Описание**
Входы	ENQ	BOOL	Сигнал управления блоком.
	PTH	TIME	Время сигнала верхнего уровня.
	PTL	TIME	Время сигнала нижнего уровня.
Выходы	Q	BOOL	Выход генератора.
Используемые модули	T_PLC_MS		

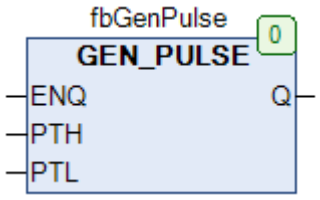


Рис. 18.6. Внешний вид ФБ **GEN_PULSE** на языке CFC

Функциональный блок **GEN_PULSE** представляет собой генератор прямоугольных импульсов. Когда вход **EHQ** принимает значение **TRUE**, блок запускается в работу, и на выходе **Q** начинают генерироваться прямоугольные импульсы с длительностью **PTL** (нижний уровень) и **PTH** (верхний уровень) соответственно. Блок начинает работу с импульса нижнего уровня. Если вход **ENQ** принимает значение **FALSE**, то блок прекращает работу; выход **Q** при этом принимает значение **FALSE**. Импульс с длительностью **0** соответствует импульсу длиной в цикл ПЛК.

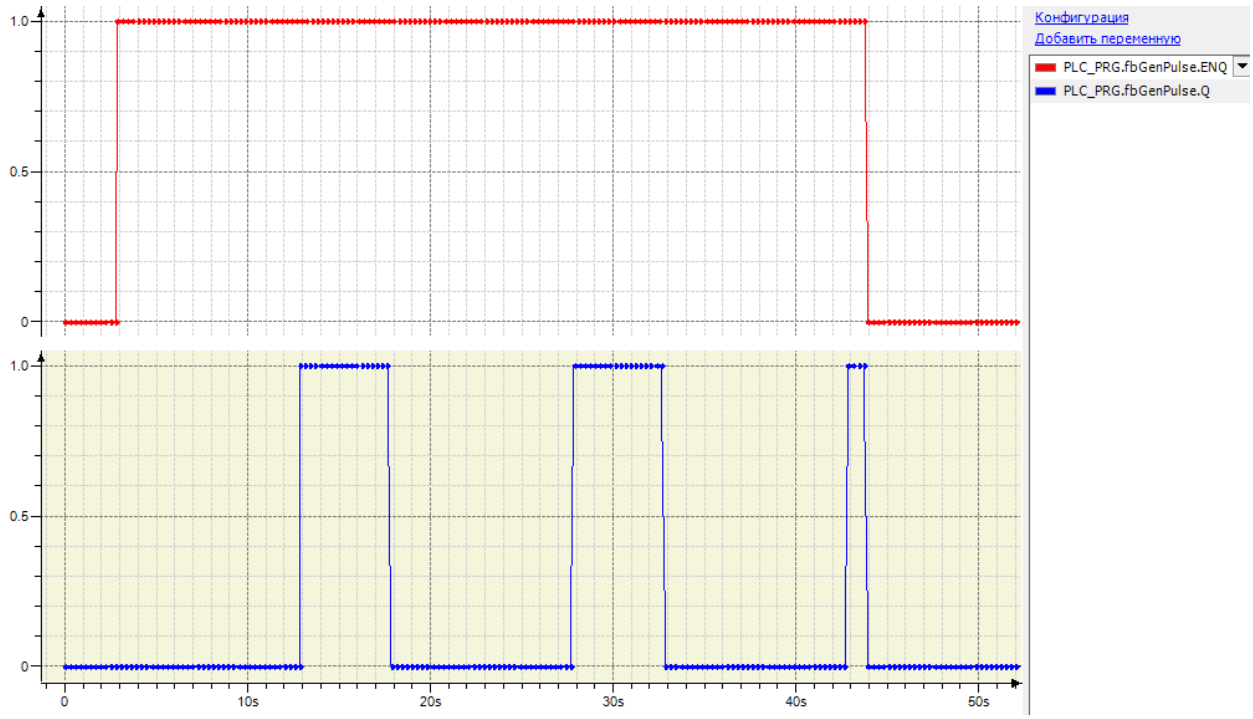


Рис. 18.7. Трассировка работы ФБ **GEN_PULSE** (PTH=T#5s, PTL=T#10s)

18.5. GEN_PW2

Тип модуля: ФБ	Переменная	Тип	Описание
Входы	ENQ	BOOL	Сигнал управления блоком.
	TH1	TIME	Время сигнала верхнего уровня (набор 1).
	TL1	TIME	Время сигнала нижнего уровня (набор 1).
	TH2	TIME	Время сигнала верхнего уровня (набор 2).
	TL2	TIME	Время сигнала нижнего уровня (набор 2).
	TS	BOOL	Сигнал переключения наборов.
Выходы	Q	BOOL	Выход генератора.
	TH	TIME	Текущее время импульса (верхний уровень).
	TL	TIME	Текущее время импульса (нижний уровень).
Используемые модули	T_PLC_MS		

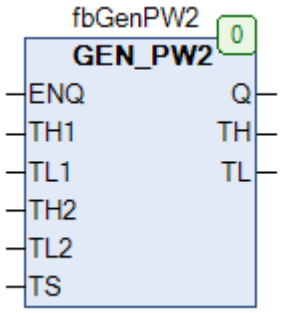


Рис. 18.8. Внешний вид ФБ **GEN_PW2** на языке CFC

Функциональный блок **GEN_PW2** представляет собой генератор прямоугольных импульсов с двумя наборами параметров. Когда вход **EHQ** принимает значение **TRUE**, блок запускается в работу, и на выходе **Q** начинают генерироваться прямоугольные импульсы. Если вход **TS** имеет значение **FALSE**, то длительность генерируемых импульсов определяется значениями входов **TH1** (верхний уровень) и **TL1** (нижний уровень). Если вход **TS** имеет значение **TRUE**, то длительность генерируемых импульсов определяется значениями входов **TH2** (верхний уровень) и **TL2** (нижний уровень). Выходы **TH** и **TL** содержат время, прошедшее с начала генерации текущего импульса (верхнего и нижнего уровня соответственно). Блок начинает работу с импульса нижнего уровня. Если вход **ENQ** принимает значение **FALSE**, то блок прекращает работу; выход **Q** при этом принимает значение **FALSE**. Импульс с длительностью **0** соответствует импульсу длиной в цикл ПЛК.

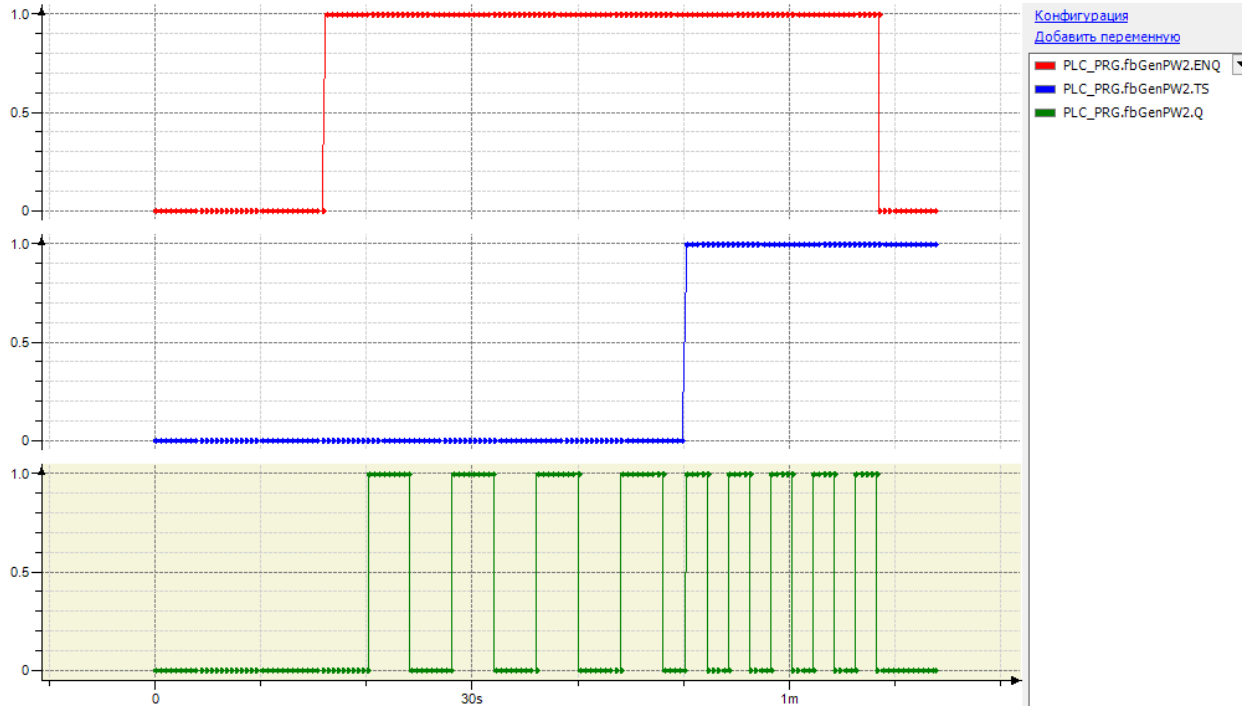


Рис. 18.9. Трассировка работы ФБ **GEN_PW2** (TH1=TL1=T#4s, TH2=TL2=T#2s)

18.6. GEN_RDM

Тип модуля: ФБ	Переменная	Тип	Описание
Входы	PT	BOOL	Период генерации значений.
	AM	REAL	Амплитуда.
	OS	REAL	Смещение.
Выходы	Q	BOOL	Флаг «новое значение».
	Out	REAL	Выход генератора.
Используемые модули	T_PLC_MS, RDM		

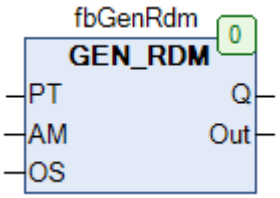


Рис. 18.10. Внешний вид ФБ **GEN_RDM** на языке CFC

Функциональный блок **GEN_RDM** представляет собой генератор псевдослучайных значений. На выходе **Out** с интервалом **PT** генерируется псевдослучайное значение с амплитудой **AM** и смещением **OS**. При появлении нового значения на выходе **Q** генерируется единичный импульс.

Связь значения на выходе **Out** с амплитудой и смещением определяется формулой:

$$\mathrm{Out} = \mathrm{AM} \cdot (\mathrm{RDM} - 0.5) + \mathrm{OS}, \text{где}$$

RDM – псевдослучайное число в диапазоне [0,1].

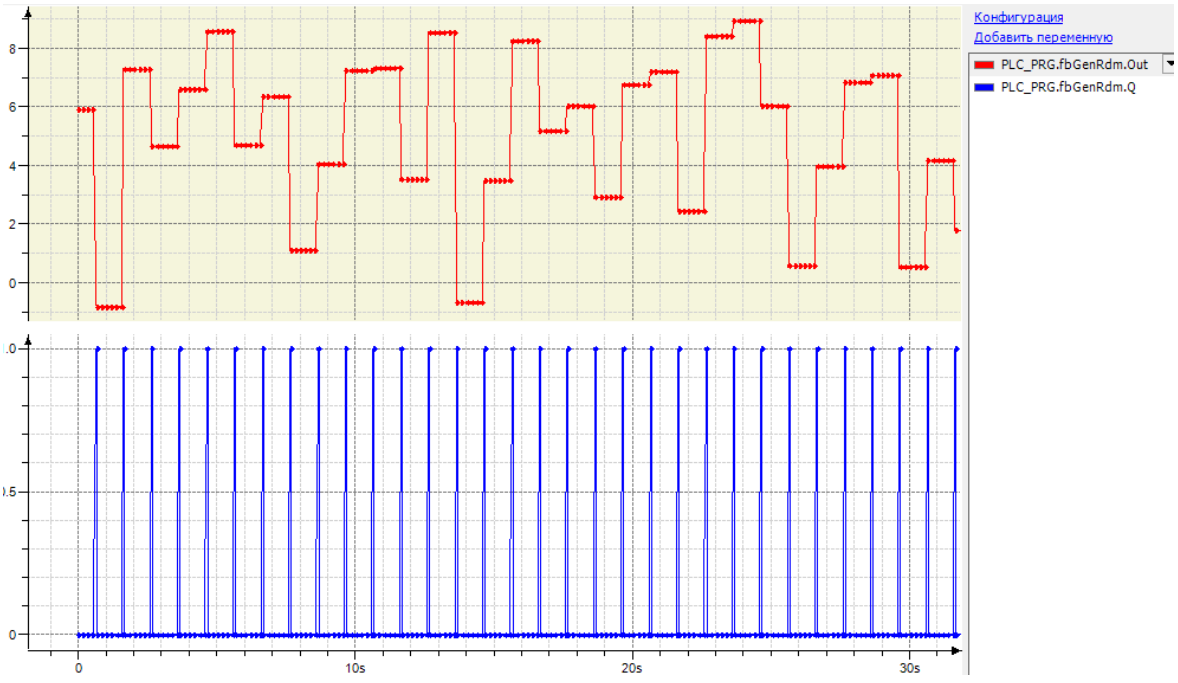


Рис. 18.11. Трассировка работы ФБ **GEN_RDM** (PT=T#1s, AM=10.0, OS=4.0)

18.7. GEN_RDT

Тип модуля: ФБ	Переменная	Тип	Описание
Входы	Enable	BOOL	Сигнал управления блоком.
	Min_Time_ms	TIME	Минимальная пауза между импульсами.
	Max_Time_ms	TIME	Максимальная пауза между импульсами.
	TP_Q	TIME	Длительность импульса.
Выходы	xQ	BOOL	Выход генератора.
Используемые модули	RDM		

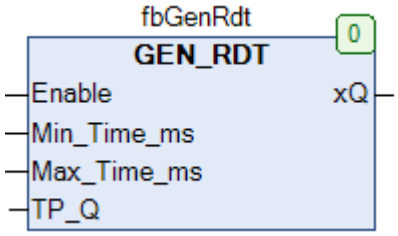


Рис. 18.12. Внешний вид ФБ **GEN_RDT** на языке CFC

Функциональный блок **GEN_RDT** представляет собой генератор импульсов заданной длины с псевдослучайным периодом. Когда вход **Enable** принимает значение **TRUE**, блок запускается в работу, и на выходе **xQ** начинают генерироваться импульсы длительностью **TP_Q**, причем пауза между импульсами является псевдослучайными числом из диапазона **Min_Time_ms…Max_Time_ms**. Импульс с длительностью **0** соответствует импульсу длиной в цикл ПЛК. Если вход **Enable** принимает значение **FALSE**, то блок прекращает работу; выход **xQ** при этом принимает значение **FALSE**.

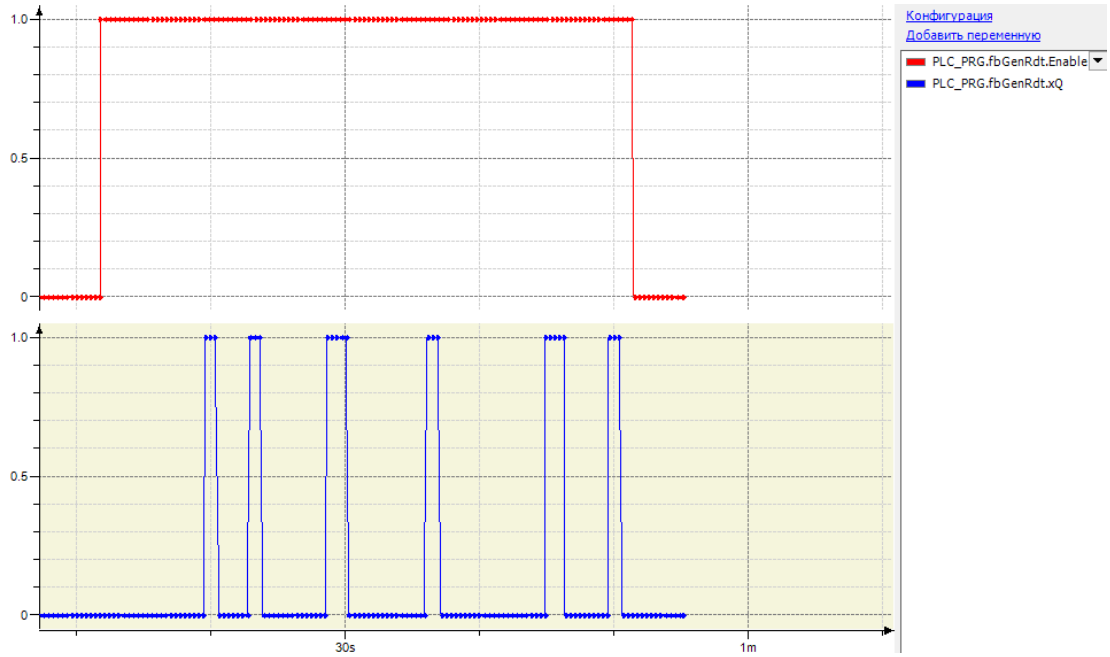


Рис. 18.13. Трассировка работы ФБ **GEN_RDT** (Min_Time_ms=T#5s, Max_Time_Ms=T#10s, TP_Q=T#1s)

18.8. GEN_RMP

Тип модуля: ФБ	Переменная	Тип	Описание
Входы	PT	BOOL	Период генерации значений.
	AM	REAL	Амплитуда.
	OS	REAL	Смещение сигнала относительно нуля.
	DL	REAL	Задержка включения в долях от PT (0…1).
Выходы	Q	BOOL	Флаг «новое значение».
	Out	REAL	Выход генератора.
Используемые модули	T_PLC_MS, MODR, MULTIME		

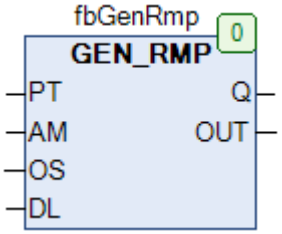


Рис. 18.14. Внешний вид ФБ **GEN_RMP** на языке CFC

Функциональный блок **GEN_RMP** представляет собой генератор пилообразного сигнала. На выходе блока **Out** циклически с периодом **PT** генерируется выходной пилообразный сигнал с амплитудой **AM**, смещением относительно нуля **OS** и задержкой включения **DL**. Задержка используется для генерации нескольких сигналов (с помощью нескольких экземпляров ФБ), смещенных относительно друг друга, и задается в долях от периода **PT**:

- DL=0 – задержки нет;
- DL=0.5 – задержка составляет 50% от времени **PT**;
- DL=1 – задержка равна времени **PT**.

На выходе **Q** генерируется единичный импульс в начале каждого цикла генерации.

На рис. 18.16 приведена трассировка работы двух экземпляров блока для следующих значений входных переменных:

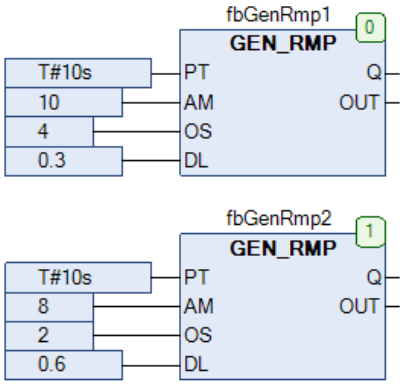


Рис. 18.15. Настройки двух экземпляров ФБ

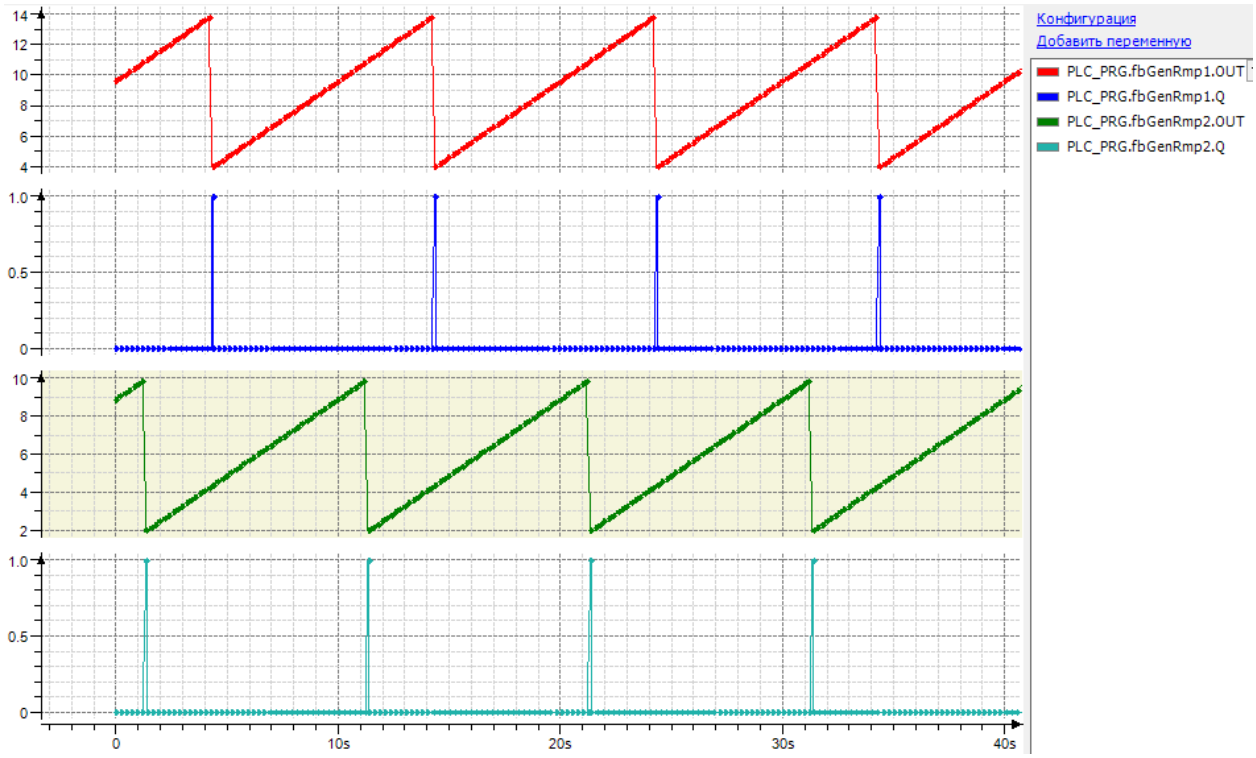


Рис. 18.16. Трассировка работы экземпляров ФБ **GEN_RMP** (см. настройки на рис. 18.15)

18.9. GEN_SIN

Тип модуля: ФБ	Переменная	Тип	Описание
Входы	PT	BOOL	Период колебаний.
	AM	REAL	Амплитуда.
	OS	REAL	Смещение сигнала относительно нуля.
	DL	REAL	Задержка включения в долях от PT (0…1).
Выходы	Q	BOOL	Флаг «знак полупериода».
	Out	REAL	Выход генератора.
Используемые модули	T_PLC_MS, MODR, MULTIME, SIGN_R		

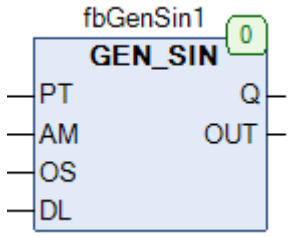


Рис. 18.17. Внешний вид ФБ **GEN_SIN** на языке CFC

Функциональный блок **GEN_SIN** представляет собой генератор синусоидального сигнала. На выходе блока **Out** циклически с периодом **PT** генерируется выходной синусоидальный сигнал с амплитудой **AM**, смещением относительно нуля **OS** и задержкой включения **DL**. Задержка используется для генерации нескольких сигналов (с помощью нескольких экземпляров ФБ), смещенных относительно друг друга, и задается в долях от периода **PT**:

- DL=0 – задержки нет;
- DL=0.5 – задержка составляет 50% от времени **PT**;
- DL=1 – задержка равна времени **PT**.

Выход **Q** имеет значение **TRUE** во время положительного полупериода синусоиды и **FALSE** – во время отрицательного.

На рис. 18.19 приведена трассировка работы двух экземпляров блока для следующих значений входных переменных:

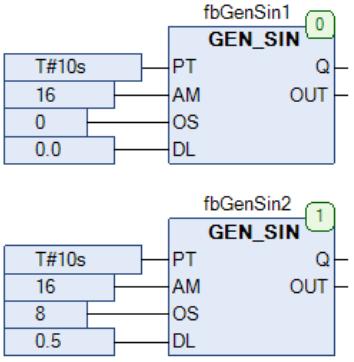


Рис. 18.18. Настройки двух экземпляров ФБ

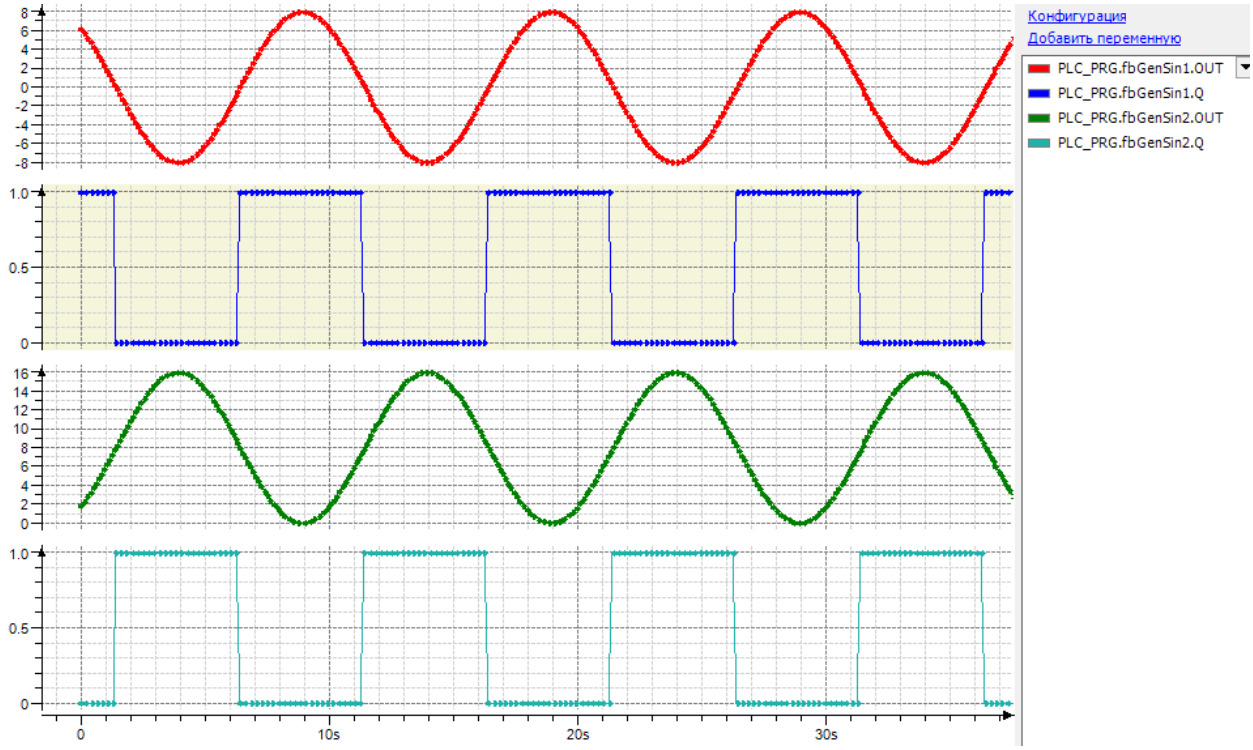


Рис. 18.19. Трассировка работы экземпляров ФБ **GEN_SIN** (см. настройки на рис. 18.18)

18.10. GEN_SQR

Тип модуля: ФБ	Переменная	Тип	Описание
Входы	PT	BOOL	Период.
	AM	REAL	Амплитуда.
	OS	REAL	Смещение сигнала относительно нуля.
	DC	REAL	Время сигнала верхнего уровня в долях от PT (0…1).
	DL	REAL	Задержка включения в долях от PT (0…1).
Выходы	Q	BOOL	Флаг «знак полупериода».
	Out	REAL	Выход генератора.
Используемые модули	T_PLC_MS, MODR, MULTIME		

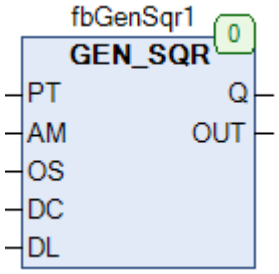


Рис. 18.20. Внешний вид ФБ **GEN_SQR** на языке CFC

Функциональный блок **GEN_SQR** представляет собой генератор прямоугольного сигнала. На выходе блока **Out** циклически с периодом **PT** генерируется выходной прямоугольный сигнал с амплитудой **AM**, смещением относительно нуля **OS** и задержкой включения **DL**. Задержка используется для генерации нескольких сигналов (с помощью нескольких экземпляров ФБ), смещенных относительно друг друга, и задается в долях от периода **PT**:

- DL=0 – задержки нет;
- DL=0.5 – задержка составляет 50% от времени **PT**;
- DL=1 – задержка равна времени **PT**.

Вход **DC** определяет время сигнала верхнего уровня в долях от периода **PT**. Например, если **PT=T#1s** и **DC=0.5**, то на выходе блока в течение 500 мс будет сигнал верхнего уровня, затем в течение 500 мс – нижнего, после чего цикл повторится.

Выход **Q** имеет значение **TRUE** во время сигнала верхнего уровня на выходе генератора и **FALSE** – во время сигнала нижнего уровня.

На рис. 18.21 приведена трассировка работы двух экземпляров блока для следующих значений входных переменных:

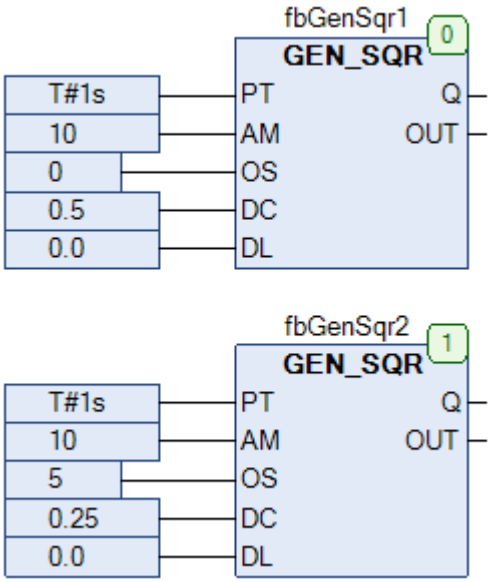


Рис. 18.20. Настройки двух экземпляров ФБ

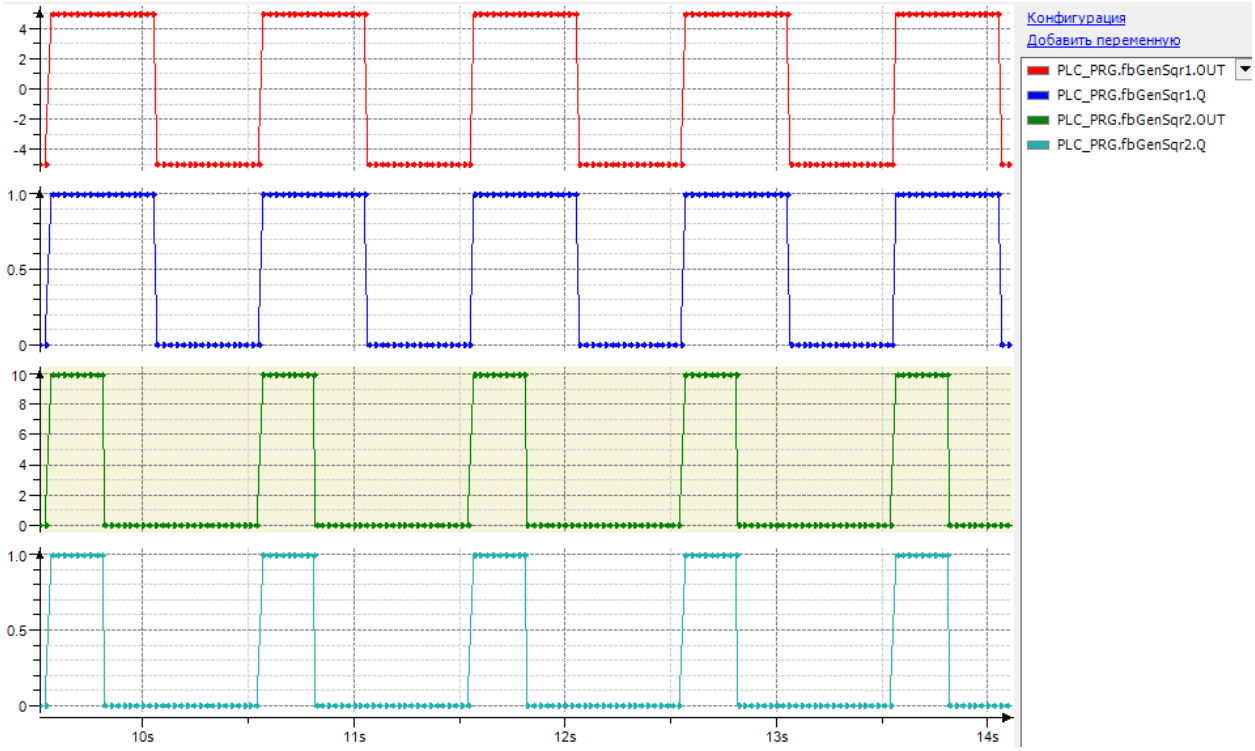


Рис. 18.21. Трассировка работы экземпляров ФБ **GEN_SQR** (см. настройки на рис. 18.20)

18.11. PWM_DC

Тип модуля: ФБ	**Переменная**	**Тип**	**Описание**
Входы	F	TIME	Частота генератора (в Гц).
	DC	REAL	Время сигнала верхнего уровня в долях от 1/F (0…1).
Выходы	Q	BOOL	Выход генератора.
Используемые модули	CLK_PRG , TP_X		

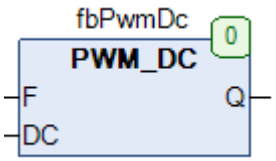


Рис. 18.22. Внешний вид ФБ **PWM_DC** на языке CFC

Функциональный блок **PWM_DC** представляет собой генератор импульсов с задаваемой частотой **F**. Вход **DC** определяет время сигнала верхнего уровня в долях от периода **PT**. Например, если **F=20 Гц** и **DC=0.1**, то период генератора = 1/F = 50 мс, причем время сигнала верхнего уровня = 0.1·50 = 5 мс, а время сигнала нижнего уровня – 45 мс.

Сгенерированный сигнал подается на выход **Q**.

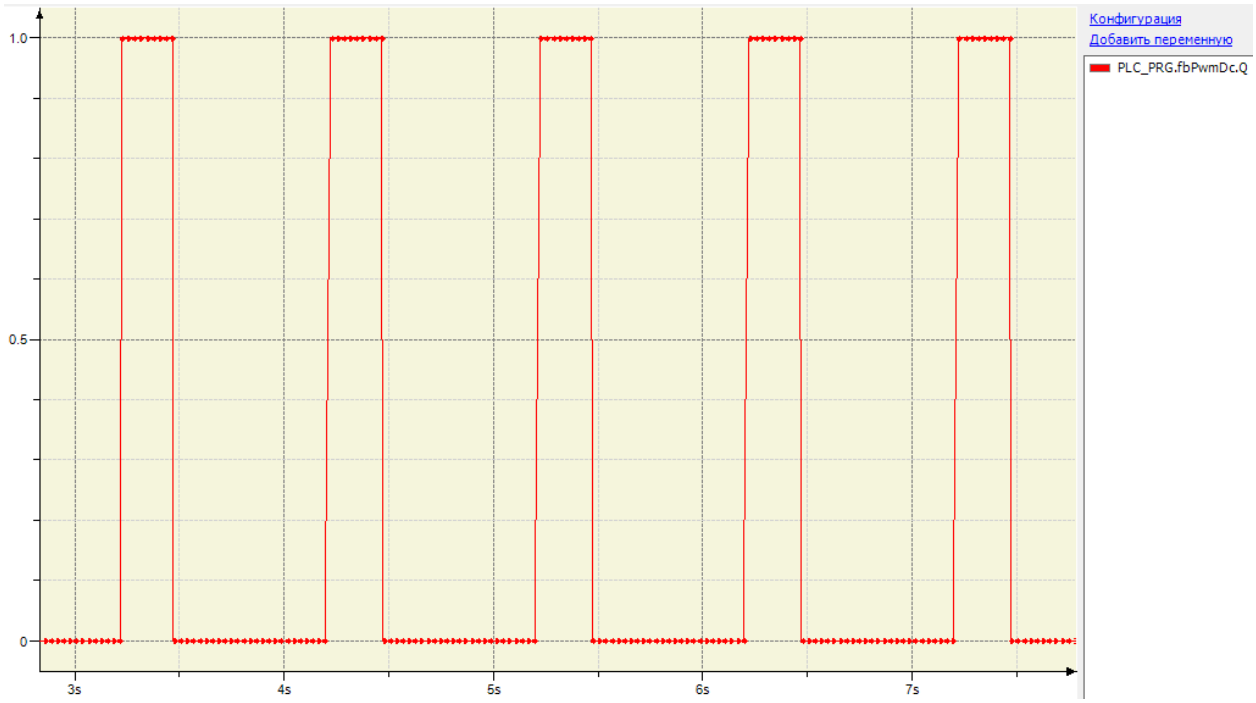


Рис. 18.23. Трассировка работы ФБ **PWM_DC** (F=1, DC=0.25)

18.12. PWM_PW

Тип модуля: ФБ	Переменная	Тип	Описание
Входы	F	TIME	Частота генератора (в Гц).
	PW	REAL	Время сигнала верхнего уровня.
Выходы	Q	BOOL	Выход генератора.
Используемые модули	CLK_PRG , TP_X		

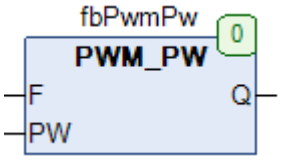


Рис. 18.24. Внешний вид ФБ PWM_PW на языке CFC

Функциональный блок **PWM_DC** представляет собой генератор импульсов с задаваемой частотой **F** и временем сигнала верхнего уровня **PW**. Сгенерированный сигнал подается на выход **Q**.

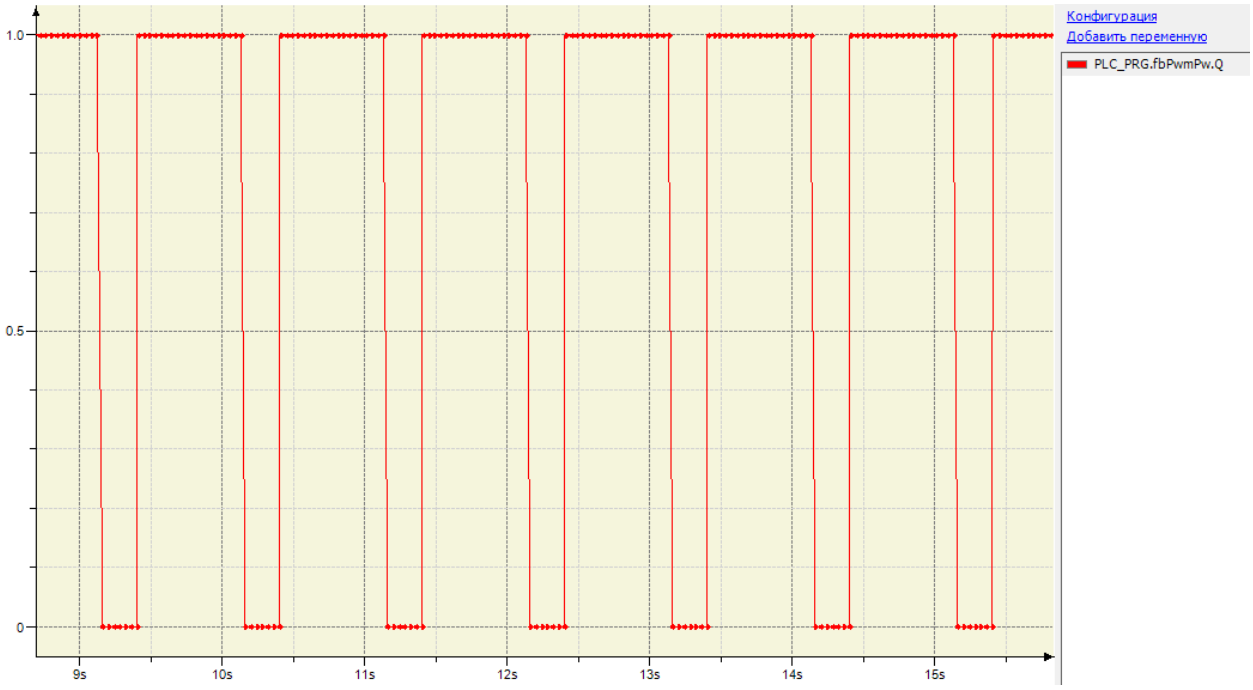


Рис. 18.25. Трассировка работы ФБ **PWM_PW** (F=1, PW=T#750ms)

18.13. RMP_B

Тип модуля: ФБ	Переменная	Тип	Описание
Входы	SET	BOOL	Сигнал установки на выход макс. значения.
	PT	TIME	Время генерации.
	E	BOOL	Сигнал управления блоком.
	UP	BOOL	Направление изменения выхода.
	RST	BOOL	Сигнал установки на выход мин. значения.
Входы-выходы	OUT	BYTE	Выход блока.
	BUSY	BOOL	Флаг «генератор в работе».
	HIGH	BOOL	Флаг «достигнуто макс. значение».
	LOW	BOOL	Флаг «достигнуто мин. значение».
Используемые модули	_RMP_B		

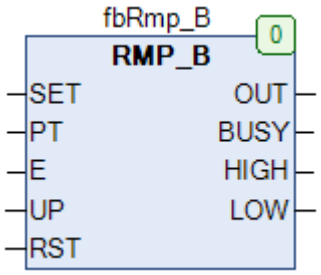


Рис. 18.26. Внешний вид ФБ **RMP_B** на языке CFC

Функциональный блок **RMP_B** представляет собой генератор линейной функции. Пока вход **E** имеет значение **TRUE** блок находится в работе и значение выхода **OUT** линейно изменяется от текущего значения до максимального (при **UP=TRUE**) или минимального (при **UP=FALSE**). Минимальное и максимальное значения определяются диапазоном типа **BYTE** и соответственно составляют **0** и **255**. Скорость изменения выходного сигнала определяется отношением (**255/PT**). Сигнал по переднему фронту на входе **SET** устанавливает на выход максимальное значение (**255**), сигнал по переднему фронту на входе **RST** – минимальное (**0**). Выход **BUSY** имеет значение **TRUE**, пока выход меняет свое значение. Выходы **HIGH** и **LOW** принимают значение **TRUE** при достижении выходом **OUT** максимального и минимального значения соответственно.

Если вход **E** принимает значение **FALSE**, то блок прекращает работу, при этом выход **OUT** сохраняет свое последнее значение.

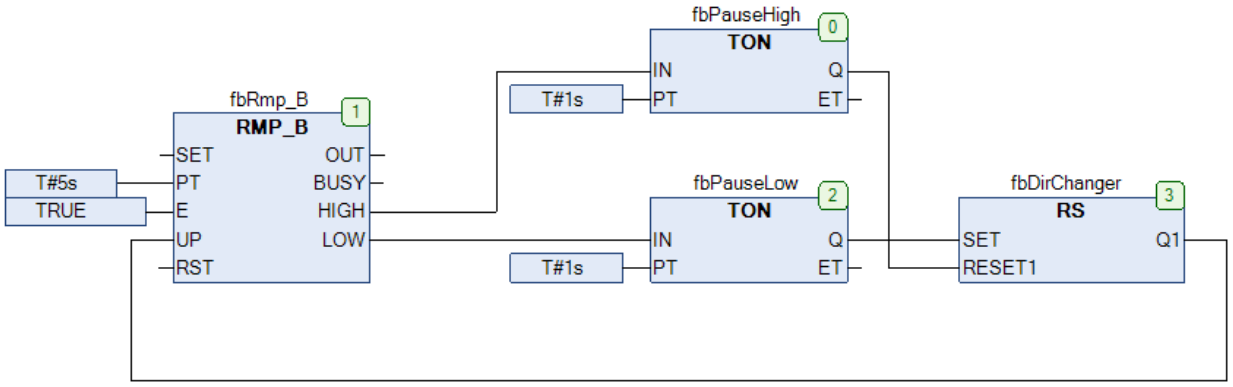
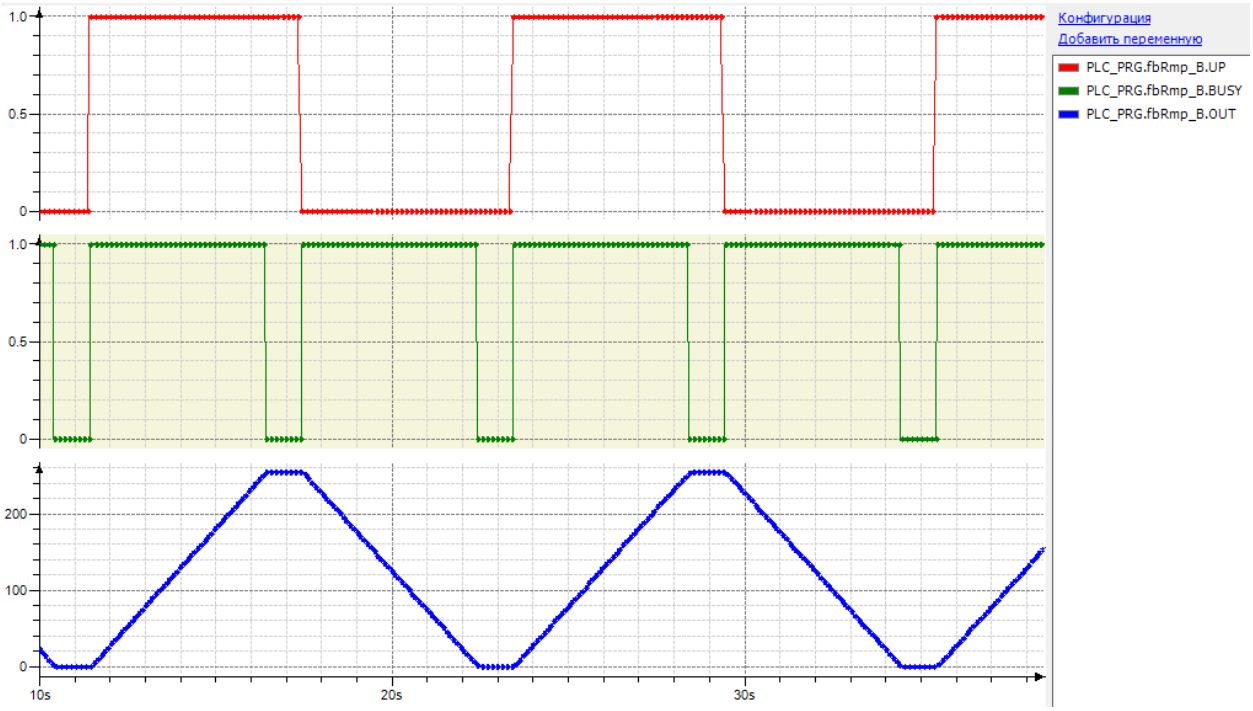


Рис. 18.27. Пример работы с ФБ **RMP_B** на языке CFC



18.28. Трассировка работы ФБ **RMP_B** (см. настройки на рис. 18.27)

18.14. RMP_SOFT

Тип модуля: ФБ	**Переменная**	**Тип**	**Описание**
Входы	IN	BOOL	Сигнал управления блоком.
	VAL	BYTE	Максимальное значение выхода.
Входы-выходы	OUT	BYTE	Выход блока.
Параметры	PT_OFF	TIME	Время спада сигнала.
	PT_ON	TIME	Время нарастания сигнала.
Используемые модули	_RMP_B		

Рис. 18.29. Внешний вид ФБ **RMP_SOFT** на языке CFC

Функциональный блок **RMP_SOFT** представляет собой генератор линейной функции с ограничением выходного сигнала. Пока вход **IN** имеет значение **TRUE**, значение выхода **OUT** линейно изменяется от нулевого значения до значения **VAL** за время **PT_ON**. Пока вход **IN** имеет значение **FALSE**, значение выхода **OUT** линейно изменяется от текущего значения до нулевого за время **PT_OFF**.

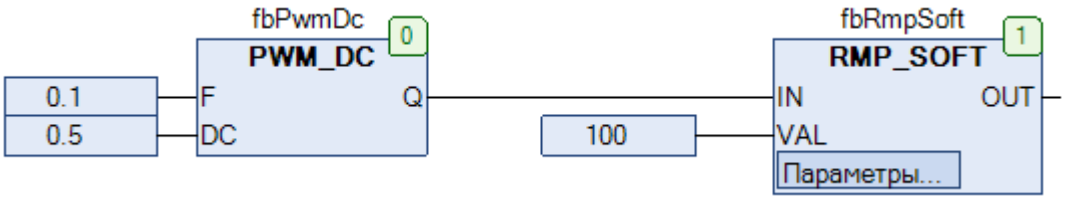


Рис. 18.30. Пример работы с ФБ **RMP_SOFT** на языке CFC

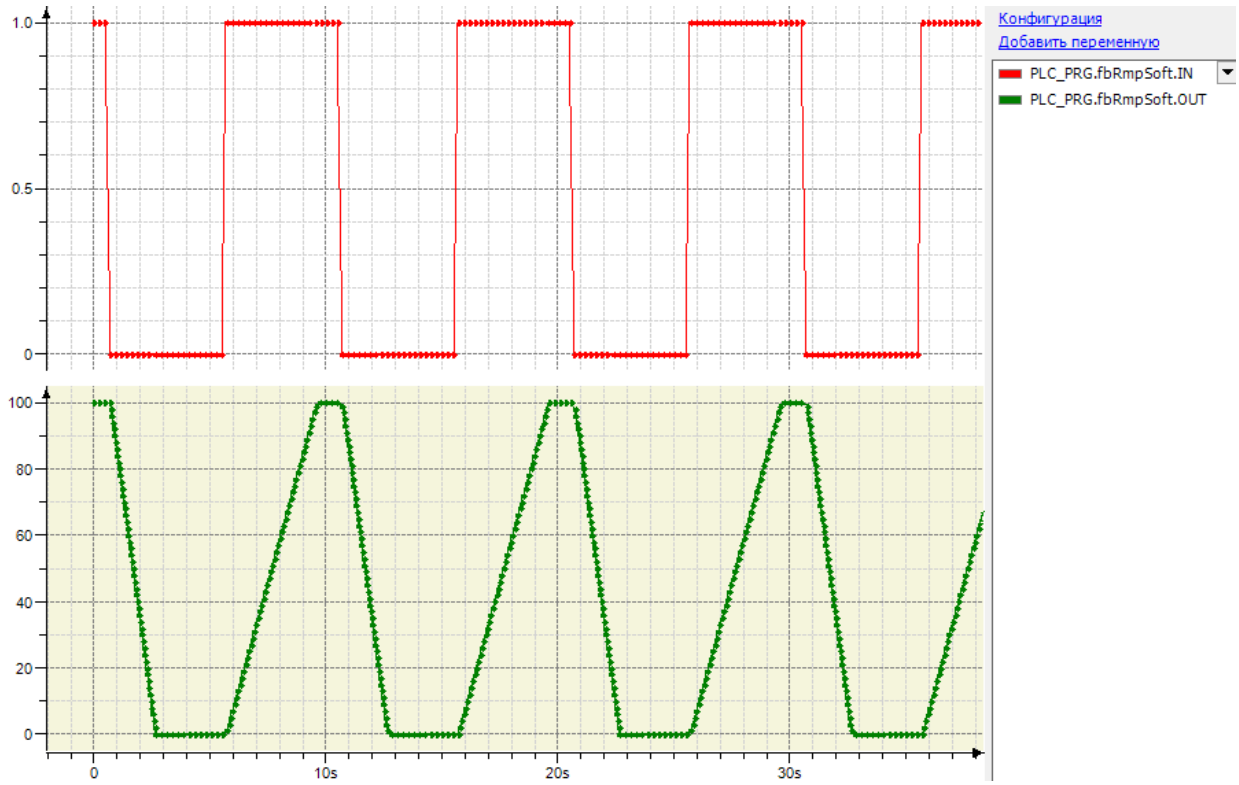


Рис. 18.31. Трассировка работы ФБ **RMP_SOFT** (см. настройки на рис. 18.30, PT_ON=T#10s, PT_OFF=T#5s)

18.15. RMP_W

Тип модуля: ФБ	**Переменная**	**Тип**	**Описание**
Входы	SET	BOOL	Сигнал установки на выход макс. значения.
	PT	TIME	Время генерации.
	E	BOOL	Сигнал управления блоком.
	UP	BOOL	Направление изменения выхода.
	RST	BOOL	Сигнал установки на выход мин. значения.
Входы-выходы	OUT	WORD	Выход блока.
	BUSY	BOOL	Флаг «генератор в работе».
	HIGH	BOOL	Флаг «достигнуто макс. значение».
	LOW	BOOL	Флаг «достигнуто мин. значение».
Используемые модули	_RMP_W		

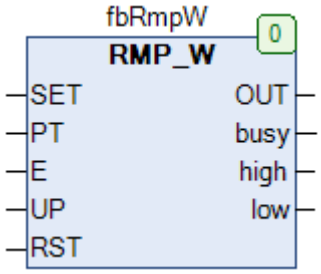


Рис. 18.32. Внешний вид ФБ **RMP_W** на языке CFC

Функциональный блок **RMP_W** представляет собой генератор линейной функции. Принцип работы блока полностью соответствует блоку RMP_B, единственным отличие является тип выхода генератора – у данного блока он имеет тип **WORD**.

19. Обработка сигналов

19.1. AIN

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	in	DWORD	Цифровой сигнал.
	Bits	BYTE	Число значащих бит.
	sign	BYTE	Номер бита, определяющего знак.
	low	REAL	Нижний предел аналогового сигнала.
	high	REAL	Верхний предел аналогового сигнала.
Выходы	AIN	REAL	Аналоговый сигнал.

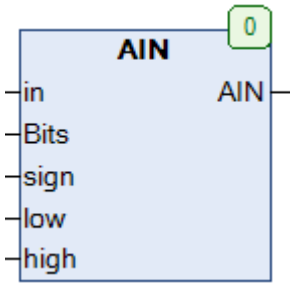


Рис. 19.1. Внешний вид функции **AIN** на языке CFC

Функция **AIN** используется для преобразования цифрового сигнала **in** в аналоговый **AIN**. Вход **Bits** определяет число значащих бит цифрового сигнала (максимальное значение – **32**). Вход **sign** определяет номер бита, содержащего знак (его значение **FALSE** соответствует знаку «+»); если такого бита нет, то на вход **sign** подается значение **16#FF**. Входы **low** и **high** определяют минимальное и максимальное значение аналогового сигнала. Нумерация бит ведется с нуля.

Ниже приведено несколько примеров настройки входов для разных сигналов.

- Аналоговый сигнал в диапазоне 0–10 представлен цифровым 12-битным сигналом: Bits=12, sign=16#255, low=0.0, high=10.0;
- Аналоговый сигнал в диапазоне -10–10 представлен цифровым 14-битным сигналом (биты 0-13) с битом знака (бит 14): Bits=14, sign=14, low=-10.0, high=10.0;
- Аналоговый сигнал в диапазоне -10–10 представлен цифровым 24-битным сигналом без бита знака: Bits=24, sign=16#255, low=-10.0, high=10.0.

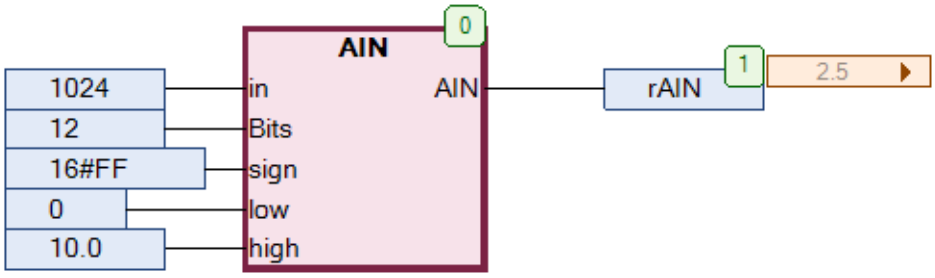


Рис. 19.2. Пример работы с функцией **AIN** на языке CFC

19.2. AIN1

Тип модуля: ФБ	Переменная	Тип	Описание
Входы	in	DWORD	Цифровой сигнал.
Выходы	out	REAL	Аналоговый сигнал.
	sign	BOOL	Знак сигнала.
	error	BOOL	Флаг «ошибка».
	overflow	BOOL	Флаг «переполнение».
Параметры	sign_bit	INT	Номер бита, определяющего знак.
	error_bit	INT	Номер бита ошибка.
	error_code_en	BOOL	Бит обработки кода ошибки.
	error_code	DWORD	Код ошибки.
	overflow_bit	INT	Номер бита переполнения.
	overflow _code_en	BOOL	Бит обработки кода переполнения.
	overflow _code	DWORD	Код переполнения.
	Bit_0	INT	Номер первого значащего бита.
	Bit_N	INT	Номер последнего значащего бита.
	out_min	REAL	Нижний предел аналогового сигнала.
	out_max	REAL	Верхний предел аналогового сигнала.
	code_min	DWORD	Нижний предел цифрового сигнала.
	code_max	DWORD	Верхний предел цифрового сигнала.
	error_output	REAL	Значение выхода в случае ошибки.
	overflow_output	REAL	Значение выхода в случае переполнения.

Рис. 19.3. Внешний вид ФБ **AIN1** на языке CFC

Функциональный блок **AIN** используется для преобразования цифрового сигнала **in** в аналоговый **out**. Параметр **sign_bit** определяет номер бита, содержащего знак (его значение **FALSE** соответствует знаку «+»); если такого бита нет, то вход можно оставить пустым. Нумерация бит ведется с нуля.

Параметр **error_bit** определяет номера бита ошибки; если такого бита нет, то вход можно оставить пустым. Некоторые АЦП в случае ошибки транслируют на выход ее код. Этот код может быть задан в параметре **error_code**; параметр **error_code_en** в этом случае должен иметь значение **TRUE** для разрешения обработки кода. Параметр **error_output** определяет значение, которое будет подано на выход **out** в том случае, если диагностирована ошибка.

Параметр **overflow_bit** определяет номера бита переполнения (выхода за диапазон допустимых значений) если такого бита нет, то вход можно оставить пустым. Некоторые АЦП в случае ошибки переполнения транслируют на выход ее код. Этот код может быть задан в параметре **overflow_code**; параметр **overflow_code_en** в этом случае должен иметь значение **TRUE** для разрешения обработки кода. Параметры **code_min** и **code_max** определяют минимальное и максимальное значение цифрового сигнала. Параметр **overflow_output** определяет значение, которое будет подано на выход **out** в том случае, если выходной сигнал выходит за допустимый диапазон.

Параметры **out_min** и **out_max** определяют минимальное и максимальное значение аналогового сигнала.

Параметры **Bit_0** и **Bit_N** определяют номера первого и последнего значащих бит.

Выход **out** содержит полученное аналоговое значение. Выход **sign** определяет значение бита знака. Выходы **error** и **overflow** принимают значение **TRUE** при возникновении ошибки и переполнения соответственно.

Ниже приведен пример работы блока со следующими значениями параметров: code_min=0, code_max=4095, out_min=0.0, out_max=10.0, Bit_0=0, Bit_N=16, overflow_output=9999.

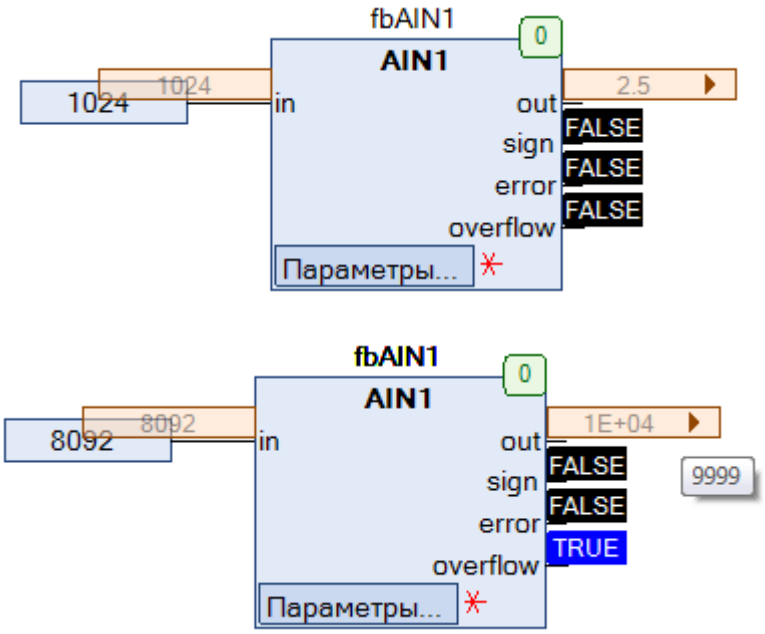


Рис. 19.4. Пример работы с ФБ **AIN1** на языке CFC

19.3. AOUT

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	in	REAL	Аналоговый сигнал.
Выходы	AOUT	DWORD	Цифровой сигнал.
Параметры	Bits	BYTE	Число значащих бит.
	sign	BYTE	Номер бита, определяющего знак.
	low	REAL	Нижний предел аналогового сигнала.
	high	REAL	Верхний предел аналогового сигнала.

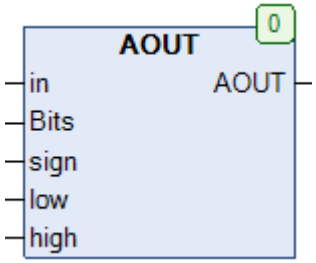


Рис. 19.5. Внешний вид функции **AOUT** на языке CFC

Функция **AOUT** используется для преобразования аналогового сигнала **in** в цифровой **AOUT**. Вход **Bits** определяет число значащих бит цифрового сигнала (максимальное значение – 32). Вход **sign** определяет номер бита, содержащего знак (его значение **FALSE** соответствует знаку «+»); если такой бит не требуется, то на вход подается значение **16#FF**. Входы **low** и **high** определяют минимальное и максимальное значение аналогового сигнала. Нумерация бит ведется с нуля.

Ниже приведено несколько примеров настройки входов для разных сигналов.

- Аналоговый сигнал в диапазоне 0–10 преобразуется в цифровой 12-битный сигнал: Bits=12, sign=16#255, low=0.0, high=10.0;
- Аналоговый сигнал в диапазоне -10–10 преобразуется в цифровой 14-битный сигнал (биты 0-13) с битом знака (бит 14): Bits=14, sign=14, low=-10.0, high=10.0;
- Аналоговый сигнал в диапазоне -10–10 преобразуется в цифровой 24-битный сигнал без бита знака: Bits=24, sign=16#255, low=-10.0, high=10.0.

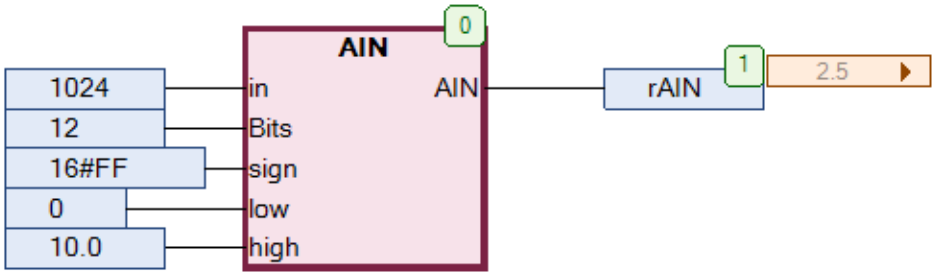


Рис. 19.6. Пример работы с функцией **AOUT** на языке CFC

19.4. AOUT1

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	in	REAL	Аналоговый сигнал.
Выходы	AOUT1	DWORD	Цифровой сигнал.
Параметры	Bit_0	INT	Номер первого значащего бита.
	Bit_N	INT	Номер последнего значащего бита.
	sign	INT	Номер бита, определяющего знак.
	low	REAL	Нижний предел аналогового сигнала.
	high	REAL	Верхний предел аналогового сигнала.
Используемые модули	SIGN_R		

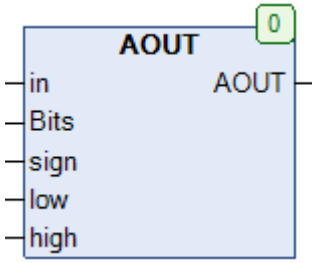


Рис. 19.7. Внешний вид функции **AOU1T** на языке CFC

Функция **AOUT1** используется для преобразования аналогового сигнала **in** в цифровой **AOUT1**. Параметр **sign_bit** определяет номер бита, содержащего знак (его значение **FALSE** соответствует знаку «+»); если такой бит не требуется, то вход можно оставить пустым. Параметры **Bit_0** и **Bit_N** определяют номера первого и последнего значащих бит. Входы **low** и **high** определяют минимальное и максимальное значение аналогового сигнала. Нумерация бит ведется с нуля.

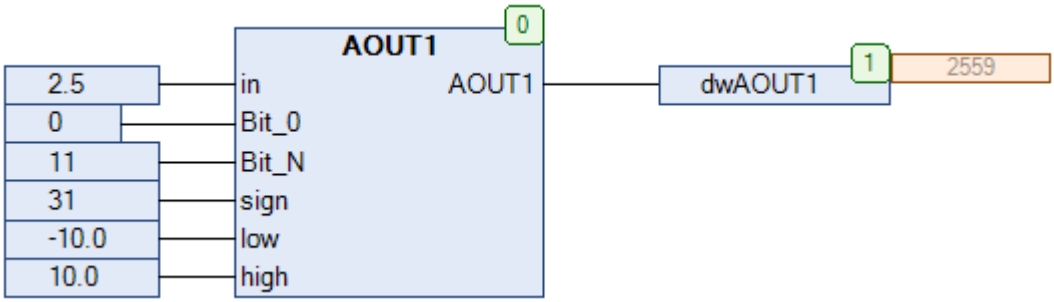


Рис. 19.8. Пример работы с функцией **AOUT1** на языке CFC

19.5. BYTE_TO_RANGE

Тип модуля: функция	Переменная	Тип	Описание
Входы	x	BYTE	Исходное значение.
	low	REAL	Нижний предел отмасштабированного значения.
	high	REAL	Верхний предел отмасштабированного значения.
Выходы	BYTE_TO_RANGE	REAL	Отмасштабированное значение.

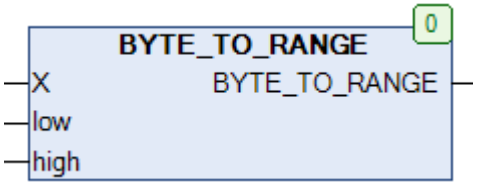


Рис. 19.9. Внешний вид функции **BYTE_TO_RANGE** на языке CFC

Функция **BYTE_TO_RANGE** линейно масштабирует входное целочисленное значение **x** из диапазона (**0…255**) в значение с плавающей точкой из диапазона (**low…high**). См. также обратную функцию RANGE_TO_BYTE.

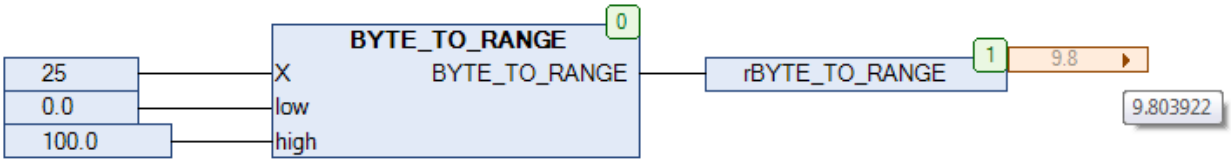


Рис. 19.10. Пример работы с функцией **BYTE_TO_RANGE** на языке CFC

19.6. DELAY

Тип модуля: функция	Переменная	Тип	Описание
Входы	IN	REAL	Исходное значение.
	N	INT	Кол-во циклов задержки.
	RST	BOOL	Сигнал сброса блока.
Выходы	OUT	REAL	Значение с задержкой.
Используемые модули	INC1		

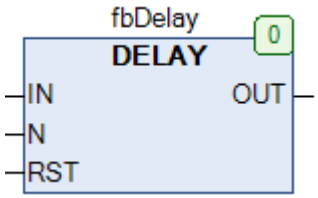


Рис. 19.11. Внешний вид ФБ **DELAY** на языке CFC

Функциональный блок **DELAY** транслирует на выход **OUT** значение входа **IN** спустя время задержки, равное **N** циклов ПЛК. По переднему фронту входа **RST** память блока очищается, и задержка начинает отсчитываться заново.

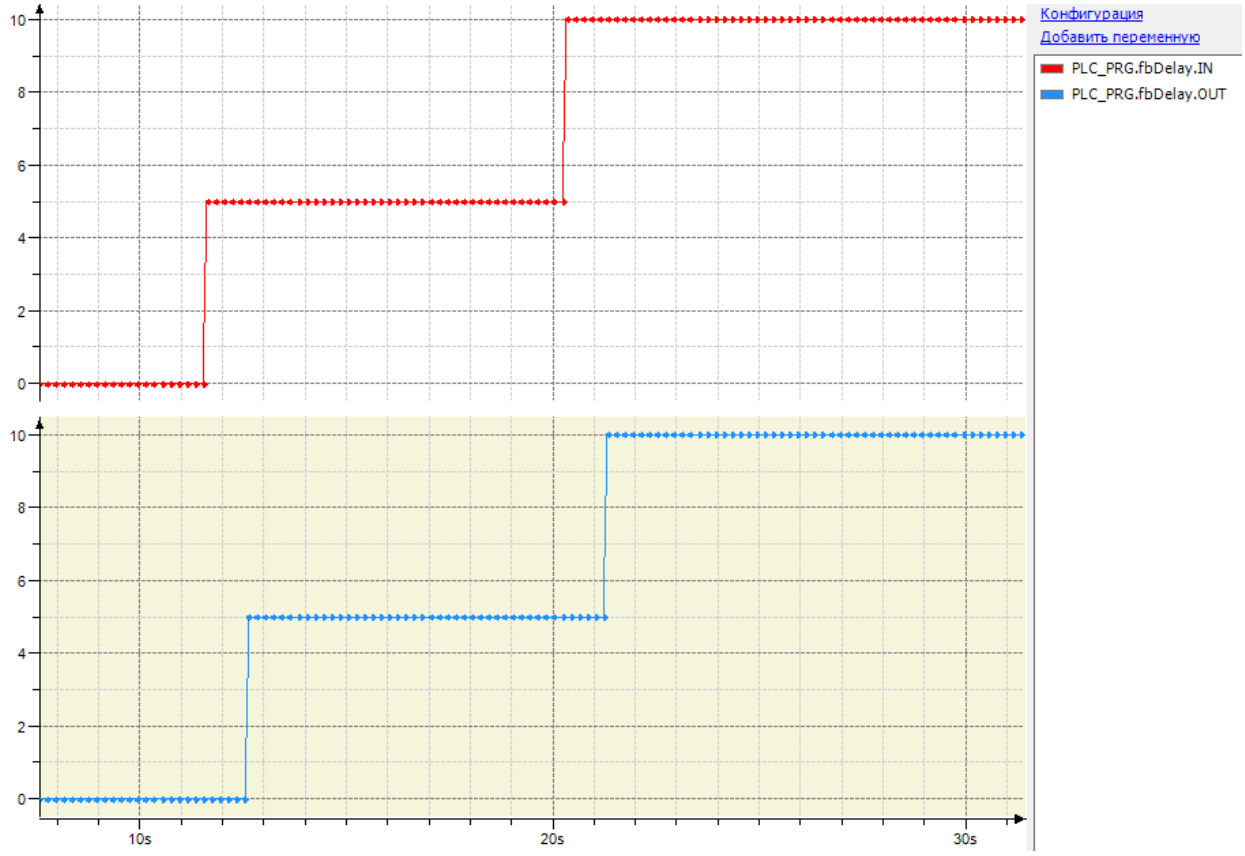


Рис. 19.12. Трассировка работы ФБ **DELAY** для ПЛК с временем цикла=100 мс, N=10

19.7. DELAY_4

Тип модуля: функция	Переменная	Тип	Описание
Входы	IN	REAL	Исходное значение.
Выходы	OUT1	REAL	Значение с задержкой (1 цикл).
	OUT2	REAL	Значение с задержкой (2 цикла).
	OUT3	REAL	Значение с задержкой (3 цикла).
	OUT4	REAL	Значение с задержкой (4 цикла).

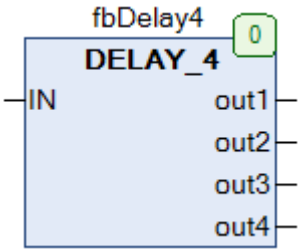


Рис. 19.13. Внешний вид ФБ **DELAY_4** на языке CFC

Функциональный блок **DELAY_4** транслирует на выходы **OUT1…OUT4** значение входа **IN** спустя время задержки, равное 1…4 цикла ПЛК.

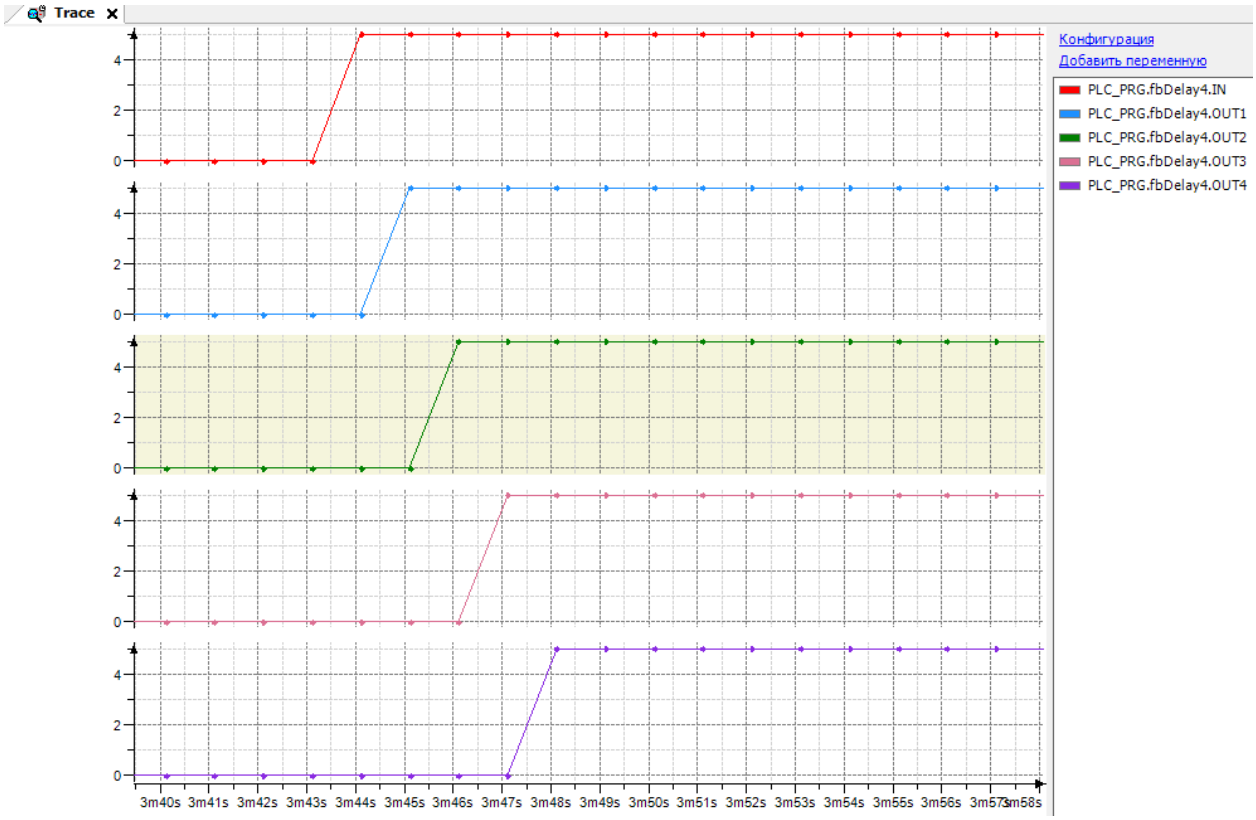


Рис. 19.14. Трассировка работы ФБ **DELAY_4** для ПЛК с временем цикла=1 с

19.8. FADE

Тип модуля: ФБ	Переменная	Тип	Описание
Входы	IN1	REAL	Значение 1.
	IN2	REAL	Значение 2.
	F	BOOL	Направление изменения.
	TF	TIME	Время переключения значений.
	rst	BOOL	Сигнал сброса блока.
Выходы	Y	REAL	Выход блока.
Используемые модули	RMP_W		

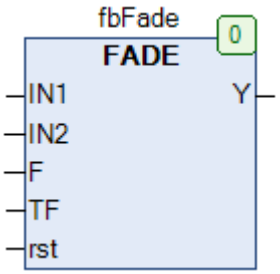


Рис. 19.15. Внешний вид ФБ **FADE** на языке CFC

Функциональный блок **FADE** реализует плавное переключение входных значений **IN1** и **IN2** на выходе **Y** – от меньшего к большему, если вход **F** имеет значение **TRUE**, или от большего к меньшему, если вход **F** имеет значение **FALSE**.

Вход **TF** определяет время переключения (**T** – время с начала работы блока):

- $\mathrm{Y}(\mathrm{T}=0)=\mathrm{IN1}$
- $\mathrm{Y}(\mathrm{T}=\mathrm{TF})=\mathrm{IN2}$
- $\mathrm{Y}(0<\mathrm{T}<\mathrm{TF})=\frac{\mathrm{T}}{\mathrm{TF}}\cdot\mathrm{IN1}+(1-\frac{\mathrm{T}}{\mathrm{TF}})\cdot\mathrm{IN2}$

По переднему фронту входа **RST** выход **Y** принимает значение I**N1** (если **F=FALSE**) или **IN2** (если **F=TRUE**).

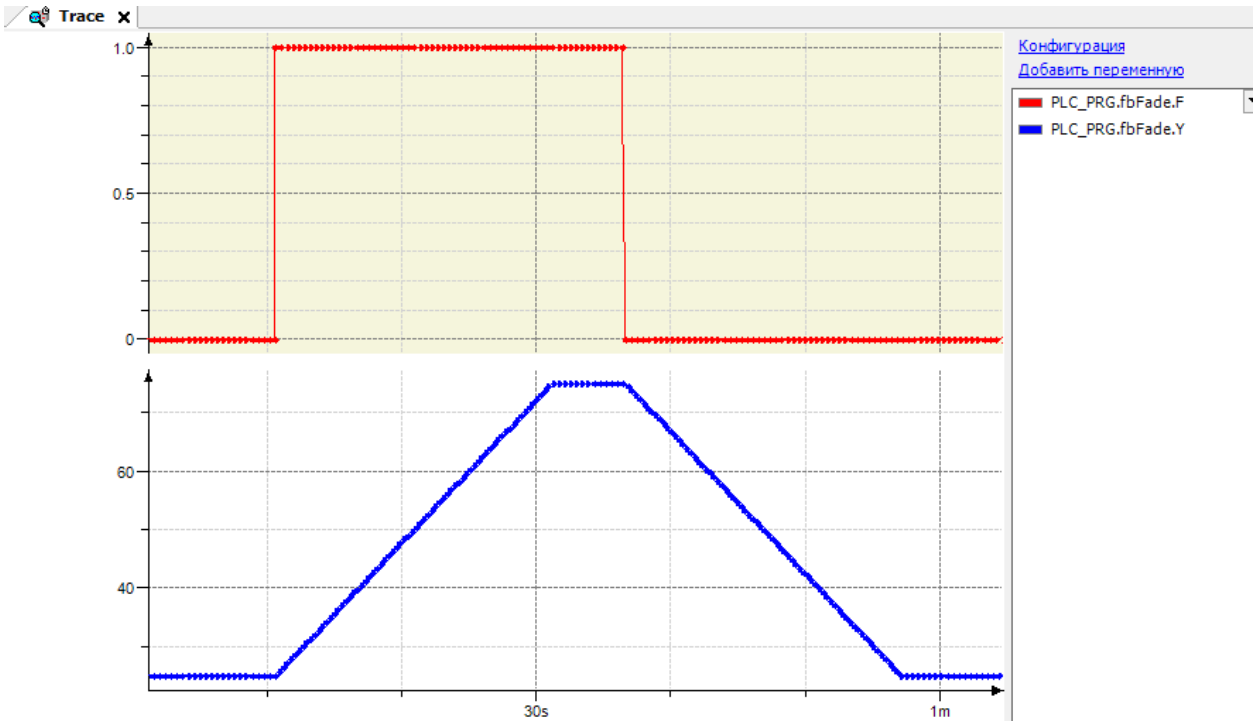


Рис. 19.16. Трассировка работы ФБ **FADE** (IN1=25.0, IN=75.0, TF=T#20s)

19.9. FILTER_DW

Тип модуля: ФБ	Переменная	Тип	Описание
Входы	X	DWORD	Фильтруемое значение
	T	TIME	Постоянная времени фильтра.
Выходы	Y	DWORD	Отфильтрованное значение.
Используемые модули	T_PLC_MS		

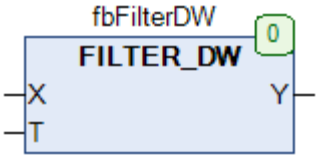


Рис. 19.17. Внешний вид ФБ **FILTER_DW** на языке CFC

Функциональный блок **FILTER_DW** представляет собой фильтр нижних частот для входного значения **X** типа **DWORD**. Это может быть полезным при фильтрации шума в сигналах от датчиков. Вход **T** определяет постоянную времени фильтра, которая связана с частотой пропускания следующей формулой: $f_{\text{гр}} = \frac{1}{2\pi T}$

На выход **Y** подается отфильтрованное значение.

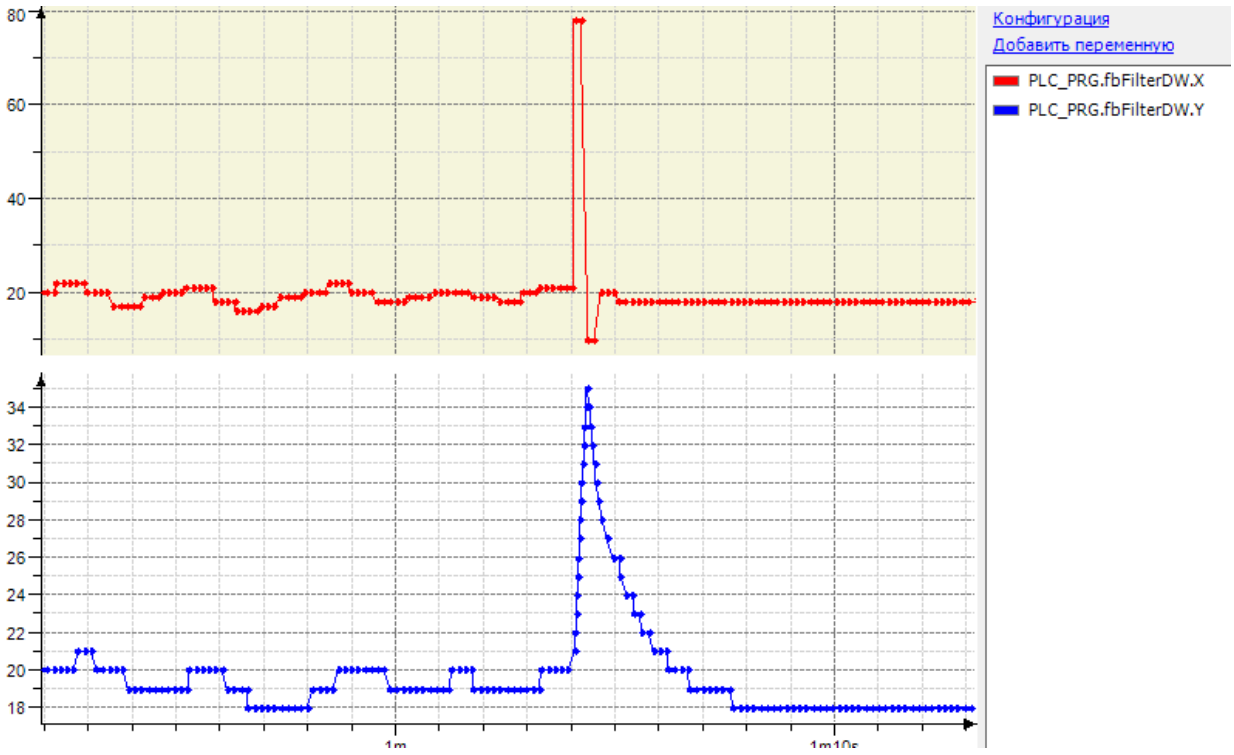


Рис. 19.18. Трассировка работы ФБ **FILTER_DW** (T=T#1s)

19.10. FILTER_I

Тип модуля: ФБ	Переменная	Тип	Описание
Входы	X	INT	Фильтруемое значение
	T	TIME	Постоянная времени фильтра.
Выходы	Y	INT	Отфильтрованное значение.
Используемые модули	T_PLC_MS		

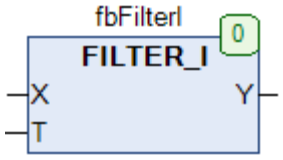


Рис. 19.19. Внешний вид ФБ **FILTER_I** на языке CFC

Функциональный блок **FILTER_I** представляет собой фильтр нижних частот для входного значения **X** типа **INT**. Это может быть полезным при фильтрации шума в сигналах от датчиков. Вход **T** определяет постоянную времени фильтра, которая связана с частотой пропускания следующей формулой: $f_{\text{гр}} = \frac{1}{2\pi \text{T}}$

На выход **Y** подается отфильтрованное значение.

На рис. 19.18 приведена трассировка работы аналогичного по принципу блока **FILTER_DW** (который отличается только типом фильтруемой переменной).

19.11. FILTER_MAV_DW

Тип модуля: ФБ	Переменная	Тип	Описание
Входы	X	DWORD	Фильтруемое значение
	N	UINT	Число точек усреднения (0..31).
	RST	BOOL	Сигнал сброса блока.
Выходы	Y	DWORD	Отфильтрованное значение.
Используемые модули	INC1		

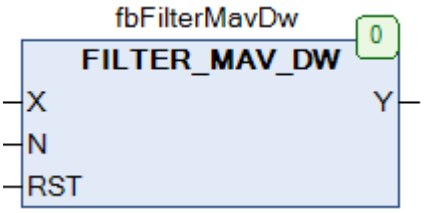


Рис. 19.20. Внешний вид ФБ **FILTER_MAV_DW** на языке CFC

Обратите внимание, что в текущей версии библиотеки ФБ работает **некорректно** (пруф). Для корректной работы необходимо отредактировать область объявления переменных ФБ следующим образом:

```
FILTER_MAV_DW
1  FUNCTION_BLOCK FILTER_MAV_DW
2  VAR_INPUT
3      X : DWORD;
4      N : UINT;
5      RST : BOOL;
6  END_VAR
7  VAR_OUTPUT
8      Y : DWORD;
9  END_VAR
10 VAR
11     init: BOOL;
12     buffer : ARRAY[0..31] OF DWORD; // bug-fix
13     i: INT;
14     sum : DWORD;
15 END_VAR
16 VAR
17     tmp: INT;
18 END_VAR
```

Рис. 19.21. Исправление исходного кода ФБ **FILTER_MAV_DW** для корректной работы

Функциональный блок **FILTER_MAV_DW** представляет собой фильтр скользящей средней для входного значения **X** типа **DWORD**. Это может быть полезным для исключения влияния на сигнал случайной составляющей. Вход **N** определяет сохраняемое число значений входа **X** (**N** принадлежит диапазону 0...31; **0** – только текущее значение, **1** – текущее и предыдущее и т.д.) для вычисления текущего отфильтрованного значения. По переднему фронту входа **RST** информация о предыдущих входных значениях удаляется.

На выход **Y** подается отфильтрованное значение, вычисленное по формуле (**K** – номер текущего цикла ПЛК):

$$Y_K = \frac{1}{\mathrm{N}} \sum_{i=K-N}^{K} X_i$$

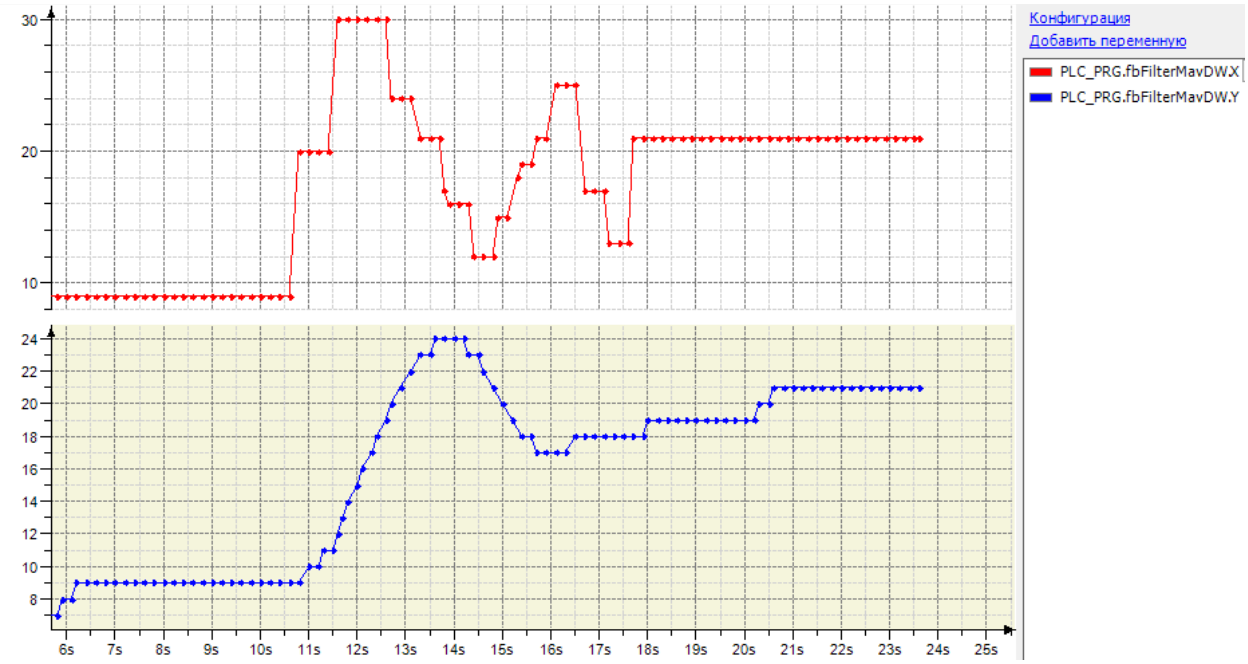


Рис. 19.22. Трассировка работы ФБ **FILTER_MAV_DW** (N=30)

19.12. FILTER_MAV_W

Тип модуля: ФБ	Переменная	Тип	Описание
Входы	X	WORD	Фильтруемое значение
	N	UINT	Число точек усреднения (1..32).
	RST	BOOL	Сигнал сброса блока.
Выходы	Y	WORD	Отфильтрованное значение.
Используемые модули	INC1		

Рис. 19.23. Внешний вид ФБ **FILTER_MAV_W** на языке CFC

Обратите внимание, что в текущей версии библиотеки ФБ работает **некорректно** (пруф). Для корректной работы необходимо отредактировать код ФБ следующим образом:

```
FILTER_MAV_W
1  FUNCTION_BLOCK FILTER_MAV_W
2  VAR_INPUT
3      X : WORD;
4      N : UINT;
5      RST : BOOL;
6  END_VAR
7  VAR_OUTPUT
8      Y : WORD;
9  END_VAR
10 VAR
11     init: BOOL;
12     buffer : ARRAY[0..31] OF WORD; // buf-fix
13     i: INT;
14     sum : DWORD;
15 END_VAR
16 VAR
17     tmp : INT;
18 END VAR
```

Рис. 19.24. Исправление исходного кода **ФБ FILTER_MAV_W** для корректной работы

Функциональный блок FILTER_MAV_W представляет собой фильтр скользящей средней для входного значения **X** типа **WORD**. Принцип работы блока полностью соответствует блоку FILTER_MAV_DW, единственным отличием является тип фильтруемого значения.

19.13. FILTER_W

Тип модуля: ФБ	Переменная	Тип	Описание
Входы	X	WORD	Фильтруемое значение
	T	TIME	Постоянная времени фильтра.
Выходы	Y	WORD	Отфильтрованное значение.
Используемые модули	T_PLC_MS		

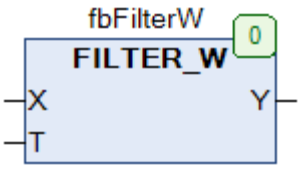


Рис. 19.25. Внешний вид ФБ **FILTER_W** на языке CFC

Функциональный блок **FILTER_W** представляет собой фильтр нижних частот для входного значения **X** типа **WORD**. Это может быть полезным при фильтрации шума в сигналах от датчиков. Принцип работы блока полностью соответствует блоку FILTER_DW, единственным отличием является тип фильтруемого значения.

19.14. FILTER_WAV

Тип модуля: ФБ	**Переменная**	**Тип**	**Описание**
Входы	X	REAL	Фильтруемое значение.
	W	ARRAY [0..15] OF REAL	Массив коэффициентов.
	RST	BOOL	Сигнал сброса блока.
Выходы	Y	REAL	Отфильтрованное значение.
Используемые модули	INC1, DEC1		

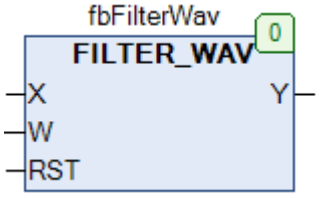


Рис. 19.26. Внешний вид ФБ **FILTER_WAV** на языке CFC

Функциональный блок **FILTER_WAV** представляет собой КИХ-фильтр для входного значения **X** типа **REAL**. Вход **W** определяет весовые коэффициенты для входных значений предыдущих 16-ти циклов ПЛК. На выход **Y** подается отфильтрованное значение, вычисленное по формуле (**K** – номер текущего цикла ПЛК):

$$Y_K = \sum_{i=K-16}^{K} W_i \cdot X_i$$

По переднему фронту входа **RST** информация о предыдущих входных значениях удаляется.

19.15. MIX

Тип модуля: функция	Переменная	Тип	Описание
Входы	A	REAL	Значение 1.
	B	REAL	Значение 2.
	M	REAL	Коэффициент смешения.
Выходы	MIX	REAL	Вычисленное значение.

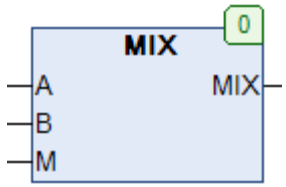


Рис. 19.26. Внешний вид функции **MIX** на языке CFC

Функция **MIX** возвращается значение, вычисленное по формуле:

$$\mathrm{MIX} = (\mathrm{M} - 1) \cdot \mathrm{A} + \mathrm{M} \cdot \mathrm{B}$$

Подразумевается, что значение коэффициента смешения **M** должно принадлежать диапазона **[0,1]**.

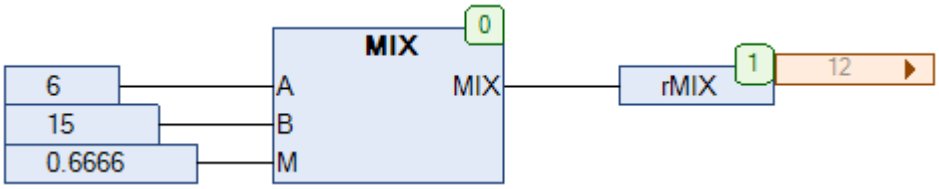


Рис. 19.27. Пример работы с функцией **MIX** на языке CFC

19.16. MUX_R2

Тип модуля: функция	Переменная	Тип	Описание
Входы	IN0	REAL	Информационный вход 0.
	IN1	REAL	Информационный вход 1.
	A	BOOL	Адресный вход.
Выходы	MUX_R2	REAL	Выход мультиплексора.

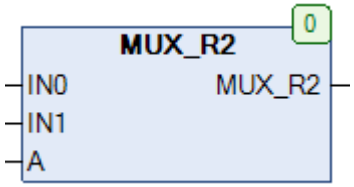


Рис. 19.28. Внешний вид функции **MUX_R2** на языке CFC

Функция **MUX_2R** реализует мультиплексор для двух **REAL** переменных. Принцип работы функции:

- если A=FALSE, то MUX_2R:=IN0;
- если A=TRUE, то MUX_2R:=IN1.

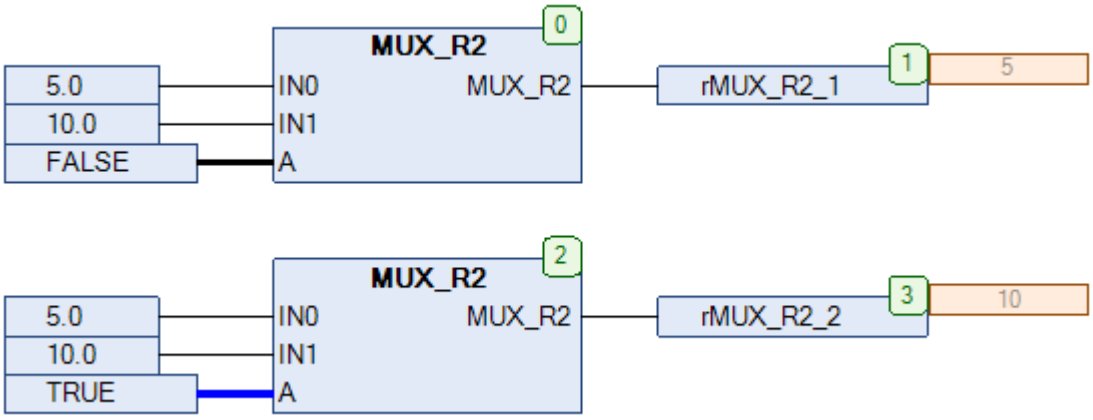


Рис. 19.29. Пример работы с функцией **MUX_R2** на языке CFC

19.17. MUX_R4

Тип модуля: функция	Переменная	Тип	Описание
Входы	IN0	BOOL	Информационный вход 0.
	IN1	BOOL	Информационный вход 1.
	IN2	BOOL	Информационный вход 2.
	IN3	BOOL	Информационный вход 3.
	A0	BOOL	Адресный вход 0.
	A1	BOOL	Адресный вход 1.
Выходы	MUX_R4	REAL	Выход мультиплексора.

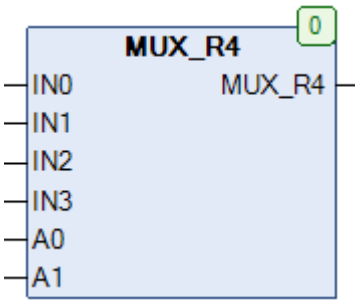


Рис. 19.30. Внешний вид функции **MUX_R4** на языке CFC

Функция **MUX_4R** реализует мультиплексор для четырех **REAL** переменных. Принцип работы функции:

- если A0=FALSE и A1=FALSE, то MUX_4R:=IN0;
- если A0=TRUE и A1=FALSE, то MUX_4R:=IN1;
- если A0=FALSE и A1=TRUE то MUX_4R:=IN2;
- если A0=TRUE и A1=TRUE, то MUX_4R:=IN3.

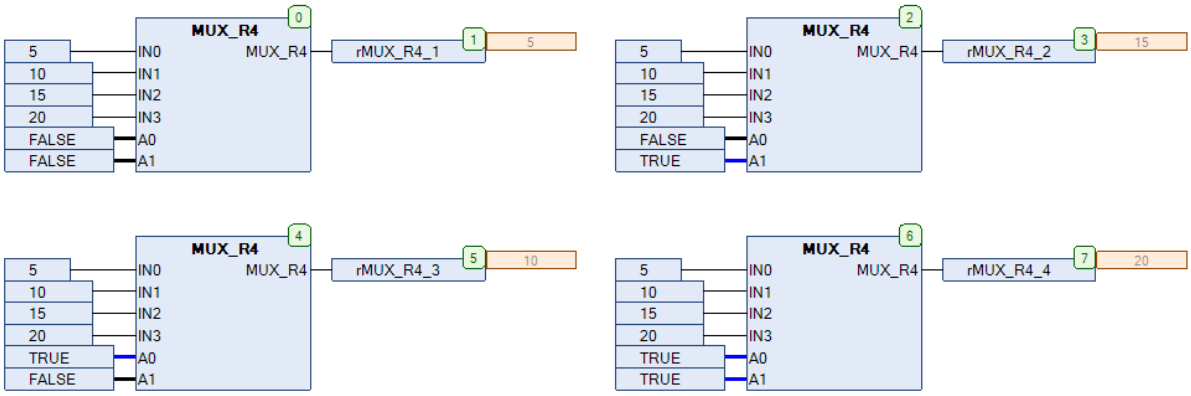


Рис. 19.31. Пример работы с функцией **MUX_R4** на языке CFC

19.18. OFFSET

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	X	REAL	Исходное значение.
	O1	BOOL	Режим «смещение 1».
	O2	BOOL	Режим «смещение 2».
	O3	BOOL	Режим «смещение 3».
	O4	BOOL	Режим «смещение 4».
	D	BOOL	Режим «имитация входа».
	Offset_1	REAL	Коэффициент смещения 1.
	Offset_1	REAL	Коэффициент смещения 2.
	Offset_1	REAL	Коэффициент смещения 3.
	Offset_1	REAL	Коэффициент смещения 4.
	default	REAL	Значение входа в имитации.
Выходы	OFFSET	REAL	Смещенное значение.

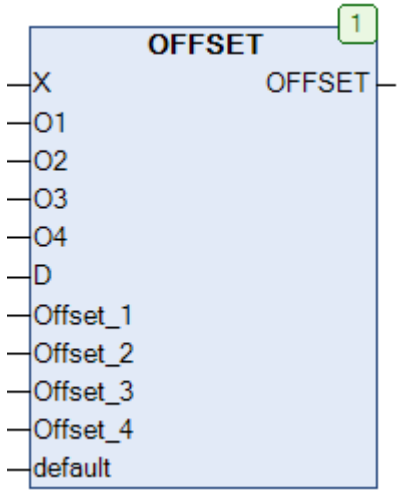


Рис. 19.32. Внешний вид функции **OFFSET** на языке CFC

Функция **OFFSET** возвращает смещенное значение входной переменной **X**. Входы **O1…O4** определяют, какие коэффициенты смещения будут прибавляться к входному значению, а входы **Offset_1…Offset_4** определяют величины этих коэффициентов. Если одновременно будут активны несколько входов, то к входному значению будут прибавлены величины нескольких коэффициентов. Если вход **D** имеет значение **TRUE**, то функция вместо **X** использует значение **default**.

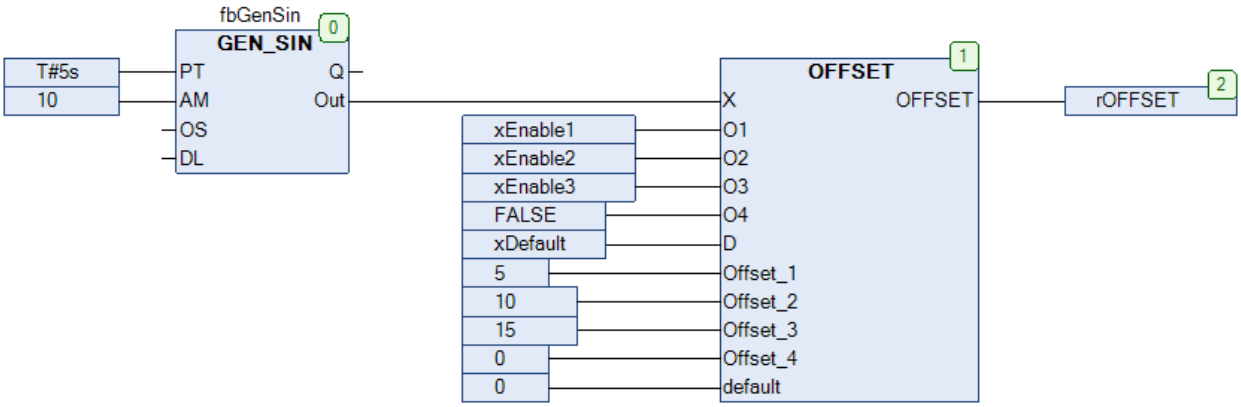


Рис. 19.33. Пример работы с функцией **OFFSET** на языке CFC

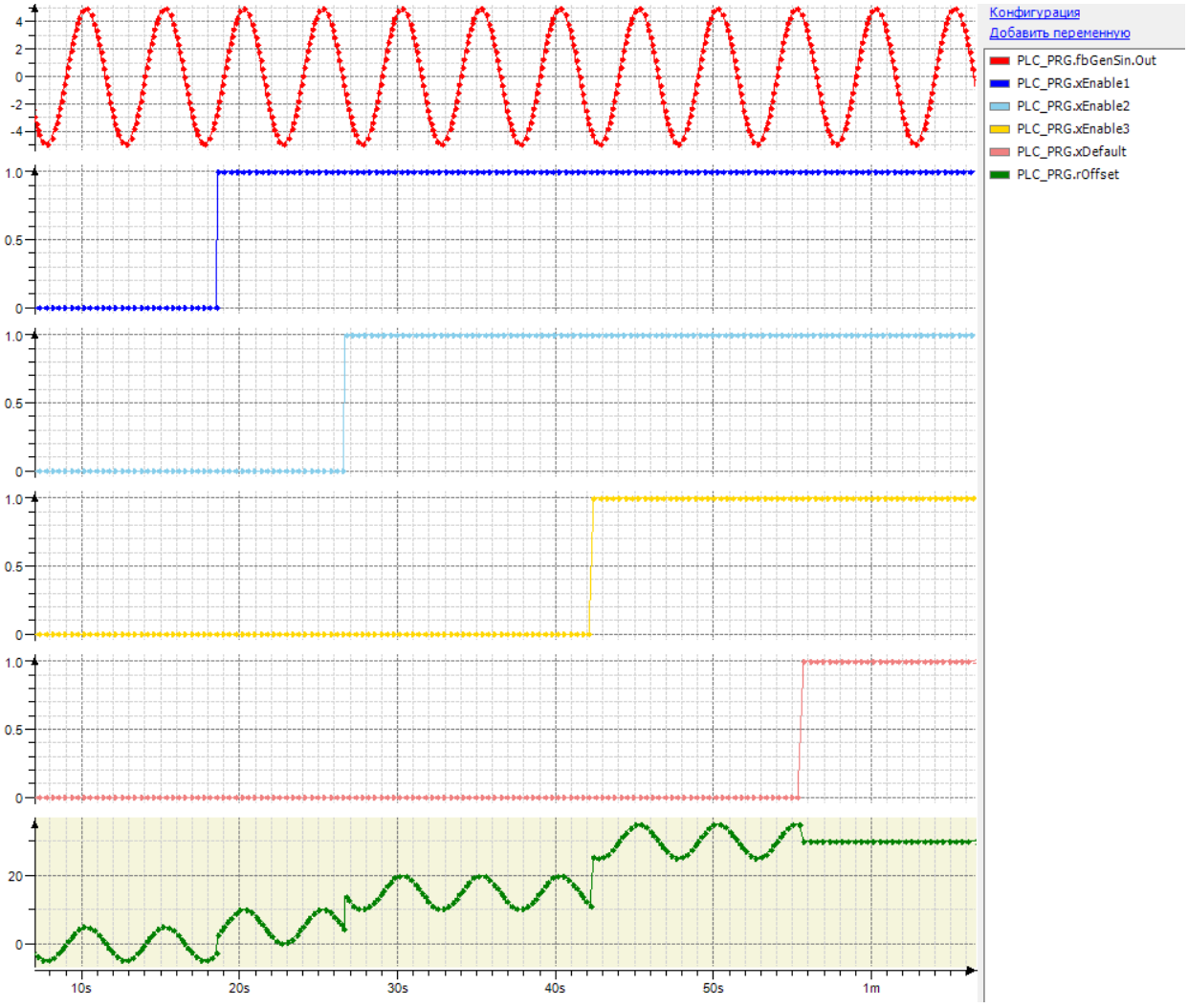


Рис. 19.34. Трассировка работы функции **OFFSET**

19.19. OFFSET2

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	X	REAL	Исходное значение.
	O1	BOOL	Режим «смещение 1».
	O2	BOOL	Режим «смещение 2».
	O3	BOOL	Режим «смещение 3».
	O4	BOOL	Режим «смещение 4».
	D	BOOL	Режим «имитация входа».
	Offset_1	REAL	Коэффициент смещения 1.
	Offset_1	REAL	Коэффициент смещения 2.
	Offset_1	REAL	Коэффициент смещения 3.
	Offset_1	REAL	Коэффициент смещения 4.
	default	REAL	Значение входа в имитации.
Выходы	OFFSET2	REAL	Смещенное значение.

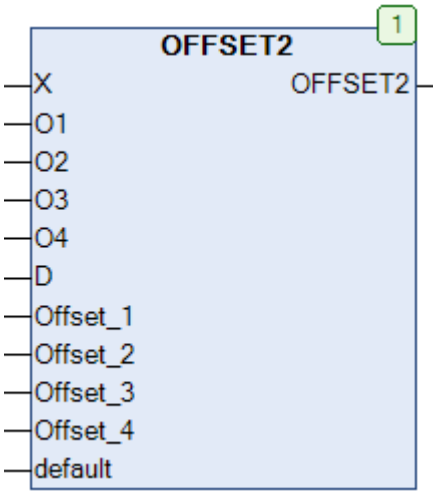


Рис. 19.35. Внешний вид функции **OFFSET2** на языке CFC

Функция **OFFSET2** возвращает смещенное значение входной переменной **X**. Входы **O1…O4** определяют, какие коэффициенты смещения будут прибавляться к входному значению, а входы **Offset_1…Offset_4** определяют величины этих коэффициентов. Если одновременно будут активны несколько входов, то к входному значению будет прибавлен коэффициент входа с наибольшим номером (в отличие от функции **OFSFSET**, в которой коэффициенты складываются). Если вход **D** имеет значение **TRUE**, то функция вместо X использует значение **default**.

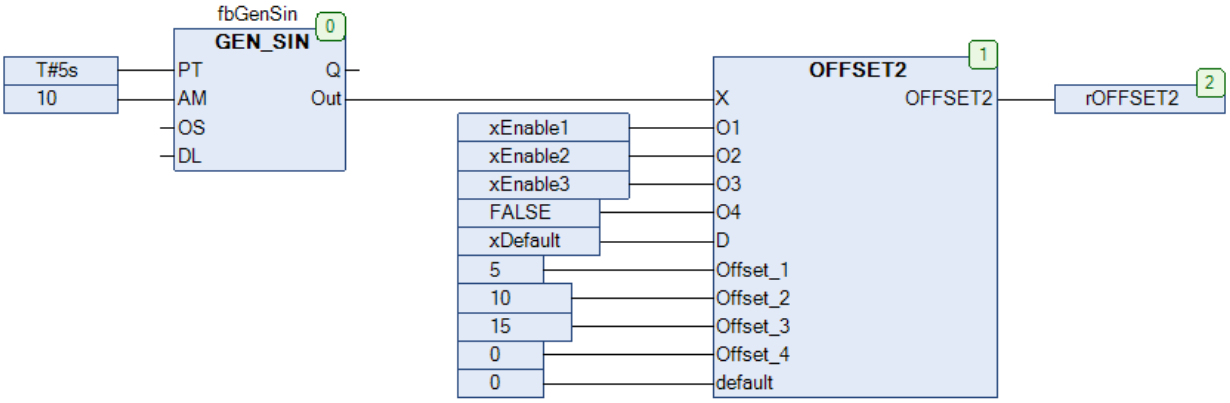


Рис. 19.36. Пример работы с функцией **OFFSET2** на языке CFC

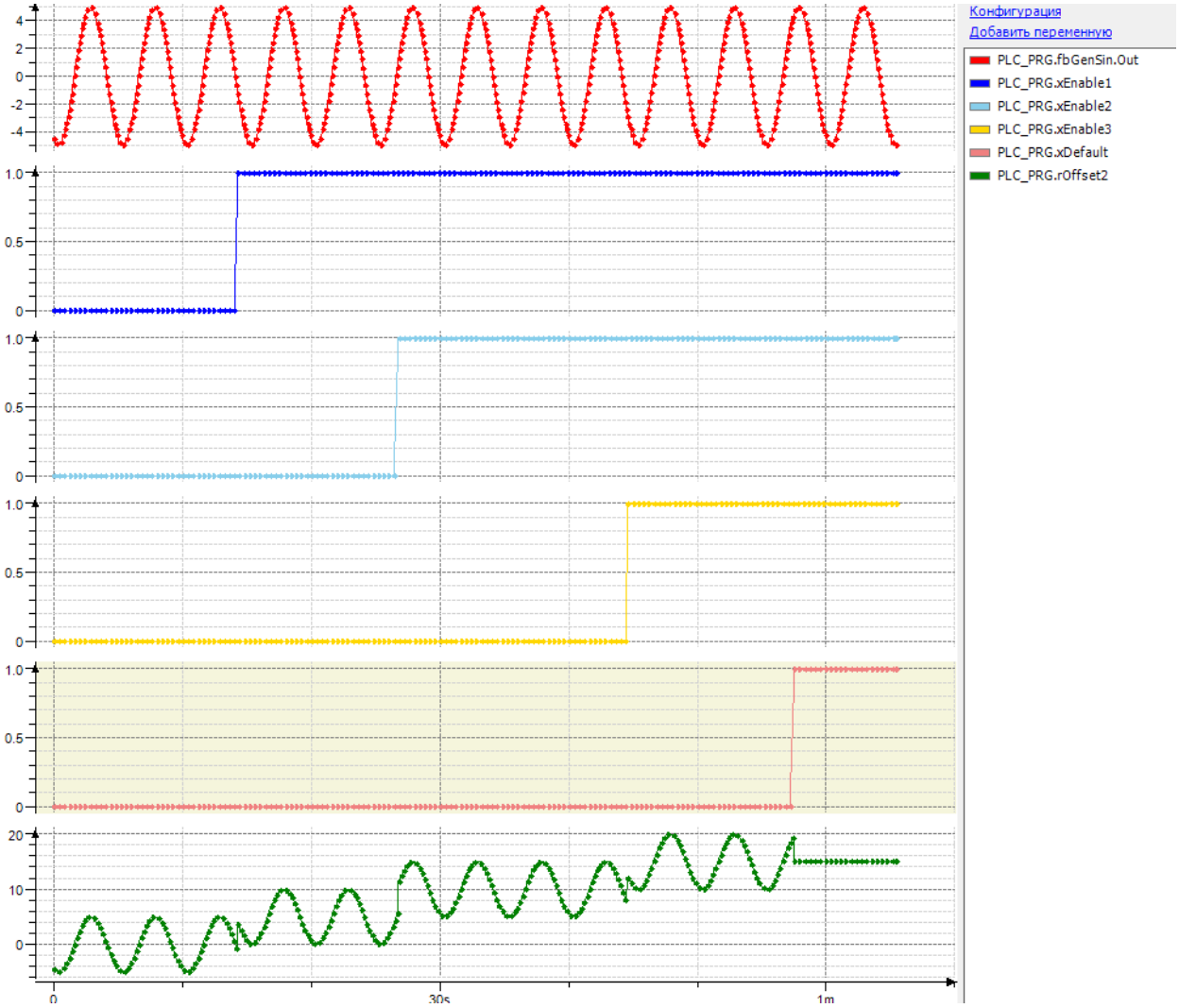


Рис. 19.37. Трассировка работы функции **OFFSET2**

19.20. OVERRIDE

Тип модуля: функция	Переменная	Тип	Описание
Входы	X1	REAL	Исходное значение 1.
	X2	REAL	Исходное значение 2.
	X3	REAL	Исходное значение 3.
	E1	BOOL	Активация значения 1.
	E2	BOOL	Активация значения 2.
	E3	BOOL	Активация значения 3.
Выходы	OVERRIDE	REAL	Наибольшее активное значение.

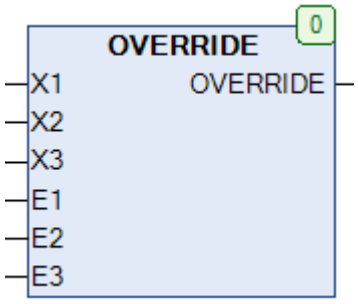


Рис. 19.38. Внешний вид функции **OVERRIDE** на языке CFC

Функция **OVERRIDE** возвращает наибольшее по модулю из активных значений **X1…X3**. Активность значения определяется входным сигналом **E1…E3** (**TRUE** – значение активно). Если активных значений нет, то функция возвращает ноль.

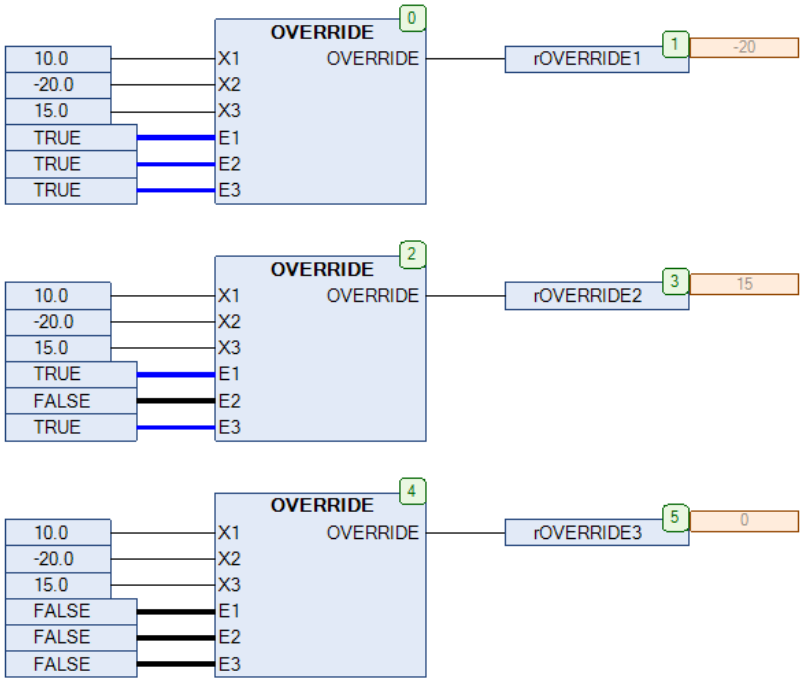


Рис. 19.39. Пример работы с функцией **OVERRIDE** на языке CFC

19.21. RANGE_TO_BYTE

Тип модуля: функция	Переменная	Тип	Описание
Входы	X	REAL	Исходное значение.
	low	REAL	Нижний предел исходного значения.
	high	REAL	Верхний предел исходного значения.
Выходы	RANGE_TO_BYTE	BYTE	Отмасштабированное значение.

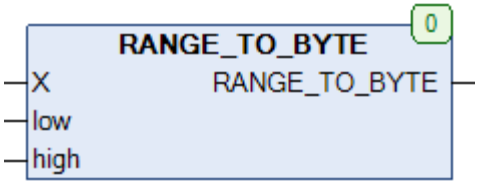


Рис. 19.40. Внешний вид функции **RANGE_TO_BYTE** на языке CFC

Функция **RANGE_TO_BYTE** линейное масштабирует входное значение с плавающей точкой **X** из диапазона (**low…high**) в целочисленное значение типа **BYTE** из диапазона (**0…255**). См. также обратную функцию BYTE_TO_RANGE.

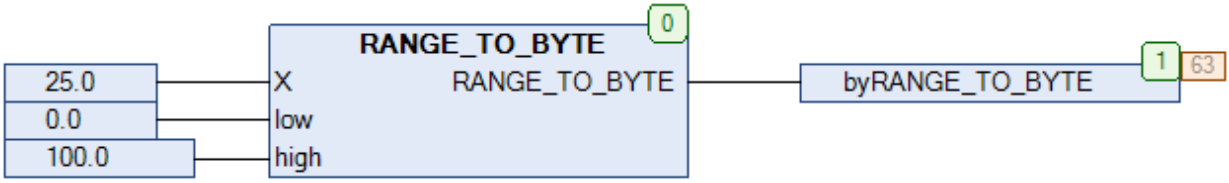


Рис. 19.41. Пример работы с функцией **RANGE_TO_BYTE** на языке CFC

19.22. RANGE_TO_WORD

Тип модуля: функция	Переменная	Тип	Описание
Входы	X	REAL	Исходное значение.
	low	REAL	Нижний предел исходного значения.
	high	REAL	Верхний предел исходного значения.
Выходы	RANGE_TO_WORD	WORD	Отмасштабированное значение.

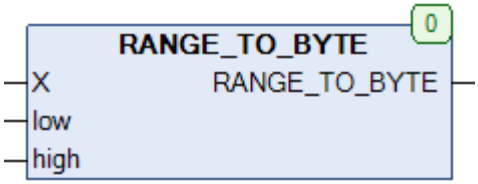


Рис. 19.42. Внешний вид функции **RANGE_TO_WORD** на языке CFC

Функция **RANGE_TO_WORD** линейное масштабирует входное значение с плавающей точкой **X** из диапазона (**low…high**) в целочисленное значение типа **WORD** из диапазона (**0…65535**). См. также обратную функцию WORD_TO_RANGE.

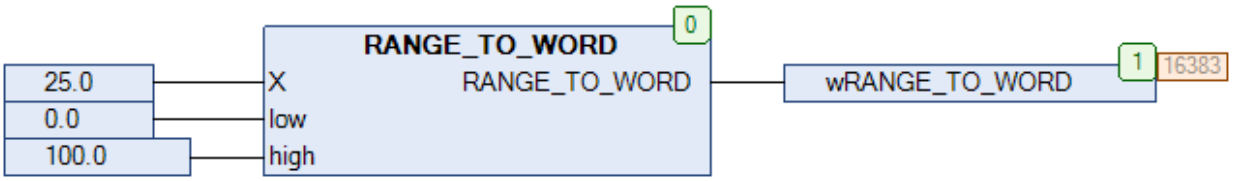


Рис. 19.43. Пример работы с функцией **RANGE_TO_WORD** на языке CFC

19.23. SCALE

Тип модуля: функция	Переменная	Тип	Описание
Входы	X	REAL	Исходное значение.
	K	REAL	Коэффициент масштабирования.
	O	REAL	Сдвиг.
	MX	REAL	Верхний предел отмасштабированного значения.
	MN	REAL	Нижний предел отмасштабированного значения.
Выходы	SCALE	REAL	Отмасштабированное значение.

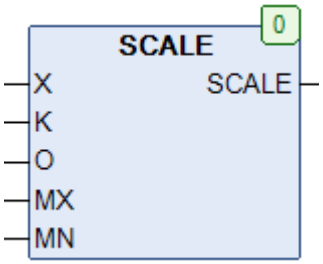


Рис. 19.44. Внешний вид функции **SCALE** на языке CFC

Функция **SCALE** возвращает значение линейной функции с коэффициентом **K** и сдвигом **O** для аргумента **X**. Входы **MN** и **MX** определяют нижний и верхний предел возвращаемого значения.

$$\mathrm{SCALE} = \mathrm{LIMIT}(\mathrm{MN}, \mathrm{K} \cdot \mathrm{X} + \mathrm{O}, \mathrm{MX})$$

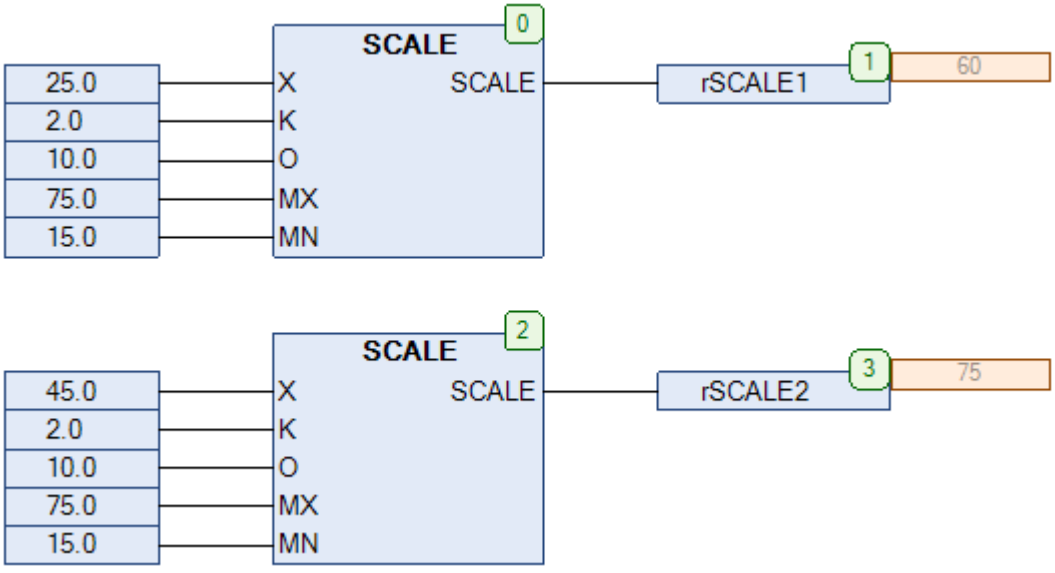


Рис. 19.45. Пример работы с функцией **SCALE** на языке CFC

19.24. SCALE_B

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	X	BYTE	Исходное значение.
	I_LO	BYTE	Нижний предел исходного значения.
	I_HI	BYTE	Верхний предел исходного значения.
	O_LO	REAL	Нижний предел отмасштабированного значения.
	O_HI	REAL	Верхний предел отмасштабированного значения.
Выходы	SCALE_B	REAL	Отмасштабированное значение.

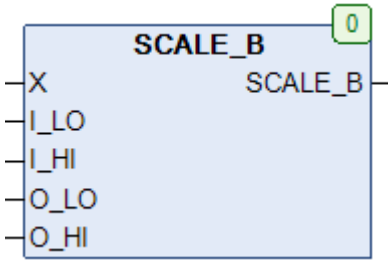


Рис. 19.46. Внешний вид функции **SCALE_B** на языке CFC

Функция **SCALE_B** линейно масштабирует входное целочисленное значение **X** из диапазона (**I_LO…I_HI**) в значение с плавающей точкой из диапазона (**O_LO…O_HI**).

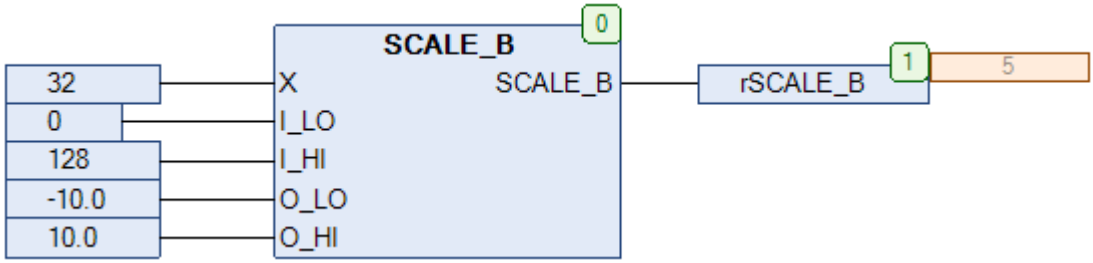


Рис. 19.47. Пример работы с функцией **SCALE_B** на языке CFC

19.25. SCALE_B2

Тип модуля: функция	Переменная	Тип	Описание
Входы	IN1	BYTE	Исходное значение 1 .
	IN2	BYTE	Исходное значение 2.
	K	REAL	Коэффициент.
	O	REAL	Сдвиг.
Выходы	SCALE_B2	REAL	Рассчитанное значение.
Параметры	IN1_MIN	REAL	Нижний предел отмасштабированного значения 1.
	IN1_MAX	REAL	Верхний предел отмасштабированного значения 1.
	IN2_MIN	REAL	Нижний предел отмасштабированного значения 2.
	IN2_MAX	REAL	Верхний предел отмасштабированного значения 2.

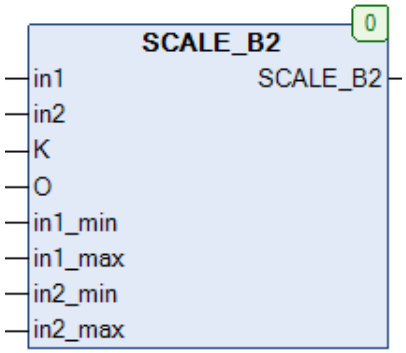


Рис. 19.48. Внешний вид функции **SCALE_B2** на языке CFC

Функция **SCALE_B2** линейно масштабирует входное целочисленное значение **IN#** из диапазона (**0…255**) в значение с плавающей точкой из диапазона (**IN#_MIN…IN#_MAX**), суммирует их и возвращает для суммы значение линейного функции с коэффициентом **K** и сдвигом **O**.

$$\mathrm{SCALE\_B2} = \mathrm{O} + \mathrm{K} \cdot \sum_{\mathrm{i}=1}^{2} \mathrm{IN_i}$$

Это может использовано, например, для расчета общего количества воздуха в системе вентиляции.

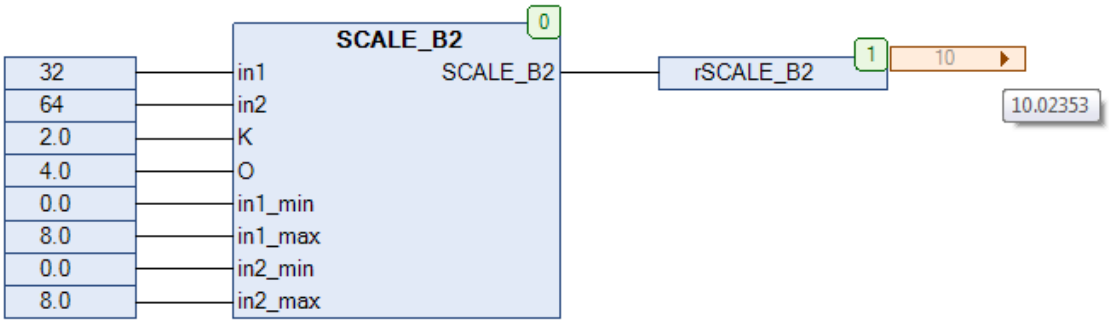


Рис. 19.49. Пример работы с функцией **SCALE_B2** на языке CFC

19.26. SCALE_B4

Тип модуля: функция	Переменная	Тип	Описание
Входы	IN1…IN4	BYTE	Исходное значение 1…4.
	K	REAL	Коэффициент.
	O	REAL	Сдвиг.
Выходы	SCALE_B4	REAL	Рассчитанное значение.
Параметры	IN1_MIN…IN4_MIN	REAL	Нижний предел отмасштабированного значения 1…4.
	IN1_MAX…IN4_MAX	REAL	Верхний предел отмасштабированного значения 1…4.

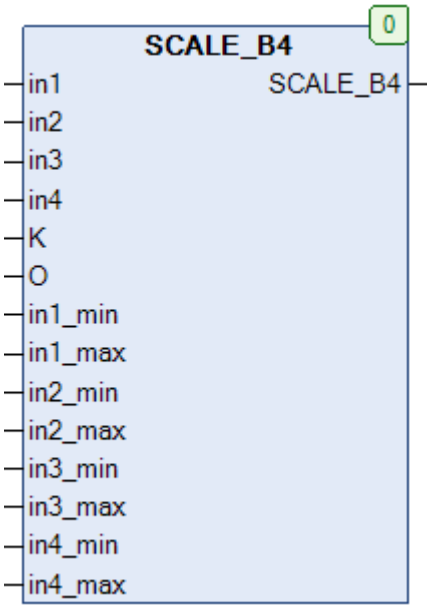


Рис. 19.50. Внешний вид функции **SCALE_B4** на языке CFC

Функция **SCALE_B4** линейно масштабирует входное целочисленное значение I**N#** из диапазона (**0…255**) в значение с плавающей точкой из диапазона (**IN#_MIN…IN#_MAX**), суммирует их и возвращает для суммы значение линейного функции с коэффициентом **K** и сдвигом **O**.

$$\mathrm{SCALE\_B4} = \mathrm{O} + \mathrm{K} \cdot \sum_{\mathrm{i}=1}^{4} \mathrm{IN_i}$$

Это может использовано, например, для расчета общего количества воздуха в системе вентиляции.

19.27. SCALE_B8

Тип модуля: функция	Переменная	Тип	Описание
Входы	IN1…IN8	BYTE	Исходное значение 1…8.
	K	REAL	Коэффициент.
	O	REAL	Сдвиг.
Выходы	SCALE_B8	REAL	Рассчитанное значение.
Параметры	IN1_MIN…IN8_MIN	REAL	Нижний предел отмасштабированного значения 1…8.
	IN1_MAX…IN8_MAX	REAL	Верхний предел отмасштабированного значения 1…8.

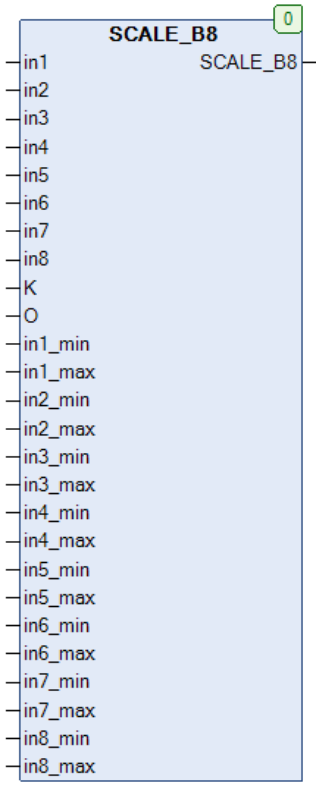


Рис. 19.51. Внешний вид функции **SCALE_B8** на языке CFC

Функция **SCALE_B8** линейно масштабирует входное целочисленное значение **IN#** из диапазона (**0…255**) в значение с плавающей точкой из диапазона (**IN#_MIN…IN#_MAX**), суммирует их и возвращает для суммы значение линейного функции с коэффициентом **K** и сдвигом **O**.

$$\mathrm{SCALE\_B8} = \mathrm{O} + \mathrm{K} \cdot \sum_{\mathrm{i}=1}^{8} \mathrm{IN_i}$$

Это может использовано, например, для расчета общего количества воздуха в системе вентиляции.

19.28. SCALE_D

Тип модуля: функция	Переменная	Тип	Описание
Входы	X	DWORD	Исходное значение.
	I_LO	DWORD	Нижний предел исходного значения.
	I_HI	DWORD	Верхний предел исходного значения.
	O_LO	REAL	Нижний предел отмасштабированного значения.
	O_HI	REAL	Верхний предел отмасштабированного значения.
Выходы	SCALE_D	REAL	Отмасштабированное значение.

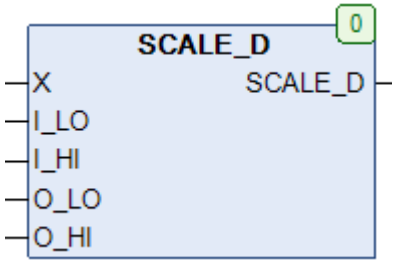


Рис. 19.52. Внешний вид функции **SCALE_D** на языке CFC

Функция **SCALE_D** линейно масштабирует входное целочисленное значение **X** из диапазона (**I_LO…I_HI**) в значение с плавающей точкой из диапазона (**O_LO…O_HI**).

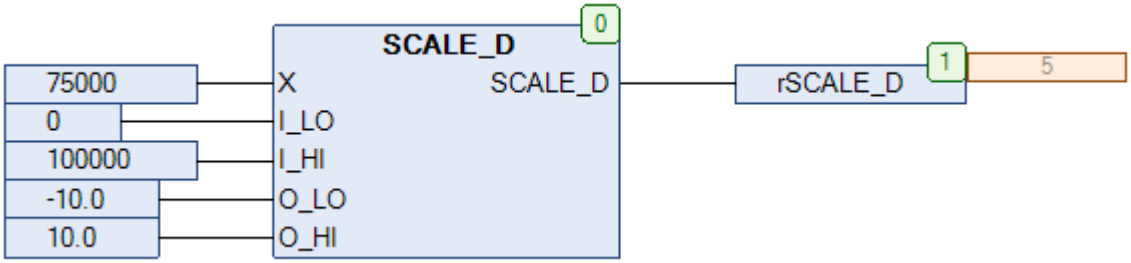


Рис. 19.53. Пример работы с функцией **SCALE_D** на языке CFC

19.29. SCALE_R

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	X	REAL	Исходное значение.
	I_LO	REAL	Нижний предел исходного значения.
	I_HI	REAL	Верхний предел исходного значения.
	O_LO	REAL	Нижний предел отмасштабированного значения.
	O_HI	REAL	Верхний предел отмасштабированного значения.
Выходы	SCALE_R	REAL	Отмасштабированное значение.

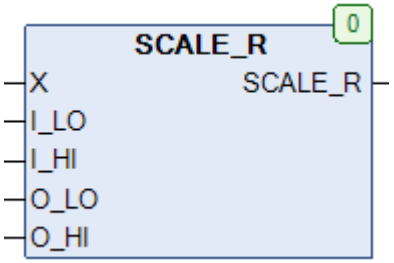


Рис. 19.54. Внешний вид функции **SCALE_R** на языке CFC

Функция **SCALE_R** линейно масштабирует входное значение с плавающей точкой **X** из диапазона (**I_LO…I_HI**) в диапазон (**O_LO…O_HI**).

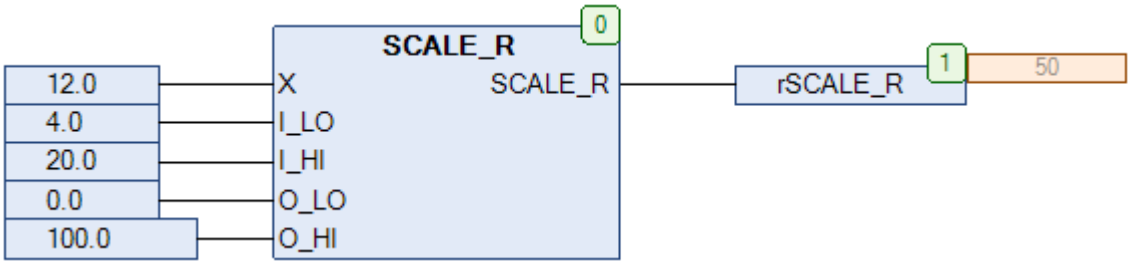


Рис. 19.55. Пример работы с функцией **SCALE_R** на языке CFC

19.30. SCALE_X2

Тип модуля: функция	Переменная	Тип	Описание
Входы	IN1	BOOL	Исходное значение 1 .
	IN2	BOOL	Исходное значение 2.
	K	REAL	Коэффициент.
	O	REAL	Сдвиг.
Выходы	SCALE_X2	REAL	Отмасштабированное значение.
Параметры	IN1_MIN	REAL	Отмасштабированное значение 1 для IN1=FALSE.
	IN1_MAX	REAL	Отмасштабированное значение 1 для IN1=TRUE.
	IN2_MIN	REAL	Отмасштабированное значение 2 для IN2=FALSE.
	IN2_MAX	REAL	Отмасштабированное значение 2 для IN2=TRUE.

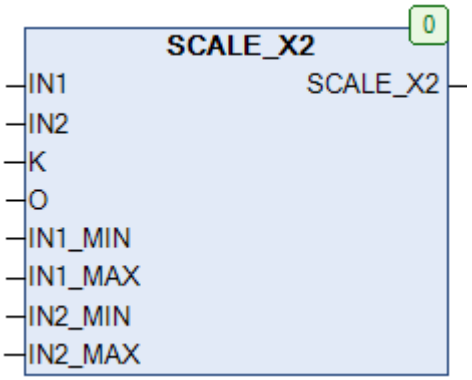


Рис. 19.56. Внешний вид функции **SCALE_X2** на языке CFC

Функция **SCALE_X2** преобразует входное булевское значение **IN#** в значение с плавающей точкой (**IN#_MIN** для **IN#=FALSE** и **IN#_MAX** для **IN#=TRUE**), суммирует их и возвращает для суммы значение линейного функции с коэффициентом **K** и сдвигом **O**.

$$\mathrm{SCALE\_X2} = \mathrm{O} + \mathrm{K} \cdot \sum_{\mathrm{i}=1}^{2} [\mathrm{IN\_MIN_i} + \mathrm{IN_i} \cdot (\mathrm{IN\_MAX_i} - \mathrm{IN\_MIN_i})]$$

Это может использовано, например, для расчета общего количества воздуха в системе вентиляции.

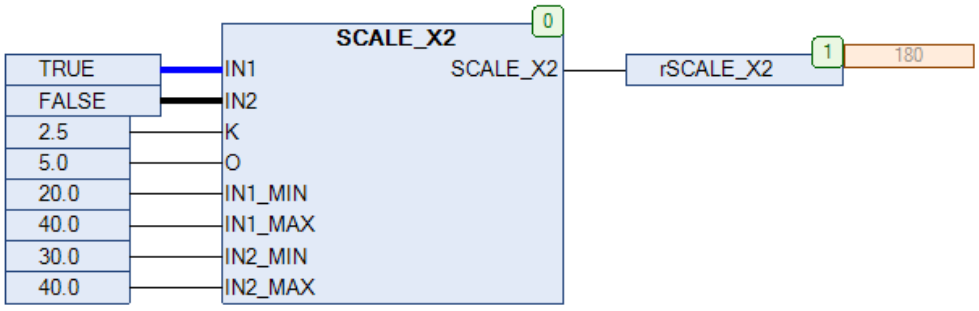


Рис. 19.57. Пример работы с функцией **SCALE_X2** на языке CFC

19.31. SCALE_X4

Тип модуля: функция	Переменная	Тип	Описание
Входы	IN1…IN4	BYTE	Исходное значение 1…4.
	K	REAL	Коэффициент.
	O	REAL	Сдвиг.
Выходы	SCALE_X4	REAL	Рассчитанное значение.
Параметры	IN1_MIN…IN4_MIN	REAL	Отмасштабированное значение для IN1…IN4=FALSE.
	IN1_MAX…IN4_MAX	REAL	Отмасштабированное значение для IN1…IN4=TRUE.

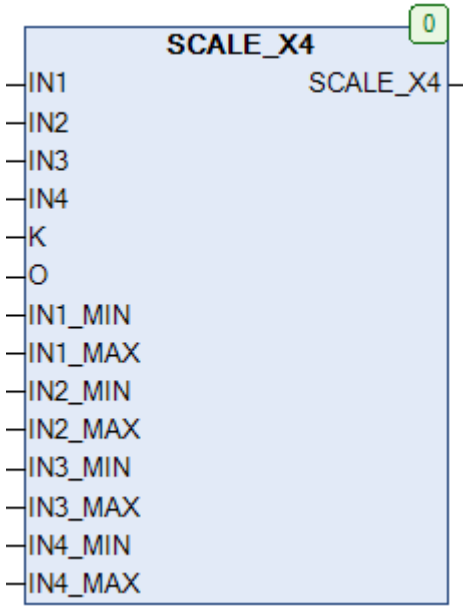


Рис. 19.58. Внешний вид функции **SCALE_X4** на языке CFC

Функция **SCALE_X4** преобразует входное булевское значение **IN#** в значение с плавающей точкой (**IN#_MIN** для **IN#=FALSE** и **IN#_MAX** для **IN#=TRUE**), суммирует их и возвращает для суммы значение линейного функции с коэффициентом **K** и сдвигом **O**.

$$\mathrm{SCALE\_X4} = \mathrm{O} + \mathrm{K} \cdot \sum_{\mathrm{i}=1}^{4} [\mathrm{IN\_MIN_i} + \mathrm{IN_i} \cdot (\mathrm{IN\_MAX_i} - \mathrm{IN\_MIN_i})]$$

Это может использовано, например, для расчета общего количества воздуха в системе вентиляции.

19.32. SCALE_X8

Тип модуля: функция	Переменная	Тип	Описание
Входы	IN1…IN8	BYTE	Исходное значение 1…8.
	K	REAL	Коэффициент.
	O	REAL	Сдвиг.
Выходы	SCALE_X8	REAL	Рассчитанное значение.
Параметры	IN1_MIN…IN8_MIN	REAL	Отмасштабированное значение для IN1…IN8=FALSE.
	IN1_MAX…IN8_MAX	REAL	Отмасштабированное значение для IN1…IN8=TRUE.

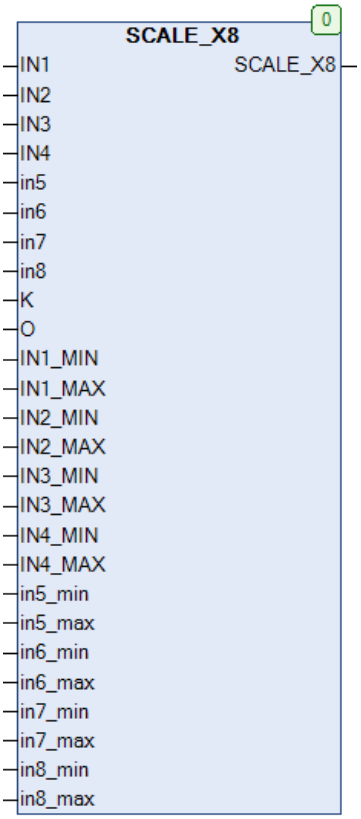


Рис. 19.58. Внешний вид функции SCALE_X8 на языке CFC

Функция **SCALE_X8** преобразует входное булевское значение **IN#** в значение с плавающей точкой (**IN#_MIN** для **IN#=FALSE** и **IN#_MAX** для **IN#=TRUE**), суммирует их и возвращает для суммы значение линейного функции с коэффициентом **K** и сдвигом **O**.

$$\mathrm{SCALE\_X8} = \mathrm{O} + \mathrm{K} \cdot \sum_{\mathrm{i}=1}^{8} [\mathrm{IN\_MIN_i} + \mathrm{IN_i} \cdot (\mathrm{IN\_MAX_i} - \mathrm{IN\_MIN_i})]$$

Это может использовано, например, для расчета общего количества воздуха в системе вентиляции.

19.33. SEL2_OF_3

Тип модуля: ФБ	Переменная	Тип	Описание
Входы	IN1	REAL	Контролируемое значение 1.
	IN2	REAL	Контролируемое значение 2.
	IN3	REAL	Контролируемое значение 3.
	D	REAL	Допустимое отклонение.
Выходы	Y	REAL	Усредненное значение.
	W	INT	Код состояния.
	E	BOOL	Флаг ошибки.

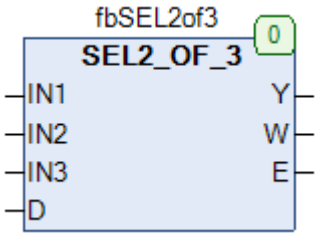


Рис. 19.59. Внешний вид **ФБ SEL2_OF_3** на языке CFC

Функциональный блок **SEL2_OF_3** используется для контроля показаний датчиков по схеме «2 из 3». Если входные значения **IN1…IN3** отличаются друг от друга на величину, не превышающую **D**, то на выход **Y** поступает усредненное по трем сигналам значение, а выход **W=0**. Если одно из входных значений отличается от остальных на величину, превышающую **D**, то на выход **Y** поступает усредненное по двум другим сигналам значение, а на выходе **W** отображается номер входа с некорректным значением. Если все три входных значения отличаются на величину, превышающую **D**, то выход **Y** сохраняет свое последнее значение, выход **W=4**, а выход **E** принимает значение **TRUE**, сигнализируя о сбое в системе.

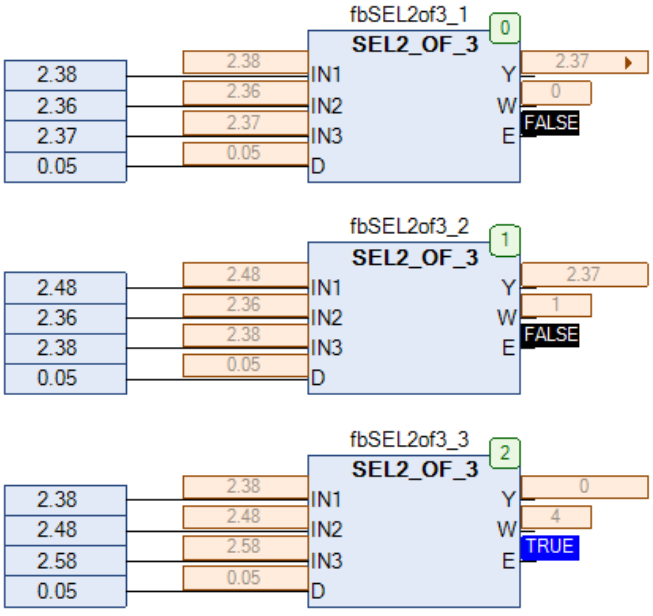


Рис. 19.60. Пример работы с ФБ **SEL2_OF_3** на языке CFC

19.34. SEL2_OF_3B

Тип модуля: ФБ	Переменная	Тип	Описание
Входы	IN1	REAL	Контролируемый сигнал 1.
	IN2	REAL	Контролируемый сигнал 2.
	IN3	REAL	Контролируемый сигнал 3.
	TD	TIME	Время задержки индикации ошибки.
Выходы	Q	BOOL	Выходной сигнал.
	W	BOOL	Флаг «ошибка».

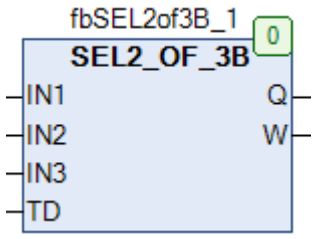


Рис. 19.61. Внешний вид ФБ **SEL2_OF_3B** на языке CFC

Функциональный блок **SEL2_OF_3B** используется для контроля дискретных сигналов по схеме «2 из 3». Если входные значения **IN1…IN3** совпадают, то выход **Q** равен этому же значению. Если одно из входных значений отличается от остальных, то выход **Q** принимает значение, которое имеют два других входа, а выход **W** принимает значение **TRUE**, сигнализируя об ошибке. Вход **TD** определяет задержку между изменением значений входов и активацией выхода **W**.

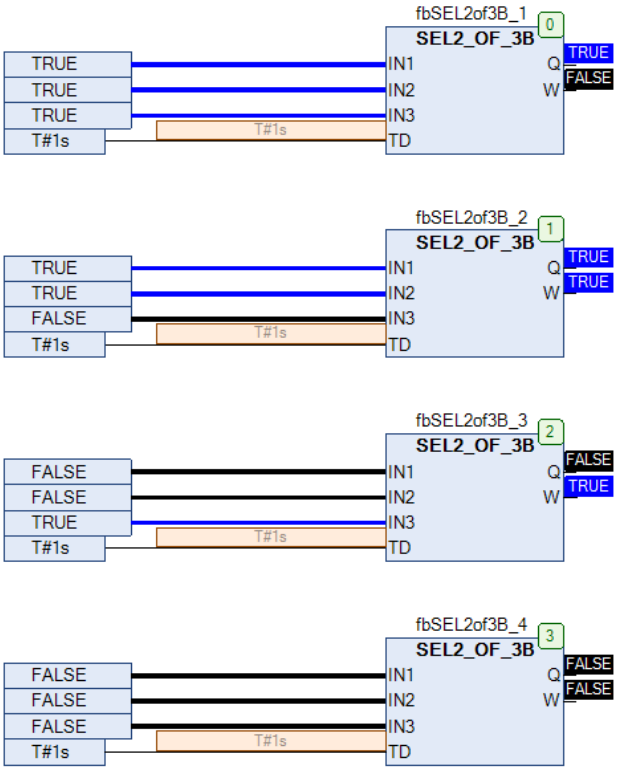


Рис. 19.62. Пример работы с ФБ **SEL2_OF_3B** на языке CFC

19.35. SH

Тип модуля: ФБ	Переменная	Тип	Описание
Входы	in	REAL	Контролируемое значение.
	CLK	BOOL	Сигнал сохранения данных.
Выходы	out	REAL	Сохраненное значение.
	trig	BOOL	Флаг «произошло сохранение».

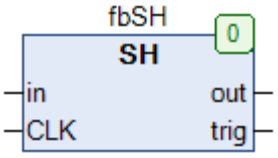


Рис. 19.63. Внешний вид **SH** на языке CFC

Функциональный блок **SH** представляет собой элемент хранения. По переднему фронту на входе **CLK** выход **out** принимает и запоминает значение входа **in**. Выход **trig** принимает значение **TRUE** на один цикл при изменении значения на выходе **out**.

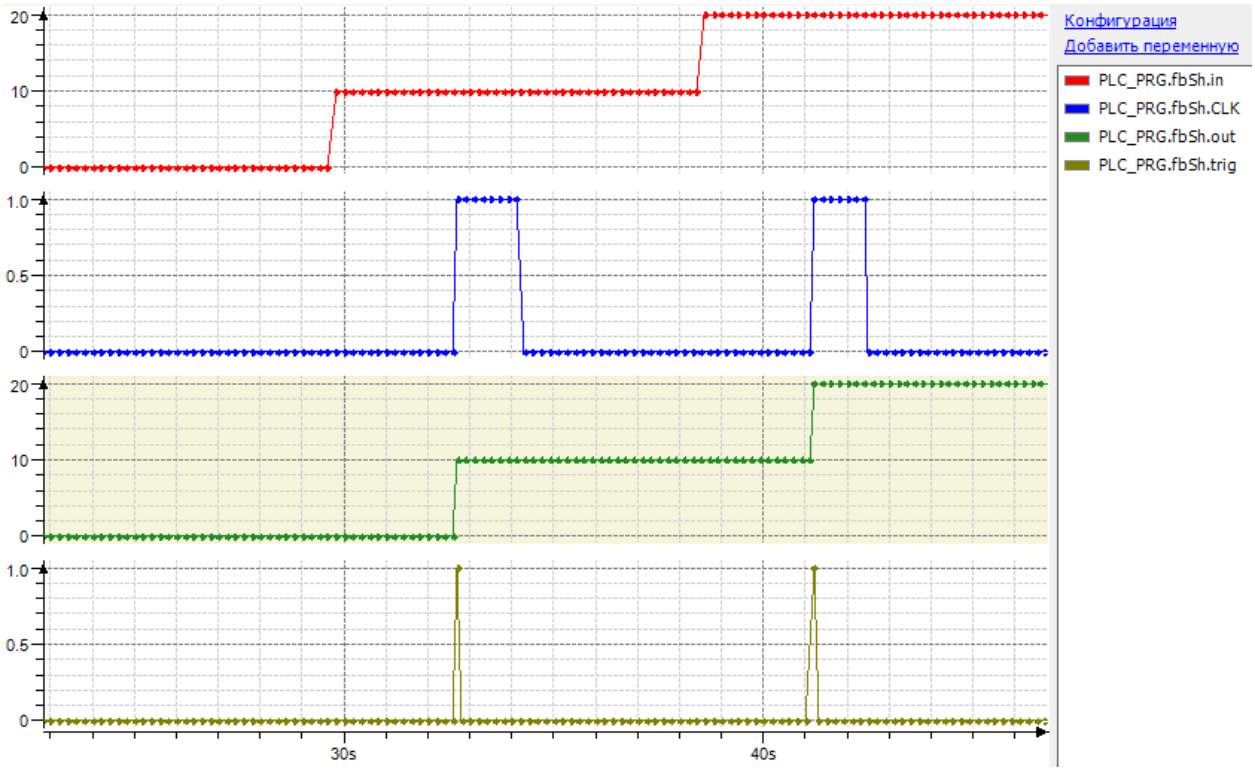


Рис. 19.64. Трассировка работы ФБ **SH**

19.36. SH_1

Тип модуля: ФБ	Переменная	Тип	Описание
Входы	in	REAL	Контролируемое значение.
	PT	TIME	Период сохранения.
Выходы	out	REAL	Сохраненное значение.
	trig	BOOL	Флаг «произошло сохранение».
Используемые модули	T_PLC_MS		

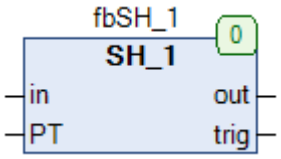


Рис. 19.65. Внешний вид **SH_1** на языке CFC

Функциональный блок **SH_1** представляет собой элемент хранения. Выход **out** принимает и запоминает значение входа **in** с заданным периодом **PT**. Выход **trig** принимает значение **TRUE** на один цикл при каждом сохранении данных.

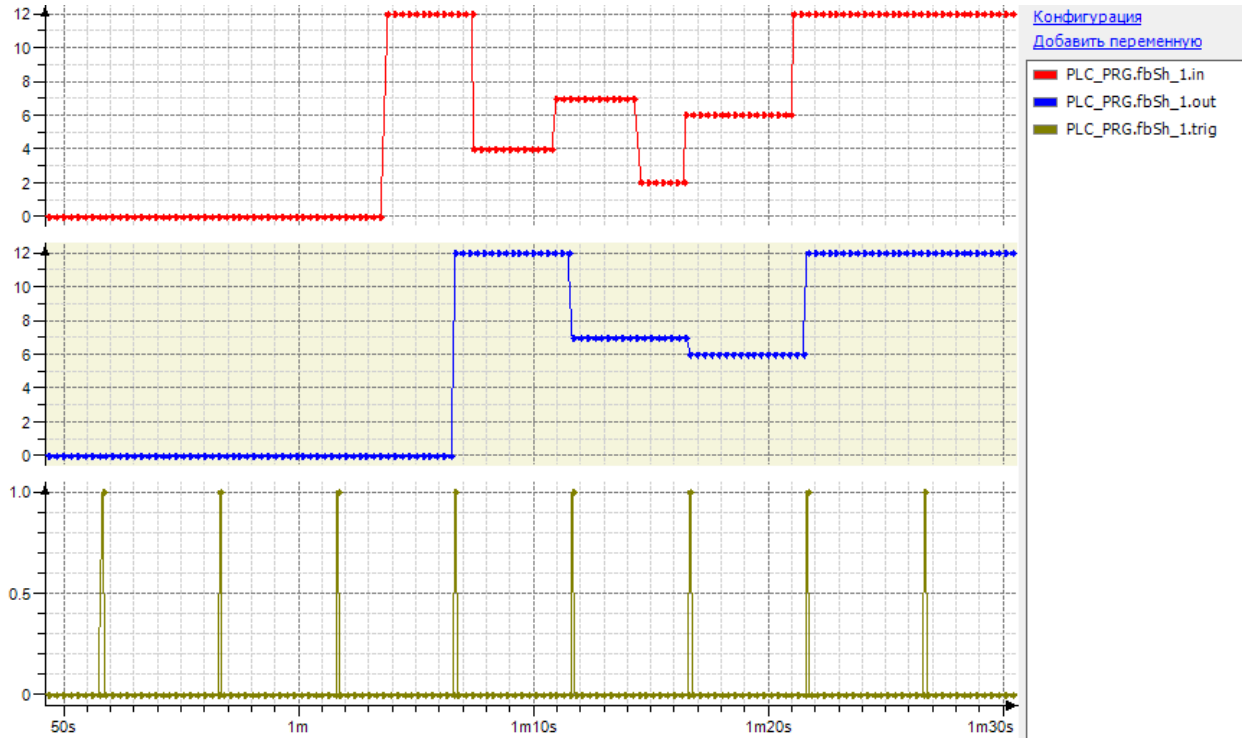


Рис. 19.66. Трассировка работы ФБ **SH_1** (PT=T#5s)

19.37. SH_2

Тип модуля: ФБ	Переменная	Тип	Описание
Входы	in	REAL	Контролируемое значение.
	PT	TIME	Период сохранения.
	N	INT	Число значений для статистики (1..16).
	disc	INT	Число отбрасываемых экстремумов.
Выходы	out	REAL	Сохраненное значение.
	trig	BOOL	Флаг «произошло сохранение».
	avg	REAL	Среднее значение.
	high	REAL	Максимальное значение.
	low	REAL	Минимальное значение.
Используемые модули	T_PLC_MS, EVEN		

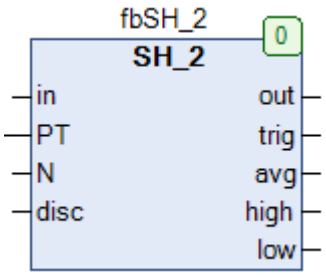


Рис. 19.67. Внешний вид **SH_2** на языке CFC

Функциональный блок **SH_2** представляет собой элемент хранения со сбором статистики. Выход **out** принимает и запоминает значение входа **in** с заданным периодом **PT**. Выход **trig** принимает значение **TRUE** на один цикл при каждом сохранении данных. Вход **N** определяет число срезов (до 16-ти) для сбора статистической информации – среднего (выход **avg**), максимального (выход **high**) и минимального значений (выход **low**). Вход **disc** позволяет определить число экстремумов, которые не будут учитываться при статистической обработке (это может быть полезно для фильтрации помех):

- **0** – в статистике учитываются все срезы;
- **1** – не учитывается наименьшее значение;
- **2** – не учитывается наибольшее значение;
- **3** – не учитываются два наименьших и одно наибольшее значение;
- **4** – не учитываются два наименьших и два наибольших значения;
- и т.д.

19.38. SH_T

Тип модуля: ФБ	Переменная	Тип	Описание
Входы	in	REAL	Контролируемое значение.
	E	BOOL	Режим сохранения значений.
Выходы	out	REAL	Сохраненное значение.

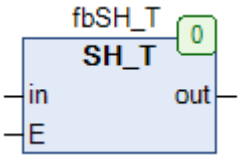


Рис. 19.68. Внешний вид **SH_T** на языке CFC

Функциональный блок **SH_T** представляет собой прозрачный элемент хранения. Пока вход **E** имеет значение **TRUE**, на выход **out** транслируется значение входа **in**. По заднему фронту на входе **E** выход **out** сохраняет свое последнее значение.

19.39. STAIR

Тип модуля: функция	Переменная	Тип	Описание
Входы	X	REAL	Входное значение.
	D	REAL	Величина ступеньки.
Выходы	STAIR	REAL	Ступенчатая функция.

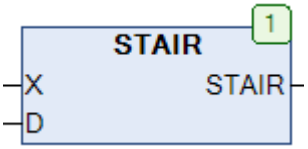


Рис. 19.69. Внешний вид функции **STAIR** на языке CFC

Функция *STAIR* возвращает значение ступенчатой функции для входного значения **X**. Вход **D** определяет величину ступеньки. Если **D=0**, то **STAIR=X**. Функция не производит фильтрацию входного значения; в случае ее необходимости следует использовать ФБ STAIR2, который учитывает гистерезис.

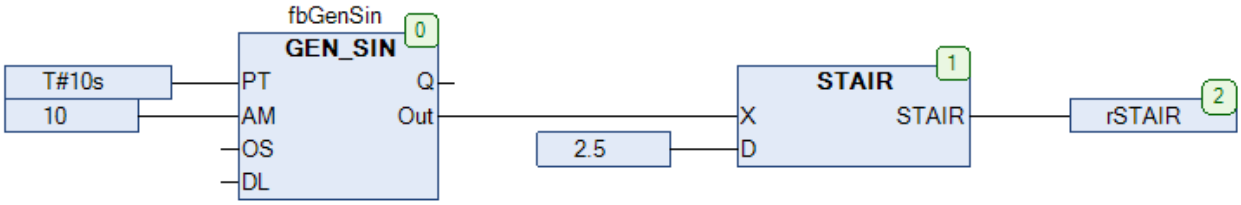


Рис. 19.70. Пример работы с функцией **STAIR** на языке CFC

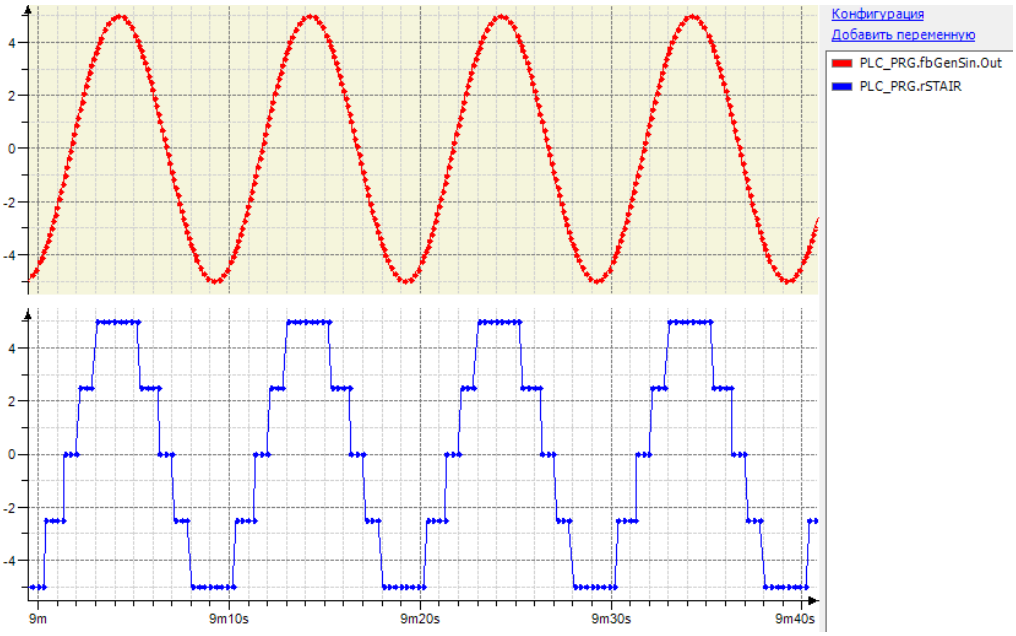


Рис. 19.71. Трассировка работы функции **STAIR** (см. рис. 19.70)

19.40. STAIR2

Тип модуля: ФБ	Переменная	Тип	Описание
Входы	X	REAL	Входное значение.
	D	REAL	Величина ступеньки/значение гистерезиса.
Выходы	Y	REAL	Ступенчатая функция.
Используемые модули	FLOOR		

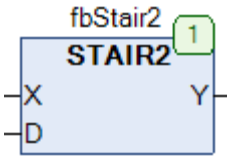


Рис. 19.72. Внешний вид ФБ **STAIR2** на языке CFC

Функциональный блок **STAIR2** возвращает на выход **Y** значение ступенчатой функции для входного значения **X**. Вход **D** определяет величину ступеньки. Если **D=0**, то **Y=X**. **D** также определяет величину гистерезиса – если изменение входного значения не превышает **D**, то значение выхода не изменится. Таким образом осуществляется подавление шума исходного сигнала.

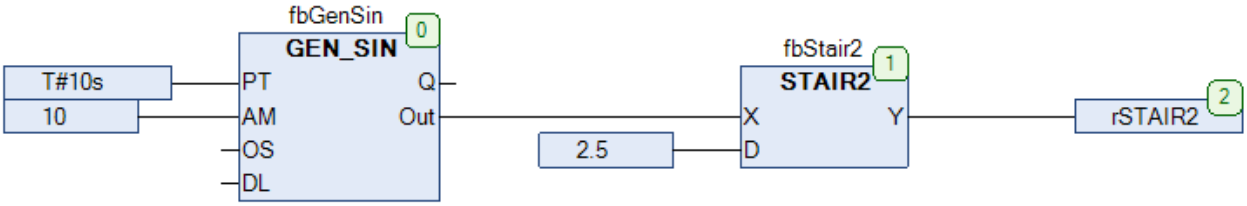


Рис. 19.73. Пример работы с ФБ **STAIR2** на языке CFC

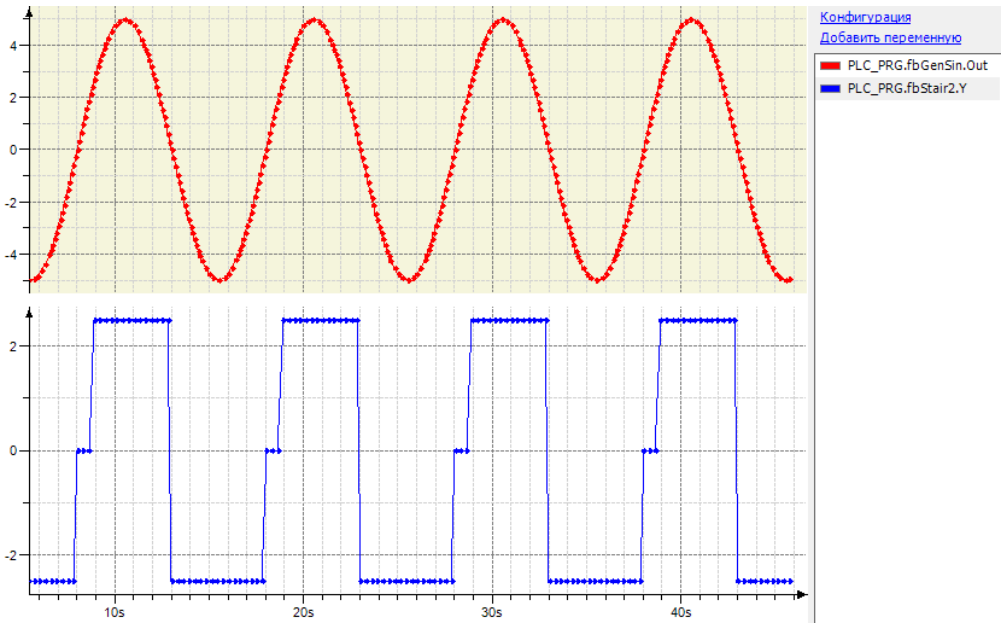


Рис. 19.74. Трассировка работы ФБ **STAIR2** (см. рис. 19.73; сравните с рис. 19.70)

19.41. TREND

Тип модуля: ФБ	**Переменная**	**Тип**	**Описание**
Входы	X	REAL	Входное значение.
Выходы	Q	BOOL	Флаг “изменение значения”.
	TU	BOOL	Флаг “увеличение значения”.
	TD	BOOL	Флаг “уменьшение значения”.
	D	REAL	Величина изменения.

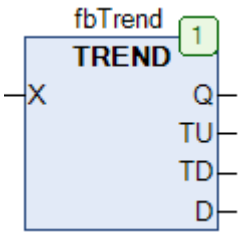


Рис. 19.75. Внешний вид ФБ **TREND** на языке CFC

Функциональный блок **TREND** используется для контроля изменения значения входной переменной **X**. Если в текущем цикле значение **X** изменилось по сравнению с предыдущим, то выход **Q** принимает значение **TRUE**. Если значение увеличилось – то выход **TU** принимает значение **TRUE**, если уменьшилось – выход **TD** принимает значение **TRUE**. На выходе **D** отображается величина, на которую изменилось значение (со знаком) в текущем цикле. Если значение не изменилось по сравнению с предыдущим циклом, то выходы **Q**, **TU** и **TD** имеют значение **FALSE**, а **D=0**.

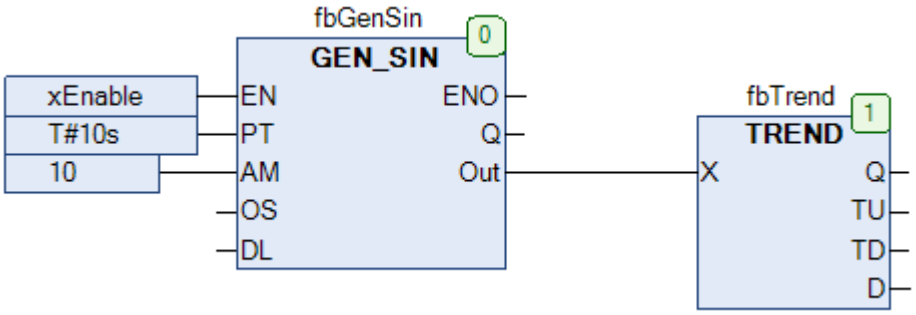


Рис. 19.76. Пример работы с ФБ **TREND** на языке CFC

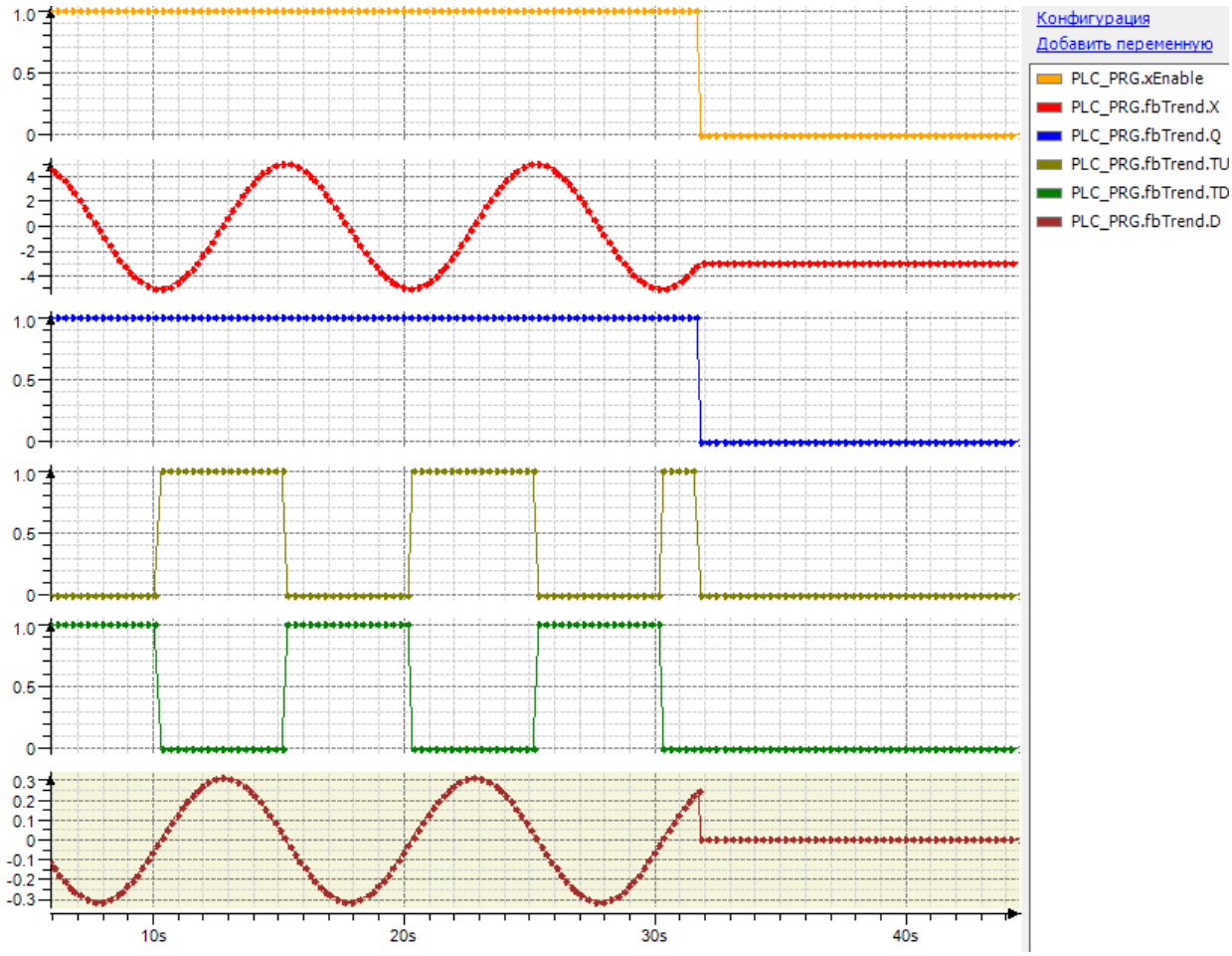


Рис. 19.77. Трассировка работы ФБ **TREND** (см. рис. 19.76)

19.42. TREND_DW

Тип модуля: ФБ	**Переменная**	**Тип**	**Описание**
Входы	X	DWORD	Входное значение.
Выходы	Q	BOOL	Флаг “изменение значения”.
	TU	BOOL	Флаг “увеличение значения”.
	TD	BOOL	Флаг “уменьшение значения”.
	D	DWORD	Величина изменения.

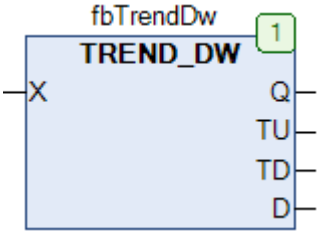


Рис. 19.78. Внешний вид ФБ **TREND_DW** на языке CFC

Блок **TREND_DW** используется для контроля изменения значения входной переменной **X**. Принцип работы полностью соответствует ФБ TREND, единственным отличием является тип используемых переменных – **DWORD**.

19.43. WORD_TO_RANGE

Тип модуля: функция	Переменная	Тип	Описание
Входы	X	WORD	Исходное значение.
	low	REAL	Нижний предел отмасштабированного значения.
	high	REAL	Верхний предел отмасштабированного значения.
Выходы	WORD_TO_RANGE	REAL	Отмасштабированное значение.

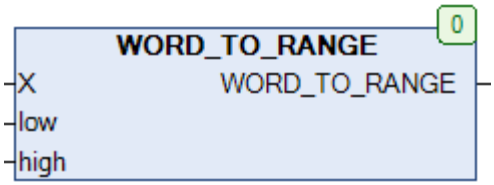


Рис. 19.79. Внешний вид функции **WORD_TO_RANGE** на языке CFC

Функция **WORD_TO_RANGE** линейно масштабирует входное целочисленное значение x из диапазона (**0…65535**) в значение с плавающей точкой из диапазона (**low…high**). См. также обратную функцию RANGE_TO_WORD.

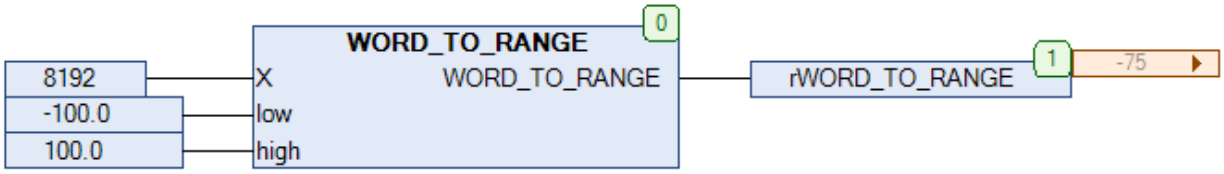


Рис. 19.80. Пример работы с функцией **WORD_TO_RANGE** на языке CFC

20. Термометры сопротивления

20.1. MULTI_IN

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	in_1	REAL	Значение 1.
	in_2	REAL	Значение 2.
	in_3	REAL	Значение 3.
	Default	REAL	Значение по умолчанию.
	in_min	REAL	Нижняя граница допустимых значений.
	in_max	REAL	Верхняя граница допустимых значений.
	mode	BYTE	Режим обработки значений.
Выходы	MULTI_IN	REAL	Выходное значение.

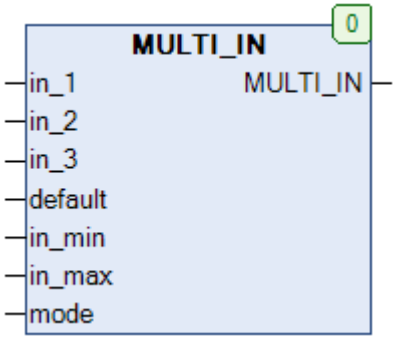


Рис. 20.1. Внешний вид функции **MULTI_IN** на языке CFC

Функция **MULTI_IN** используется для обработки сигналов от датчиков **in_1…in_3** в режиме, определяемом входом **mode**. Входы **in_min** и **in_max** определяет нижний и верхний допустимый предел входных сигналов. Если значение не входит в заданный диапазон или совпадает с его границами, то в режимах **1-4** вместо него используется величина входа **default**, а в режимах **0** и **5-7** значение не обрабатывается. Описание режимов работы функции приведено в таблице:

Значение входа mode	**Значение выхода MULTI_IN**
0	Среднее арифметическое для in_1…in_3
1	in_1
2	in_2
3	in_3
4	default
5	Наименьшее из in_1…in_3
6	Наибольшее из in_1…in_3
7	Среднее из in_1…in_3
>7	0

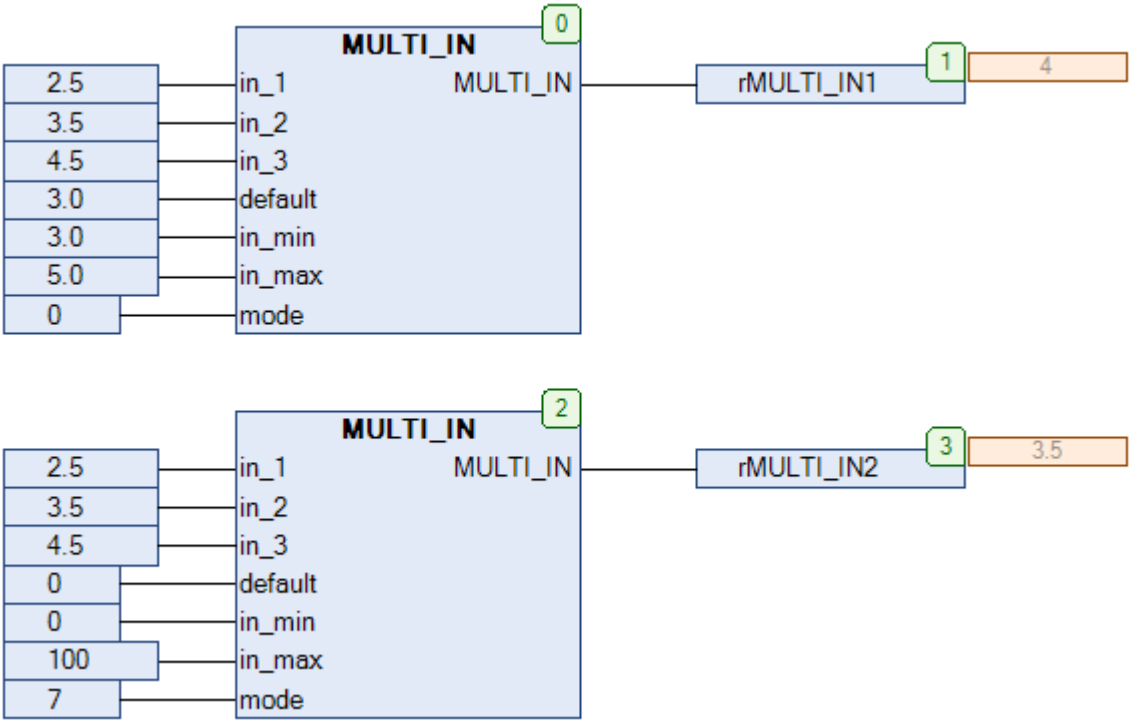


Рис. 20.2. Пример работы с функцией **MULTI_IN** на языке CFC

Обратите внимание, в первом случае значение **in_1** выходит за диапазон (**in_min…in_max**), поэтому функция возвращает среднее арифметическое для значений **in_2** и **in_3**.

20.2. RES_NI

Тип модуля: функция	Переменная	Тип	Описание
Входы	T	REAL	Измеренная температура, °C.
	R0	REAL	Номинальное сопротивление ТС, Ом.
Выходы	RES_NI	REAL	Сопротивление ТС, Ом.

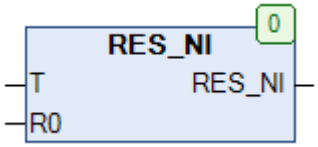


Рис. 20.3. Внешний вид функции **RES_NI** на языке CFC

Функция **RES_NI** возвращает значение сопротивления **никелевого** термометра сопротивления (ТС) с номинальным сопротивлением **R0** для измеренной температуры **T**, вычисленное по формуле:

$$\mathrm{RES\_NI} = \mathrm{R0} + \mathrm{A} \cdot \mathrm{T} + \mathrm{B} \cdot T^2 + \mathrm{B} \cdot T^4, \text{где}$$

A = 0.5485 B = $0.665 \cdot 10^{-3}$ C = $2.805 \cdot 10^{-9}$

Приемлемая точность значений функции ограничена диапазоном температур -60…+180 °C.

См. также обратную функцию TEMP_NI.

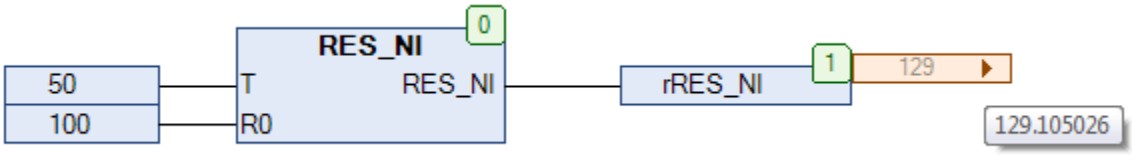


Рис. 20.4. Пример работы с функцией **RES_NI** на языке CFC

20.3. RES_NTC

Тип модуля: функция	Переменная	Тип	Описание
Входы	T	REAL	Измеренная температура, °C.
	RN	REAL	Номинальное сопротивление ТС, Ом при T=25 °C.
	B	REAL	Коэффициент температурной чувствительности, К.
Выходы	RES_NTC	REAL	Сопротивление NTC-термистора, Ом.

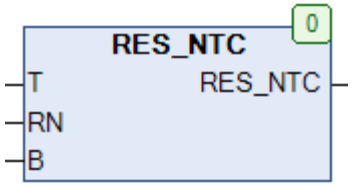


Рис. 20.5. Внешний вид функции **RES_NTC** на языке CFC

Функция **RES_NTC** возвращает значение сопротивления NTC-термистора с номинальным сопротивлением **RN** (для T=25 °C) и коэффициентом температурной чувствительности **B** для измеренной температуры **T**, вычисленное по формуле:

$$\mathrm{RES\_NTC} = \mathrm{RN} \cdot e^{\mathrm{B}\cdot\left(\frac{1}{\mathrm{T}} - \frac{1}{25+273.15}\right)}$$

Приемлемая точность значений функции ограничена диапазоном температур 0…100 °C.

См. также обратную функцию TEMP_NTC.

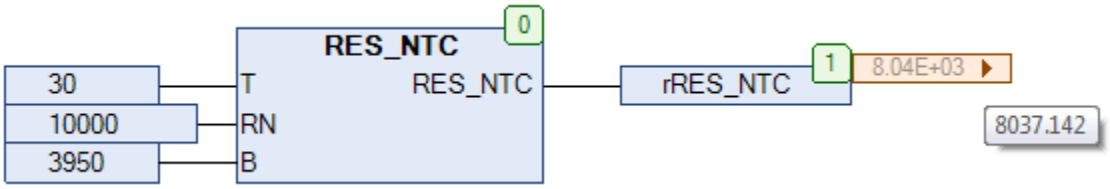


Рис. 20.6. Пример работы с функцией **RES_NTC** на языке CFC

20.4. RES_PT

Тип модуля: функция	Переменная	Тип	Описание
Входы	T	REAL	Измеренная температура, °C.
	R0	REAL	Номинальное сопротивление ТС, Ом.
Выходы	RES_PT	REAL	Сопротивление ТС, Ом.

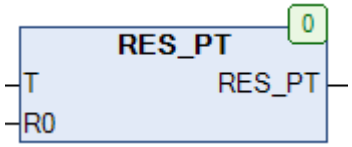


Рис. 20.7. Внешний вид функции **RES_PT** на языке CFC

Функция **RES_PT** возвращает значение сопротивления платинового термометра сопротивления (ТС) с номинальным сопротивлением **R0** для измеренной температуры **T**, вычисленное по формуле:

$$\begin{cases} \mathrm{RES\_PT} = \mathrm{R0} \cdot (1 + \mathrm{A} \cdot \mathrm{T} + \mathrm{B} \cdot T^2) \ \text{ при } \ T < 0 \\ \mathrm{RES\_PT} = \mathrm{R0} \cdot (1 + \mathrm{A} \cdot \mathrm{T} + \mathrm{B} \cdot T^2 + C \cdot (\mathrm{T} - 100) \cdot T^3) \ \text{ при } \ T > 0 \end{cases}$$

A = $3.90802 \cdot 10^{-3}$ B = $-5.802 \cdot 10^{-7}$ C = $-4.27350 \cdot 10^{-12}$

Приемлемая точность значений функции ограничена диапазоном температур -200…+850 °C.

См. также обратную функцию TEMP_PT.

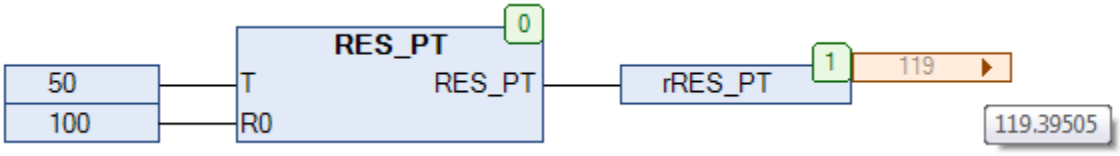


Рис. 20.8. Пример работы с функцией **RES_PT** на языке CFC

20.5. RES_SI

Тип модуля: функция	Переменная	Тип	Описание
Входы	T	REAL	Измеренная температура, °C.
	RS	REAL	Номинальное сопротивление ТС, Ом.
	TS	REAL	Температура при номинальном сопротивлении, °C.
Выходы	RES_SI	REAL	Сопротивление ТС, Ом.

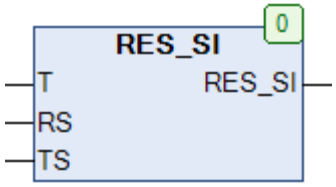


Рис. 20.9. Внешний вид функции **RES_SI** на языке CFC

Функция **RES_NI** возвращает значение сопротивления **кремниевого** термометра сопротивления (ТС) с номинальным сопротивлением **RS** при температуре **TS** для измеренной температуры **T**, вычисленное по формуле:

$$RES\_SI = RS + A \cdot (T - TS) + B \cdot (T - TS)^2, \text{где}$$

$A = 7.64 \cdot 10^{-3}$ $B = 1.66 \cdot 10^{-5}$

Приемлемая точность значений функции ограничена диапазоном температур -50…+150 °C.

См. также обратную функцию TEMP_SI.

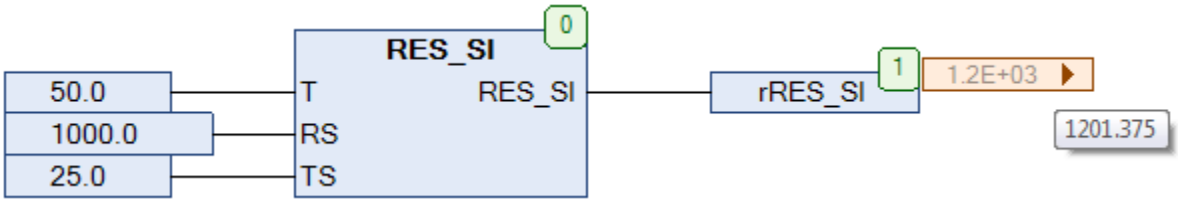


Рис. 20.10. Пример работы с функцией **RES_SI** на языке CFC (для ТС KTY83)

20.6. SENSOR_INT

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	Voltage	REAL	Напряжение, В.
	Current	REAL	Сила тока, А.
	RP	REAL	Параллельное паразитное сопротивление, Ом.
	RS	REAL	Последовательное паразитное сопротивление, Ом.
Выходы	SENSOR_INT	REAL	Сопротивление ТС, Ом.

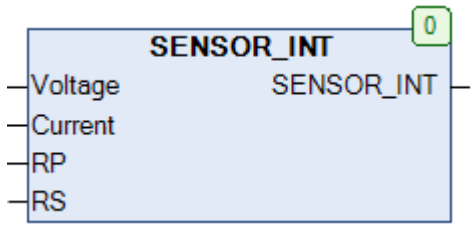


Рис. 20.11. Внешний вид функции **SENSOR_INT** на языке CFC

Функция **SENSOR_INT** вычисляет значение сопротивления для термометра сопротивления (ТС) на основании измеренных значений напряжения **Voltage** и силы тока **Current** с учетом паразитных сопротивлений **RP** и **RS**. Схема измерения приведена на рис. 20.12. Вычисленное значение сопротивления может быть преобразовано в значение температуры с помощью функций TEMP_xx.

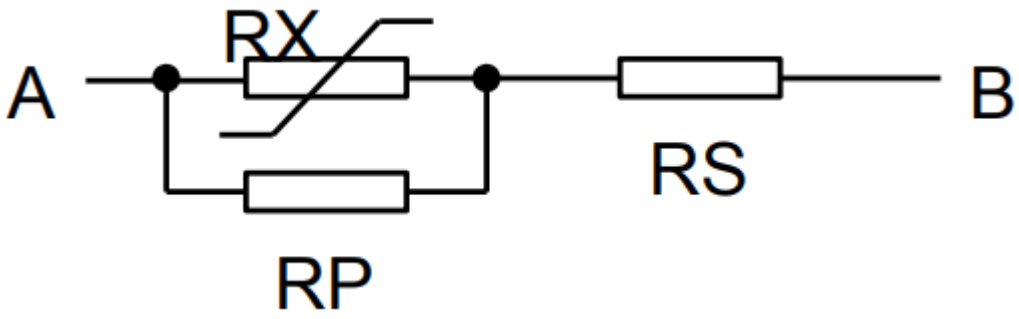


Рис. 20.12. Схема измерения для функции **SENSOR_INT**

20.7. TEMP_NI

Тип модуля: функция	Переменная	Тип	Описание
Входы	T	REAL	Измеренное сопротивление, Ом.
	R0	REAL	Номинальное сопротивление ТС, Ом.
Выходы	TEMP_NI	REAL	Температура ТС, °С.

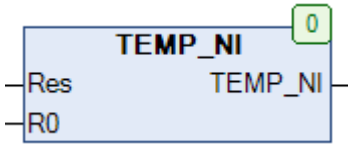


Рис. 20.13. Внешний вид функции **TEMP_NI** на языке CFC

Функция **TEMP_NI** возвращает значение температуры никелевого термометра сопротивления (ТС) с номинальным сопротивлением **R0** для измеренного сопротивления **R**. См. также обратную функцию RES_NI.

Приемлемая точность значений функции ограничена диапазоном температур -60…+180 °С.

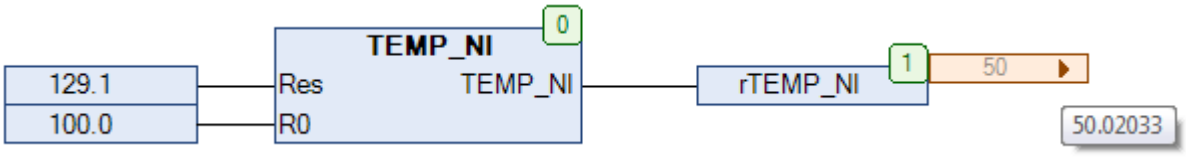


Рис. 20.14. Пример работы с функцией **TEMP_NI** на языке CFC

20.8. TEMP_NTC

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	Res	REAL	Измеренное сопротивление, Ом.
	RN	REAL	Номинальное сопротивление ТС, Ом при T=25 °C.
	B	REAL	Коэффициент температурной чувствительности, К.
Выходы	TEMP_NTC	REAL	Температура NTC-термистора, °C.

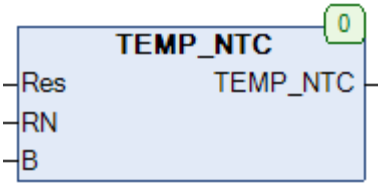


Рис. 20.15. Внешний вид функции **TEMP_NTC** на языке CFC

Функция **TEMP_NTC** возвращает значение температуры NTC-термистора с номинальным сопротивлением **RN** (для T=25 °C) и коэффициентом температурной чувствительности **B** для измеренного сопротивления **Res**. См. также обратную функцию RES_NTC.

Приемлемая точность значений функции ограничена диапазоном температур 0…100 °C.

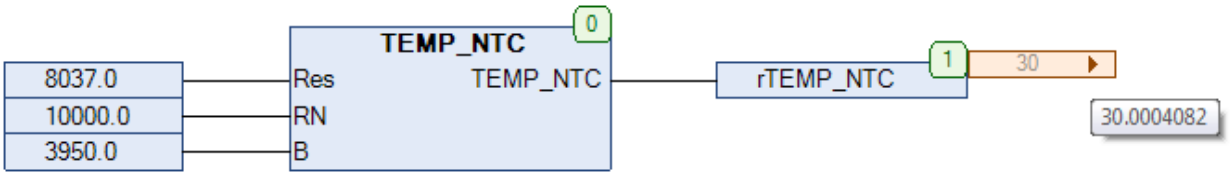


Рис. 20.16. Пример работы с функцией **TEMP_NTC** на языке CFC

20.9. TEMP_PT

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	R	REAL	Измеренное сопротивление, Ом.
	R0	REAL	Номинальное сопротивление ТС, Ом.
Выходы	TEMP_PT	REAL	Температура ТС, °С.
Используемые модули	RES_PT		

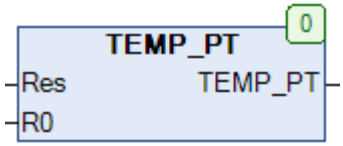


Рис. 20.17. Внешний вид функции **TEMP_PT** на языке CFC

Функция **TEMP_PT** возвращает значение температуры платинового термометра сопротивления (ТС) с номинальным сопротивлением **R0** для измеренного сопротивления **R**. См. также обратную функцию RES_PT.

Приемлемая точность значений функции ограничена диапазоном температур -200…+850 °С.

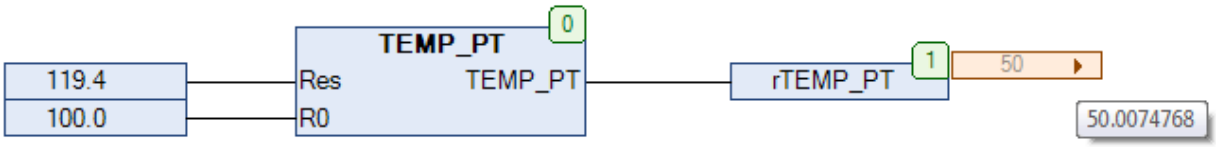


Рис. 20.18. Пример работы с функцией **TEMP_PT** на языке CFC

20.10. TEMP_SI

Тип модуля: функция	Переменная	Тип	Описание
Входы	T	REAL	Измеренное сопротивление, Ом.
	RS	REAL	Номинальное сопротивление ТС, Ом.
	TS	REAL	Температура при номинальном сопротивлении, °C.
Выходы	TEMP_SI	REAL	Температура ТС, °C.

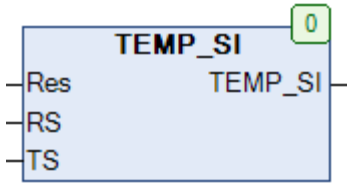


Рис. 20.19. Внешний вид функции **TEMP_SI** на языке CFC

Функция **TEMP_SI** возвращает значение температуры кремниевого термометра сопротивления (ТС) с номинальным сопротивлением **RS** при температуре **TS** для измеренного сопротивления **R**. См. также обратную функцию RES_SI.

Приемлемая точность значений функции ограничена диапазоном температур -50…+150 °C.

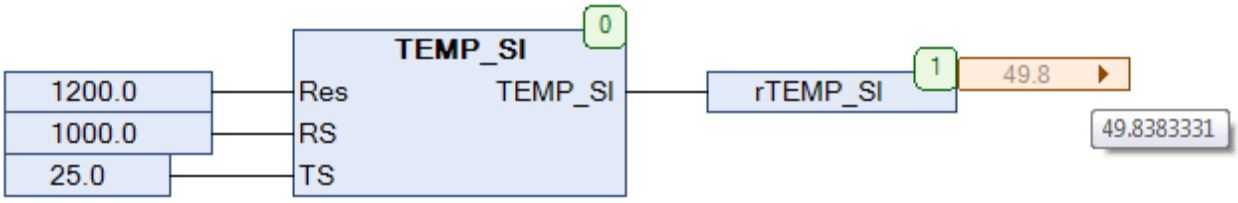


Рис. 20.20. Пример работы с функцией **TEMP_SI** на языке CFC (для ТС KTY83)

21. Модули измерения и отсчета времени

21.1. ALARM_2

Тип модуля: ФБ	**Переменная**	**Тип**	**Описание**
Входы	X	REAL	Контролируемое значение.
	LO_1	REAL	Нижний предел 1.
	HI_1	REAL	Верхний предел 1.
	LO_2	REAL	Нижний предел 2.
	HI_2	REAL	Верхний предел 2.
	HYS	REAL	Значение гистерезиса.
Выходы	Q1_LO	BOOL	Флаг «значение меньше нижнего предела 1».
	Q1_HI	BOOL	Флаг «значение больше верхнего предела 1».
	Q2_LO	BOOL	Флаг «значение меньше нижнего предела 2».
	Q2_HI	BOOL	Флаг «значение больше верхнего предела 2».

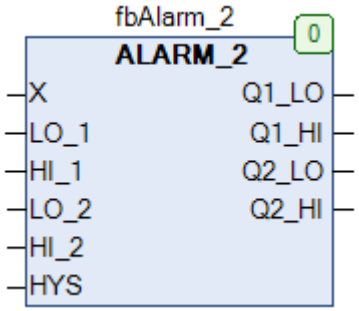


Рис. 21.1. Внешний вид ФБ **ALARM_2** на языке CFC

Функциональный блок **ALARM_2** проверяет входное значение **X** на принадлежность диапазонам [**LO_1…HI_1**] и [**LO_2…HI_2**]. Вход **HYS** определяет величину гистерезиса. Выходы блока становятся активными в следующих случаях:

$$\begin{cases} \text{Q1\_LO} = \text{TRUE если X} < \text{LO\_1} \pm \dfrac{\text{HYS}}{2} \\ \text{Q1\_HI} = \text{TRUE если X} > \text{HI\_1} \pm \dfrac{\text{HYS}}{2} \\ \text{Q2\_LO} = \text{TRUE если X} < \text{LO\_2} \pm \dfrac{\text{HYS}}{2} \\ \text{Q2\_HI} = \text{TRUE если X} > \text{HI\_2} \pm \dfrac{\text{HYS}}{2} \end{cases}$$

Обычно с помощью диапазона [**LO_1…HI_1**] задаются пороговые значения для сигнала тревоги, с помощью [**LO_2…HI_2**] – для сигнала аварии.

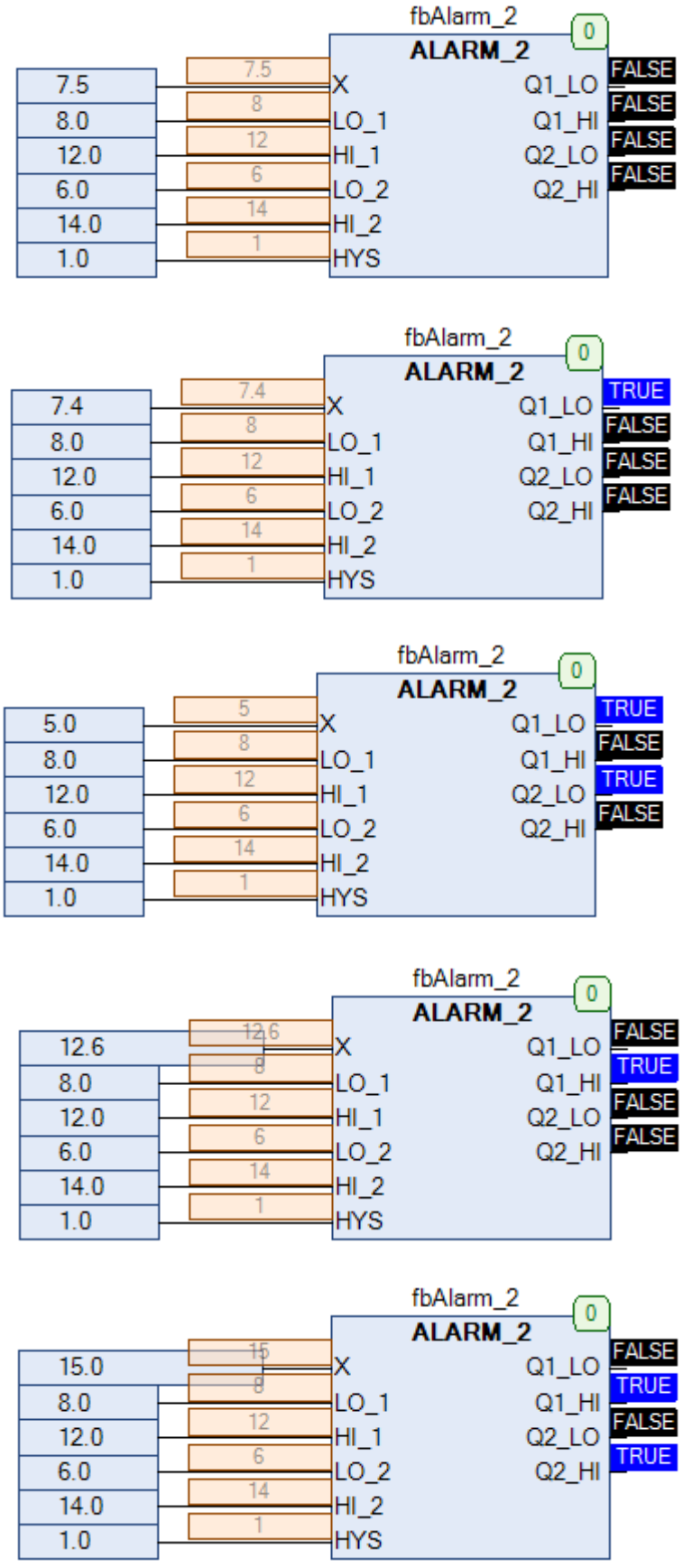


Рис. 21.2. Пример работы с ФБ **ALARM_2** на языке CFC

21.2. BAR_GRAPH

Тип модуля: ФБ	Переменная	Тип	Описание
Входы	X	REAL	Контролируемое значение.
	rst	REAL	Сигнал сброса тревоги.
Выходы	LOW	BOOL	Флаг «значение меньше нижнего предела».
	Q1…Q6	BOOL	Флаги «значение в зоне 1…6».
	HIGH	BOOL	Флаг «значение больше верхнего предела».
	Alarm	BOOL	Флаг «тревога».
	Status	BYTE	ESR-код.
Параметры	trigger_Low	REAL	Нижний предел.
	trigger_High	REAL	Верхний предел.
	Alarm_low	BOOL	Режим «обработка нижнего предела тревоги».
	Alarm_high	BOOL	Режим «обработка верхнего предела тревоги».
	log_scale	BOOL	Режим «логарифмирование интервалов».

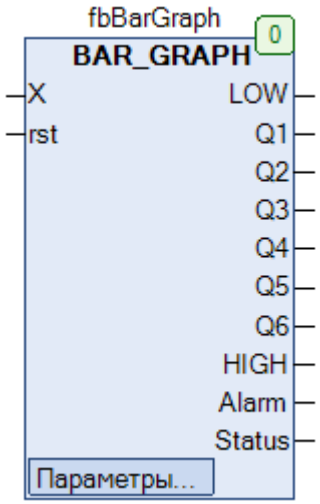


Рис. 21.3. Внешний вид ФБ **BAR_GRAPH** на языке CFC

Функциональный блок **BAR_GRAPH** проверяет входное значение **X** на принадлежность диапазону [**trigger_Low…trigger_High**]. Выход **LOW** принимает значение **TRUE**, если значение **X** меньше нижней границы диапазона, выход **HIGH** – если больше верхней границы. Если параметры **Alarm_low** и **Alarm_high** имеют значение **TRUE**, то при выходе значения **X** за границы допустимого диапазона выход **Alarm** принимает значение **TRUE**. Выход **Alarm** остается активным до тех пор, пока не будет сброшен с помощью сигнала по переднему фронту на входе **rst**.

Если значение **X** находится в пределах диапазона, то активен один из выходов **Q1…Q6**, характеризующий интервал, в котором находится значение. По умолчанию интервалы являются равными:

$$\begin{cases} Q1 = \text{TRUE если } trigger\_Low < X < trigger\_Low + 1 \cdot \dfrac{trigger\_High - trigger\_Low}{6} \\ Q2 = \text{TRUE если } trigger\_Low < X < trigger\_Low + 2 \cdot \dfrac{trigger\_High - trigger\_Low}{6} \\ \ldots \\ Q6 = \text{TRUE если } trigger\_Low < X < trigger\_Low + 6 \cdot \dfrac{trigger\_High - trigger\_Low}{6} \end{cases}$$

Если параметр **log_scale** имеет значение **TRUE**, то размеры интервалов связаны через логарифм:

$$\begin{cases} Q1 = \text{TRUE если } trigger\_Low < X < trigger\_Low \cdot \dfrac{1}{6} \cdot [e^{\ln\left(\frac{trigger\_High}{trigger\_Low}\right)}]^1 \\ Q2 = \text{TRUE если } trigger\_Low < X < trigger\_Low \cdot \dfrac{1}{6} \cdot [e^{\ln\left(\frac{trigger\_High}{trigger\_Low}\right)}]^2 \\ \ldots \\ Q6 = \text{TRUE если } trigger\_Low < X < trigger\_Low \cdot \dfrac{1}{6} \cdot [e^{\ln\left(\frac{trigger\_High}{trigger\_Low}\right)}]^6 \end{cases}$$

Выход **Status** определяет состояние блока и совместим с ESR-модулями:

Значение выхода Status	Описание
110	Значение X находится в диапазоне [trigger_Low…trigger_High].
111	Значение X меньше trigger_Low, выход LOW равен TRUE.
112	Значение X больше trigger_High выход HIGH равен TRUE.
1	Значение X меньше trigger_Low, выход ALARM равен TRUE.
2	Значение X больше trigger_High выход ALARM равен TRUE.

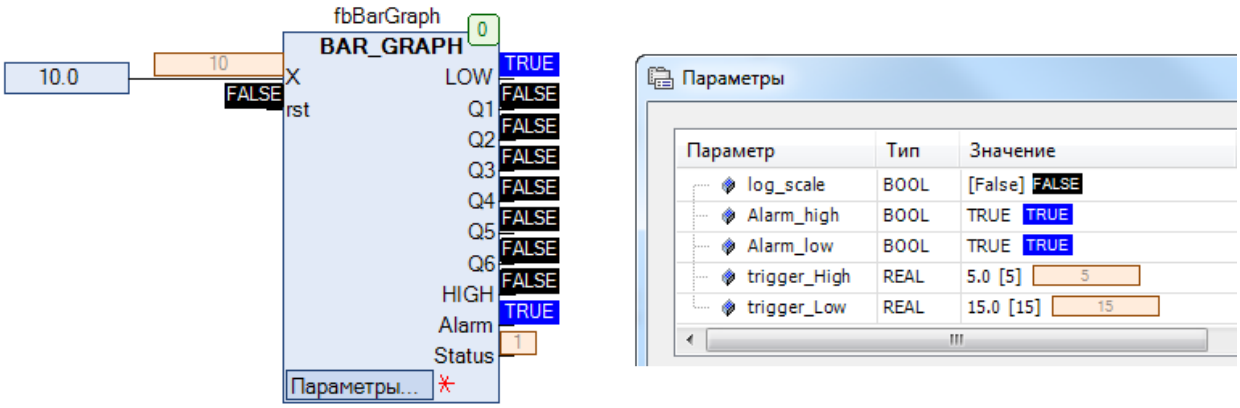


Рис. 21.4. Пример работы с ФБ **BAR_GRAPH** на языке CFC

21.3. CALIBRATE

Тип модуля: ФБ	Переменная	Тип	Описание
Входы	X	REAL	Сигнал датчика.
	CO	BOOL	Сигнал датчика для нижнего предела.
	CS	BOOL	Сигнал датчика для верхнего предела.
Выходы	Y	REAL	Измеренное значение.
Параметры	Y_Scale	REAL	Верхний предел измеренного значения.
	Y_Offset	REAL	Нижний предел измеренного значения.

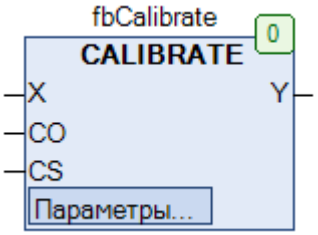


Рис. 21.5. Внешний вид ФБ **CALIBRATE** на языке CFC

Функциональный блок **CALIBRATE** используется для калибровки сигнала аналогового датчика **X**. Параметры **Y_Offset** и **Y_Scale** определяют нижний и верхний предел измеряемого значения. По переднему фронту на входе **CO** в блок записывается измеренное значение для нижнего предела, по переднему фронту на входе **CS** –для верхнего. После этого значение входа **X** будет линейно масштабироваться и подаваться на выход **Y** с учетом рассчитанных коэффициентов. Коэффициенты являются энергонезависимыми (**VAR RETAIN**).

Поясним вышесказанное на примере. Предположим, мы получаем от датчика температуры сигнал **4-20 мА**, который соответствует диапазону **0-100 °C**. Зададим **Y_Offset=0** и **Y_Scale=100**. Поместим датчик в среду с температурой **0 °C** (значение температуры должно быть измерено с помощью эталонного термометра) и подадим импульс по переднему фронту на входе **CO**. Блок вычислит значение смещения сигнала. Нагреем датчик до **100 °C** и подадим импульс по переднему фронту на входе **CS**. Блок вычислит коэффициент масштабирования для входного сигнала. В случае, если датчик изначально откалиброван, смещение должно составить

$$\mathrm{Offset} = \mathrm{Y\_OFFSET} - \mathrm{x} = 0 - 4 = -4$$

а коэффициент масштабирования –

$$\mathrm{Scale} = \frac{\mathrm{Y\_SCALE}}{\mathrm{X} + \mathrm{Offset}} = \frac{100}{20 + (-4)} = 6.5$$

Теперь если подать на вход **X** измеренное датчиком значение (например, **12 мА**), то на выходе **Y** будет получено значение температуры **– 50 °C**. Поскольку в реальной жизни по мере эксплуатации характеристики датчиков могут изменяться, использование данного блока позволит поддерживать точность измеренного значения на приемлемом уровне.

21.4. CYCLE_TIME

Тип модуля: ФБ	**Переменная**	**Тип**	**Описание**
Входы	RST	BOOL	Сигнал сброса блока.
Выходы	ct_min	TIME	Минимальное время цикла.
	ct_max	TIME	Максимальное время цикла.
	ct_last	TIME	Последнее измеренное время цикла.
	systime	TIME	Время, прошедшее со старта ПЛК.
	sysdays	INT	Число дней, прошедшее со старта ПЛК.
	cycles	DWORD	Число циклов ПЛК с момента старта.
Используемые модули	T_PLC_MS		

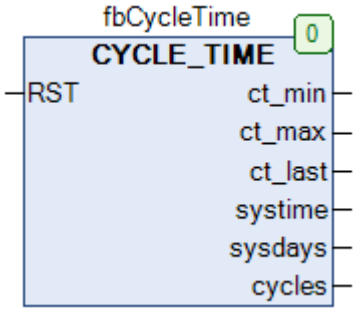


Рис. 21.6. Внешний вид ФБ **CYCLE_TIME** на языке CFC

Функциональный блок **CYCLE_TIME** используется для получения статистической информации о времени цикла ПЛК. На выходы **ct_min**, **ct_max** и **ct_last** подается минимальное, максимальное и последнее измеренное время цикла. Выходы **systime** и **sysdays** содержат информацию о времени и числе дней, прошедших со старта ПЛК. Выход **cycles** содержит число циклов ПЛК, прошедших с момента его старта. По переднему фронту на входе **RST** выходы блока обнуляются и отсчет начинается заново.

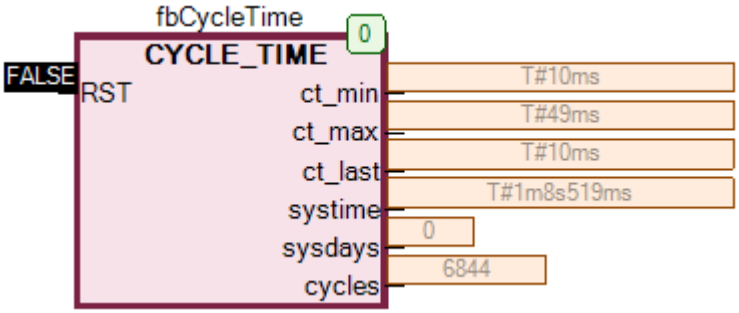


Рис. 21.7. Пример работы с ФБ **CYCLE_TIME** на языке CFC

21.5. DT_SIMU

Тип модуля: ФБ	**Переменная**	**Тип**	**Описание**
Входы	START	DT	Начальное время часов (точка отсчета).
	SPEED	REAL	Коэффициент масштаба времени.
Выходы	DTS	DT	Текущее время.
Используемые модули	T_PLC_MS		

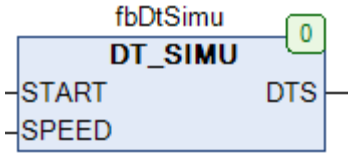


Рис. 21.6. Внешний вид ФБ **DT_SIMU** на языке CFC

Функциональный блок **DT_SIMU** представляет собой программный модуль **RTC** с задаваемым масштабом времени. Вход **START** определяет начальное время, которое будет являться точкой отсчета. Выход **DTS** отображает текущее время. Вход **SPEED** определяет масштаб времени относительно стандартного. Например, если **SPEED=5.0**, то каждую секунду к текущему значению времени на выходе **DTS** будет прибавляться 5 секунд – т.е. время будет идти в 5 раз быстрее, чем в реальном мире. Это может быть полезным при отладке проектов, связанных с медленно протекающими процессами.

21.6. FLOW_METER

Тип модуля: ФБ	Переменная	Тип	Описание
Входы	VX	REAL	Измеренное значение расхода (ед./час).
	E	BOOL	Сигнал управления блоком.
	RST	BOOL	Сигнал сброса блока.
Выходы	F	REAL	Мгновенное значение расхода (ед./час).
Входы-выходы	X	REAL	Суммарное значение расхода (дробная часть).
	Y	UDINT	Суммарное значение расхода (целая часть).
Параметры	PULSE_MODE	BOOL	Режим работы режим работы блока.
	UPDATE_TIME	TIME	Время опроса входа VX.
Используемые модули	INTEGRATE, FLOOR, T_PLC_MS		

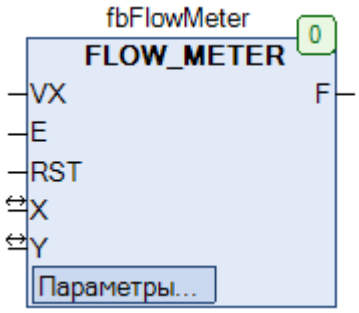


Рис. 21.7. Внешний вид ФБ **FLOW_METER** на языке CFC

Функциональный блок **FLOW_METER** используется для определения суммарного расхода. На вход **VX** подается измеренное значение расхода. Параметр **PULSE_MODE** определяет режим работы блока. Если **PULSE_MODE** имеет значение **FALSE**, то расходомер работает в режиме интегрирования и пока вход **E** имеет значение **TRUE**, значение входа **VX** интерпретируется как расход за час, а на входах-выходах **Y** и **X** отображается целая и дробная часть суммарного расхода за прошедшее с начала работы блока время. На выходе **F** отображается текущее значение расхода. Параметр **UPDATE_TIME** частоту опроса входа **VX**. Если **PULSE_MODE** имеет значение **TRUE**, то расходомер работает в импульсном режиме и при импульсе по переднему фронту входа **E** прибавляет значение **VX** к суммарному значению расхода **Y** и **X** (целая часть/дробная часть). По переднему фронту входа **RST** происходит сброс памяти блока, после чего отсчет начинается заново.

Суммарное значение расхода представлено в виде двух величин – **X** и **Y**. Если суммарный расход = 210.115 л/ч, то Y=210 (целая часть), X=0.115 (дробная часть).

Поясним вышесказанное несколькими примерами.

1. Пусть VX=3600 л/ч, PULSE_MODE=FALSE, UPDATE_TIME=T#100ms

Расходомер работает в режиме интегратора, вход **VX** опрашивается каждые 100 мс (т.е если значение входа изменится, то блок начнет использовать его вместо предыдущего с

задержкой до 100 мс). Значение **Y** каждую секунду увеличивается на +1 (3600 л/ч = 1 л/с). Значение **X** содержит дробную часть значения и изменяется с частотой цикла ПЛК.

2. Пусть VX=10 л/ч, PULSE_MODE=TRUE, UPDATE_TIME=T#100ms

Расходомер работает в импульсном режиме, вход **VX** опрашивается каждые 100 мс (т.е если значение входа изменится, то блок начнет использовать его вместо предыдущего с задержкой до 100 мс). Значение **Y** увеличивается на +10 по переднему фронту на входе **E**.

21.7. M_D

Тип модуля: ФБ	**Переменная**	**Тип**	**Описание**
Входы	start	BOOL	Сигнала начала отсчета.
	stop	BOOL	Сигнал окончания отсчета.
	tmax	TIME	Максимальное время отсчета.
	RST	BOOL	Сигнал сброса блока.
Выходы	PT	TIME	Время между сигналом старта и стопа.
	ET	TIME	Прошедшее время.
	run	BOOL	Флаг «идет отсчет времени».
Используемые модули	T_PLC_MS		

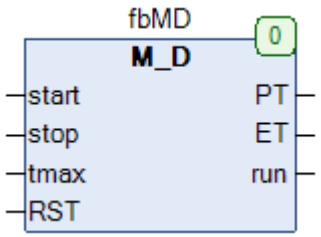


Рис. 21.8. Внешний вид ФБ **M_D** на языке CFC

Функциональный **блок M_D** представляет собой модуль измерения времени между двумя импульсами переднего фронта. По переднему фронту на входе **start** блок начинает отсчет времени, при этом на выходе **ET** отображается время, прошедшее с начала отсчета, а выход **RUN** имеет значение **TRUE**. По переднему фронту на входе **stop** отсчет времени прекращается, при этом на выходе **PT** отображается время окончания отсчета (совпадающее с текущим значением выхода **ET**), а выход **RUN** принимает значение **FALSE**. Вход **tmax** используется для ограничения продолжительности отсчета – если **ET > tmax**, то блок прекращает работу, а его выходы обнуляются. По переднему фронту на входе **RST** происходит принудительная остановка блока с обнулением выходов.

Обратите внимание, старт начала отсчета возможен только в том случае, если входы **stop** и **RST** имеют значение **FALSE**.

21.8. M_T

Тип модуля: ФБ	Переменная	Тип	Описание
Входы	IN	BOOL	Контролируемый сигнал.
	tmax	TIME	Максимальное время отсчета.
	RST	BOOL	Сигнал сброса блока.
Выходы	PT	TIME	Время, которое сигнал был в TRUE.
	ET	TIME	Прошедшее время.
Используемые модули	T_PLC_MS		

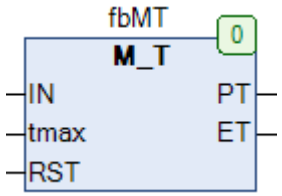


Рис. 21.9. Внешний вид ФБ **M_T** на языке CFC

Функциональный блок **M_T** представляет собой модуль измерения времени между импульсами переднего и заднего фронта. По переднему фронту на входе **IN** блок начинает отсчет времени, при этом на выходе **ET** отображается время, прошедшее с начала отсчета. По заднему фронту на входе **IN** отсчет времени прекращается, при этом на выходе **PT** отображается время окончания отсчета (совпадающее с текущим значением выхода **ET**). Таким образом, блок измеряет время, в течение которого вход **IN** находился в значении **TRUE**. Вход **tmax** используется для ограничения продолжительности отсчета – если **ET > tmax**, то блок прекращает работу, а его выходы обнуляются. По переднему фронту на входе **RST** происходит принудительная остановка блока с обнулением выходов.

Обратите внимание, старт начала отсчета возможен только в том случае, если входы **stop** и **RST** имеют значение **FALSE**.

21.9. M_TX

Тип модуля: ФБ	Переменная	Тип	Описание
Входы	IN	BOOL	Контролируемый сигнал.
	tmax	TIME	Максимальное время отсчета.
	RST	BOOL	Сигнал сброса блока.
Выходы	TH	TIME	Время, которое IN был в TRUE.
	TL	TIME	Время, которое IN был в FALSE.
	DC	REAL	Процент времени, которое IN был в TRUE (0…1).
	F	REAL	Частота изменения сигнала на входе IN.
	ET	TIME	Прошедшее время.
Используемые модули	T_PLC_MS		

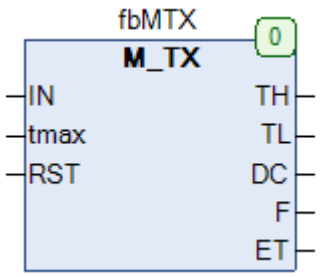


Рис. 21.10. Внешний вид ФБ M_TX на языке CFC

Функциональный блок **M_TX** представляет собой модуль сбора статистики о состоянии входа **IN.** По переднему фронту на входе **IN** блок начинает отсчет времени, при этом на выходе **ET** отображается время, прошедшее с начала отсчета. По заднему фронту на входе **IN** ***отсчет не прекращается***; для остановки работы блока необходим импульс по переднему фронту на входе **RST**, который также обнулит его выходы. На выходах **TH** и **TL** отображается время, которое сигнал **IN** находится в состоянии **TRUE** и **FALSE** соответственно. Выход **DC** сообщает, какой процент времени сигнал находился в состоянии **TRUE** (например, если **DC=0.4**, то в течение 40% времени вход **IN** имел значение **TRUE**, в течение 60% - значение **FALSE**). На выходе **F** отображается частота изменения входа **IN**. Вход **tmax** используется для ограничения продолжительности отсчета – если **ET > tmax**, то блок прекращает работу, а его выходы обнуляются.

Обратите внимание, обновление значений на выходах **TH**, **TL**, **DC** и **F** происходит после очередного импульса на входе **IN**. В первый раз значения на этих выхода появятся после **второго** импульса по переднему фронту на входе **IN** с начала работы блока.

21.10. METER

Тип модуля: ФБ	Переменная	Тип	Описание
Входы	M1	REAL	Контролируемое значение 1.
	M2	REAL	Контролируемое значение 2.
	I1	BOOL	Режим контроля значения 1.
	I2	BOOL	Режим контроля значения 2.
	D	REAL	Делитель для выхода.
	RST	BOOL	Сигнал сброса блока.
Входы-выходы	MX	REAL	Интегральное значение M1 и M2.
Используемые модули	T_PLC_MS, R2_ADD		

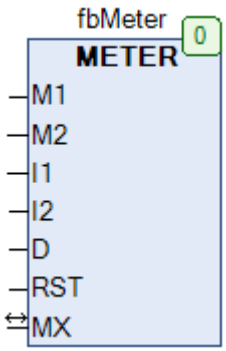


Рис. 21.11. Внешний вид ФБ **METER** на языке CFC

Функциональный блок **METER** используется для определения интегрального значения двух величин за заданный период времени. Пока вход **I1** имеет значение **TRUE**, значение **M1** каждый цикл ПЛК прибавляется к последнему сохраненному значению. Пока вход **I2** имеет значение **TRUE**, значение **M2** каждый цикл ПЛК прибавляется к последнему сохраненному значению. Вход **D** определяет значение делителя для сохраненного значения. Выход **MX** содержит сумму всех предыдущих значений **M1** и **M2** за периоды, во время которых **I1** и **I2** имели значение **TRUE**, поделенную на **D**:

$$MX = \frac{1}{D} \cdot \sum (I1 \cdot M1 + I2 \cdot M2)$$

По переднему фронту на входе **RST** значение **MX** обнуляется, после чего суммирование начинается заново.

На рис. 21.12 приведен пример использования блока для подсчета суммарной потребляемой электроэнергии для двух нагревателей с мощностью 60 и 85 кВ соответственно. Переменные **xI1** и **xI2** определяют состояние соответствующего нагревателя (**TRUE** – включен, **FALSE** – отключен).

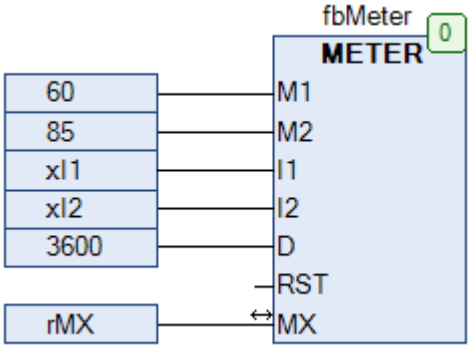


Рис. 21.12. Пример работы с ФБ **METER** на языке CFC

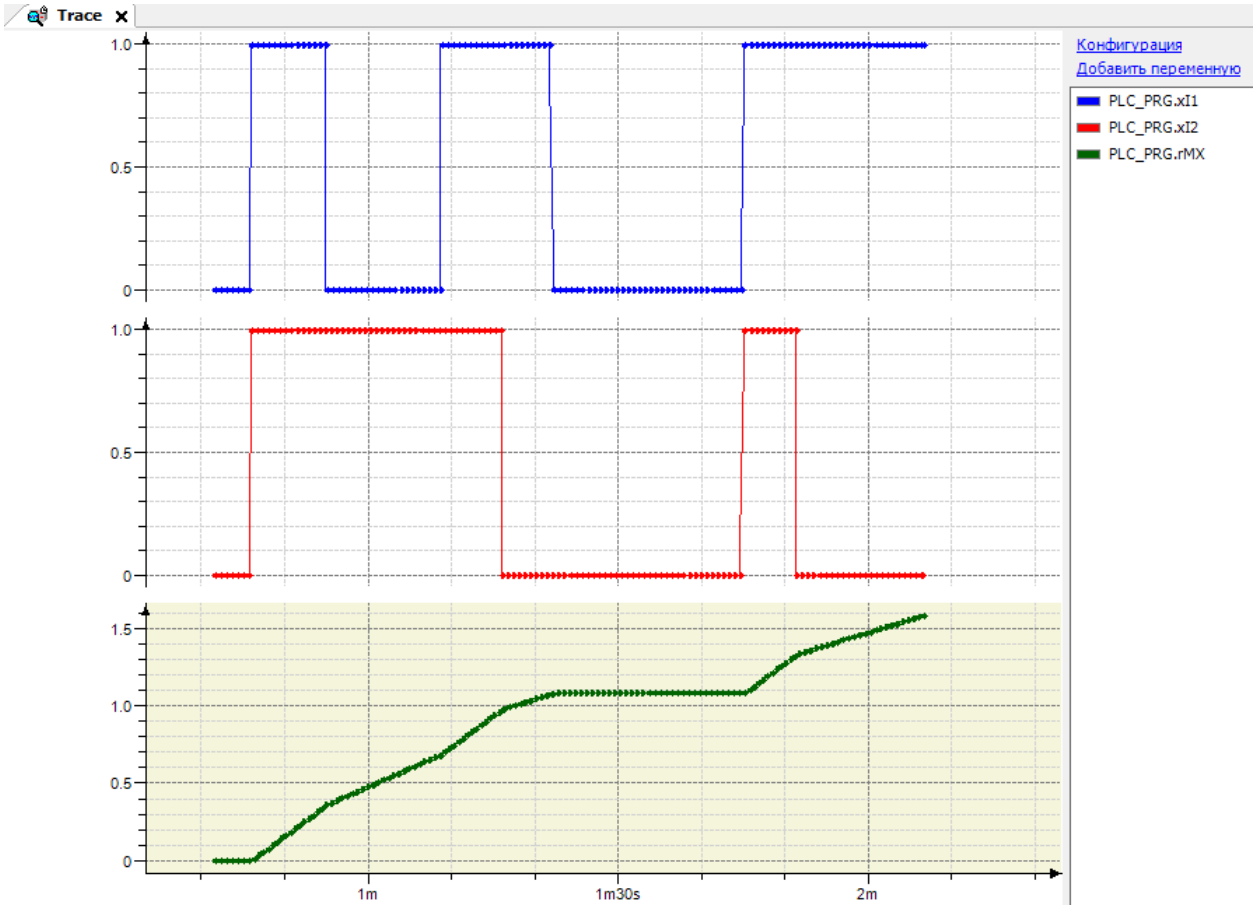


Рис. 21.13. Трассировка работы ФБ **METER** (см. рис. 21.12)

21.11. METER_STAT

Тип модуля: ФБ	Переменная	Тип	Описание
Входы	IN	REAL	Контролируемое значение.
	DI	DATE	Текущая дата.
	RST	BOOL	Сигнал сброса блока.
Входы-выходы	Last_Day	REAL	Расход за вчерашний день.
	Current_Day	REAL	Расход за текущий день.
	Last_Week	REAL	Расход за прошлую неделю.
	Current_Week	REAL	Расход за текущую неделю.
	Last_Month	REAL	Расход за прошлый месяц.
	Current_Month	REAL	Расход за текущий месяц.
	Last_Year	REAL	Расход за прошлый год.
	Current_Year	REAL	Расход за текущий год.
Используемые модули	YEAR_OF_DATE, MONTH_OF_DATE, DAY_OF_YEAR, DAY_OF_WEEK		

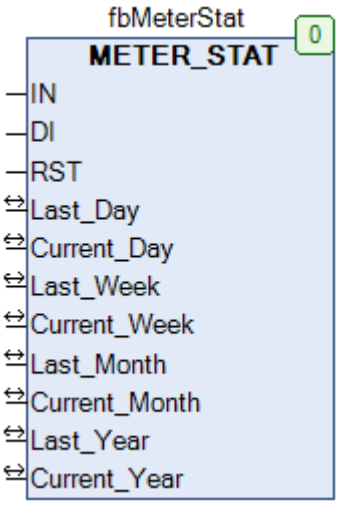


Рис. 21.14. Внешний вид ФБ **METER_STAT** на языке CFC

Функциональный блок **METER_STAT** используется для определения суммарного расхода величины **IN** за прошлые день (**Last_Day**), неделю (**Last_Week**), месяц (**Last_Month**) и год (**Last_Year**), а также за текущие день (**Current_Day**), неделю (**Current_Week**), месяц (**Current_Month**) и год (**Current_Year**). На вход **DI** подается текущая дата. По переднему фронту на входе **RST** выходы блока обнуляются, после чего сбор статистики начинается заново.

Логика работы блока подразумевает, что **IN** является возрастающей с течением времени величиной, характеризующий расход какого-либо вещества, потребление электроэнергии и т.п.

21.12. ONTIME

Тип модуля: ФБ	**Переменная**	**Тип**	**Описание**
Входы	IN	BOOL	Контролируемый сигнал.
	RST	BOOL	Сигнал сброса блока.
Входы-Выходы	SECONDS	UDINT	Время активного сигнала.
	CYCLES	UDINT	Число периодов активности сигнала.
Используемые модули	T_PLC_MS		

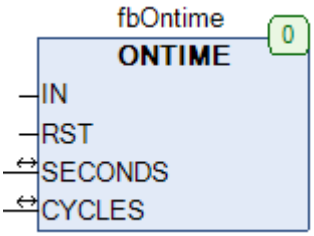


Рис. 21.15. Внешний вид ФБ **ONTIME** на языке CFC

Функциональный блок **ONTIME** представляет собой модуль измерения периода активности сигнала. Если вход **IN** имеет значение **TRUE**, то на входе-выходе **SECONDS** отображается число секунд активности сигнала. Если вход **IN** принимает значение **FALSE**, то отсчет времени останавливается, но будет продолжен при следующем переходе в состояние **TRUE**. По переднему фронту входа **IN** значение входа-выхода **CYCLES** увеличивается на +1 – таким образом, эта величина характеризует число периодов активности сигнала. По переднему фронту на входе **RST** значения **SECONDS** и **CYCLES** обнуляются, после чего подсчет начинается заново.

21.13. T_PLC_MS

Тип модуля: функция	Переменная	Тип	Описание
Выходы	T_PLC_MS	DWORD	Значение системного таймера ПЛК.
Константы	degug	BOOL	Управление режимом отладки.
	N	INT	Коэффициент масштаба времени.
	offset	DWORD	Смещение времени таймера, мс.

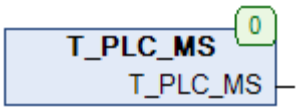


Рис. 21.16. Внешний вид функции **T_PLC_MS** на языке CFC

Функция **T_PLC_MS** возвращает значение системного таймера (время, прошедшее со старта ПЛК) в миллисекундах. В версии библиотеки **OSCAT** для среды **CODESYS 3.5** функция **T_PLC_MS** представляет собой обвязку вокруг системной функции **TIME()**, но помимо этого позволяет тестировать системный таймер на переполнение. Для этого надо внести изменения в область объявления переменных функции: присвоить константе **debug** значение **TRUE** и задать значения для констант **N** (коэффициент масштаба времени, каждую миллисекунду к значению таймера будет прибавляться 2^N миллисекунд) и **offset** (смещение значения таймера).

Следует помнить, что реализация программной обвязки для системного таймера может отличаться для разных сред программирования.

21.14. T_PLC_US

Тип модуля: функция	Переменная	Тип	Описание
Выходы	T_PLC_US	DWORD	Значение системного таймера ПЛК.
Константы	degug	BOOL	Управление режимом отладки.
	N	INT	Коэффициент масштаба времени.
	offset	DWORD	Смещение времени таймера, мкс.

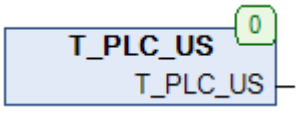


Рис. 21.17. Внешний вид функции **T_PLC_US** на языке CFC

Функция **T_PLC_US** возвращает значение системного таймера (время, прошедшее со старта ПЛК) в микросекундах. В версии библиотеки **OSCAT** для среды **CODESYS 3.5** функция фактически возвращает значение системной функции **TIME()**, умноженное на 1000. При необходимости эта реализация может быть заменена на вызов системной функции **LTIME()**, если последняя поддержана на ПЛК. Функция **T_PLC_US** также содержит режим отладки; см. подробности в описании функции T_PLC_MS.

21.15. TC_MS

Тип модуля: ФБ	**Переменная**	**Тип**	**Описание**
Входы	TC	DWORD	Время системного таймера, мс.
Используемые модули	T_PLC_MS		

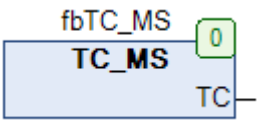


Рис. 21.18. Внешний вид ФБ **TC_MS** на языке CFC

Функциональный блок **TC_MS** возвращает значение длительности (в миллисекундах) прошедшего цикла задачи ПЛК, в которой осуществляется вызов блока. Блок представляет собой обвязку для функции T_PLC_MS; если блок возвращает некорректное значение, то следует проверить, не используется ли данная функция в режиме отладки.

21.16. TC_S

Тип модуля: ФБ	**Переменная**	**Тип**	**Описание**
Входы	TC	REAL	Время системного таймера.
Используемые модули	T_PLC_US		

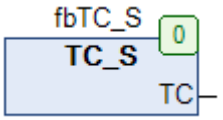


Рис. 21.19. Внешний вид ФБ **TC_S** на языке CFC

Функциональный блок **TC_S** возвращает значение длительности прошедшего цикла задачи ПЛК, в которой осуществляется вызов блока, в виде переменной с плавающей точкой с разрядностью до миллисекунд (например, 0.01 для T=10 мс). Блок представляет собой обвязку для функции T_PLC_US; если блок возвращает некорректное значение, то следует проверить, не используется ли данная функция в режиме отладки. Более подробную информацию (в т.ч. сведения о точности) см. в описании функции T_PLC_US.

21.17. TC_US

Тип модуля: ФБ	**Переменная**	**Тип**	**Описание**
Входы	TC	DWORD	Время системного таймера, мкс.
Используемые модули	T_PLC_US		

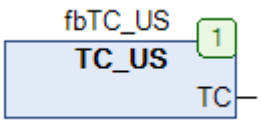


Рис. 21.20. Внешний вид ФБ **TC_US** на языке CFC

Функциональный блок TC_S возвращает значение длительности (в микросекундах) прошедшего цикла задачи ПЛК, в которой осуществляется вызов блока. Блок представляет собой обвязку для функции T_PLC_US; если блок возвращает некорректное значение, то следует проверить, не используется ли данная функция в режиме отладки. Более подробную информацию (в т.ч. сведения о точности) см. в описании функции T_PLC_US.

22. Конвертация величин

22.1. ASTRO

Тип модуля: ФБ	Переменная	Тип	Описание
Входы	m	REAL	Исходное расстояние, м.
	AE	REAL	Исходное расстояние, а.е.
	PC	REAL	Исходное расстояние, парсеки.
	LJ	REAL	Исходное расстояние, световые года.
Выходы	Ym	REAL	Конвертированное расстояние, м.
	YAE	REAL	Конвертированное расстояние, а.е.
	YPC	REAL	Конвертированное расстояние, парсеки.
	YLJ	REAL	Конвертированное расстояние, световые года.

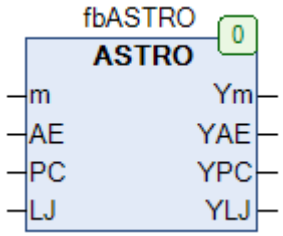


Рис. 22.1. Внешний вид ФБ **ASTRO** на языке CFC

Функциональный блок **ASTRO** конвертирует значение расстояния из одних астрономических единиц измерения в другие. Подразумевается, что при каждом вызове блока используется только один вход, а остальные имеют нулевые значения. При этом на выходах отображаются конвертированные значения в соответствующих единицах измерения. Если при вызове ФБ есть значения на нескольких входах, то будет произведена конвертация суммы этих значений.

Соотношения между единицами:

- 1 а.е. = 149 597 870 700 м
- 1 парсек ≈ 206 264,8 а. е. = $3{,}0856776 \cdot 10^{16}$ м = 3,2616 светового года.
- 1 световой год = 9 460 730 472 580 800 м ≈ 63 241,077 а. е.

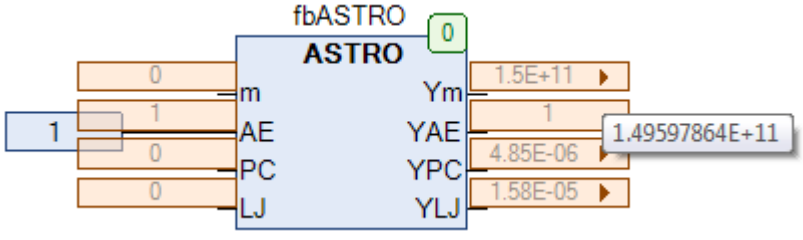


Рис. 22.2. Пример работы с ФБ **ASTRO** на языке CFC (конвертация одного значения)

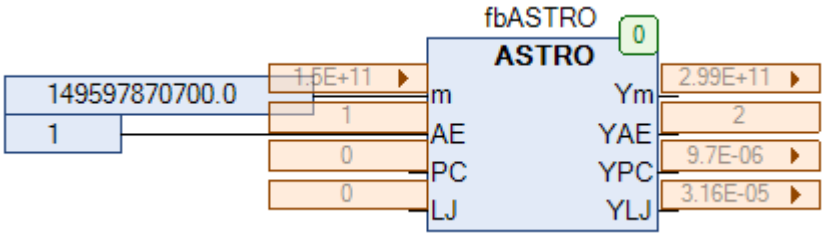


Рис. 22.3. Внешний вид ФБ **ASTRO** на языке CFC (конвертация суммы значений)

22.2. BFT_TO_MS

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	BFT	INT	Скорость ветра в баллах Бофорта (1..12).
Выходы	BFT_TO_MS	REAL	Скорость ветра в м/с.

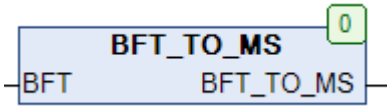


Рис. 22.4. Внешний вид функции **BFT_TO_MS** на языке CFC

Функция **BFT_TO_MS** конвертирует значение скорости ветра из шкалы Бофорта в метры в секунду. Преобразование выполняется с помощью эмпирической формулы

$$\mathrm{BFT\_TO\_MS} = 0.837 \cdot \mathrm{BFT}^{\frac{3}{2}}$$

См. также обратную функцию MS_TO_BFT.

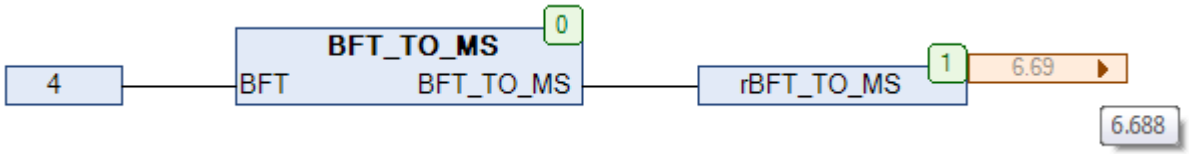


Рис. 22.5. Пример работы с функцией **BFT_TO_MS** на языке CFC

22.3. C_TO_F

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	celsius	REAL	Температура в градусах Цельсия.
Выходы	C_TO_F	REAL	Температура в градусах по Фаренгейту.

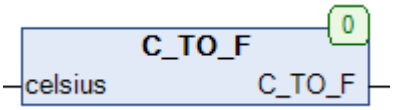


Рис. 22.6. Внешний вид функции **C_TO_F** на языке CFC

Функция **C_TO_F** конвертирует значение температуры из градусов Цельсия в градусы Фаренгейта. Преобразование выполняется с помощью формулы

$$\mathrm{C\_TO\_F} = \frac{9}{5} \cdot \mathrm{celsius} + 32$$

См. также обратную функцию F_TO_C.

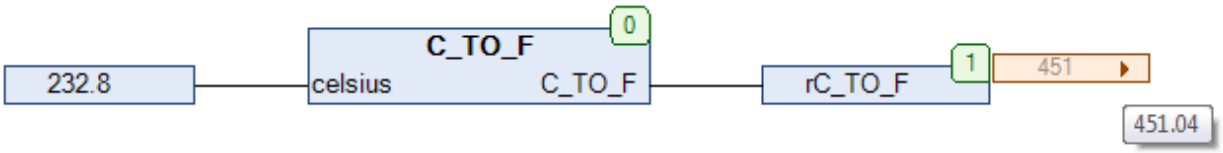


Рис. 22.7. Пример работы с функцией **C_TO_F** на языке CFC

22.4. C_TO_K

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	celsius	REAL	Температура в градусах Цельсия.
Выходы	C_TO_K	REAL	Температура в градусах Кельвина.

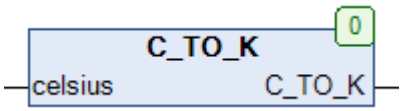


Рис. 22.8. Внешний вид функции **C_TO_K** на языке CFC

Функция **C_TO_K** конвертирует значение температуры из градусов Цельсия в кельвины. Преобразование выполняется с помощью формулы

$$C\_TO\_K = celsius + 273{,}15$$

См. также обратную функцию K_TO_C.

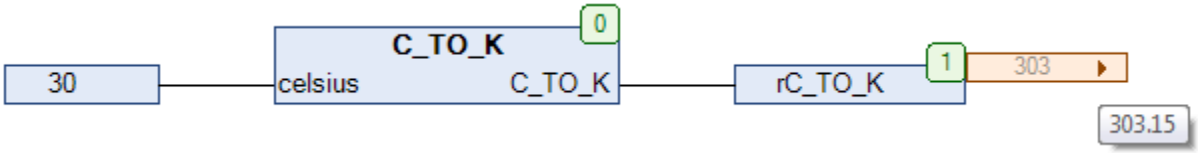


Рис. 22.9. Пример работы с функцией **C_TO_K** на языке CFC

22.5. DEG_TO_DIR

Тип модуля: функция	Переменная	Тип	Описание
Входы	DEG	INT	Азимут, градусы.
	N	INT	Коэффициент кол-ва лучей розы румбов (1..3).
	L	INT	Язык проекта.
Выходы	DEG_TO_DIR	STRING(3)	Обозначение румба.

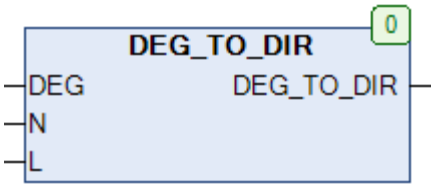


Рис. 22.10. Внешний вид функции **DEG_TO_DIR** на языке CFC

Функция **DEG_TO_DIR** конвертирует значения азимута **DEG** (в градусах) в обозначение соответствующего румба. Вход **N** определяет кол-во лучей розы румбов (см. рис. 22.12):

- N=1 – используются 4 луча;
- N=2 – используются 8 лучей;
- N=3 – используются 16 лучей.

Вход **LANG** определяет язык приложения (см. глобальную переменную LANGUAGE; в ней же содержатся и обозначения румбов). См. также обратную функцию DIR_TO_DEG.

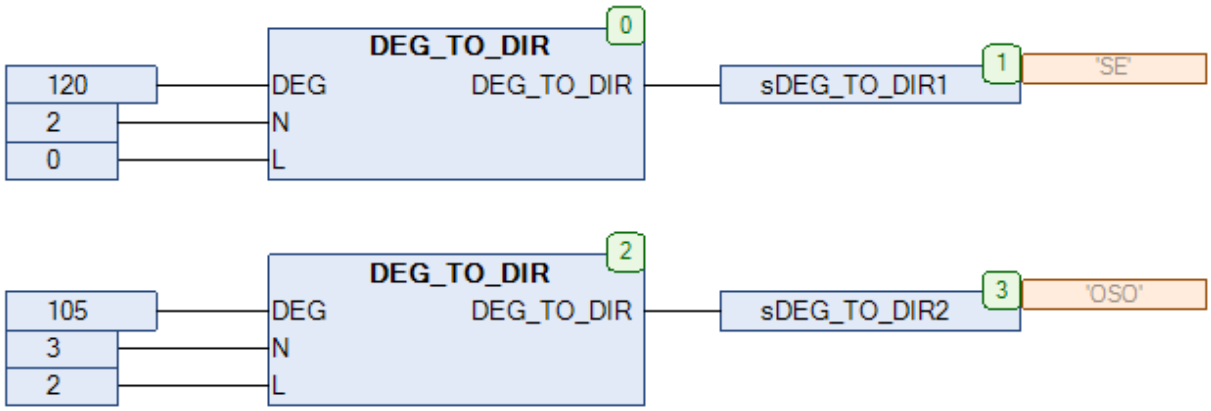


Рис. 22.11. Пример работы с функцией **DEG_TO_DIR** на языке CFC

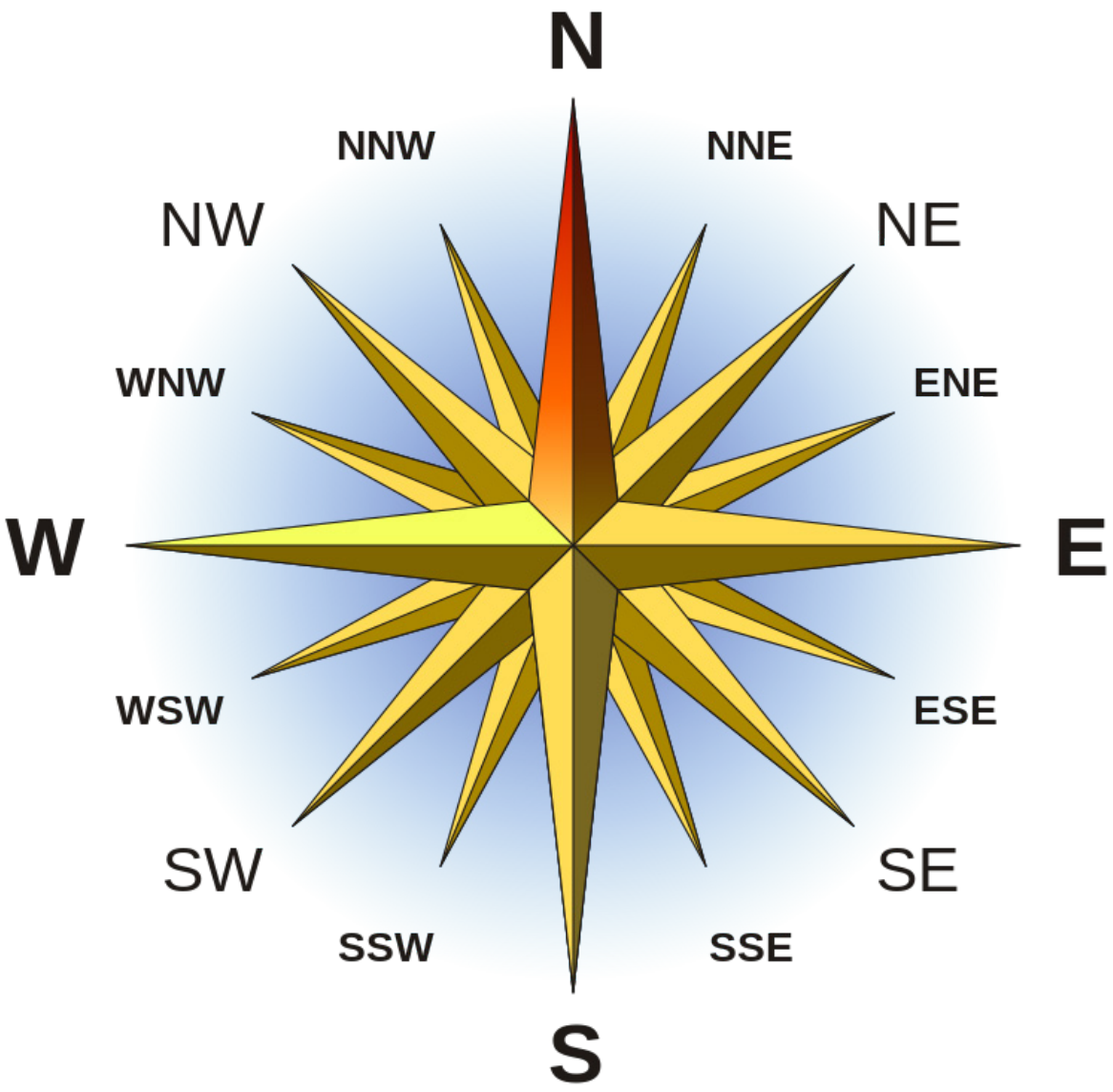


Рис. 22.12. Шестнадцатилучевая роза румбов

22.6. DIR_TO_DEG

Тип модуля: функция	Переменная	Тип	Описание
Входы	DIR	STRING(3)	Обозначение румба.
	L	INT	Язык проекта.
Выходы	DIR_TO_DEG	INT	Азимут, градусы.

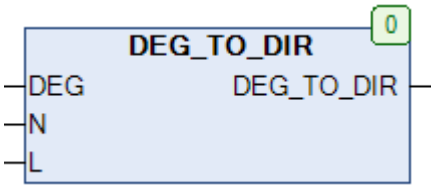


Рис. 22.13. Внешний вид функции **DIR_TO_DEG** на языке CFC

Функция **DIR_TO_DEG** конвертирует обозначения румба в значение азимута в градусах. Функция работает с 16-лучевой розой румбов (см. рис. 22.12). Вход **L** определяет язык приложения (см. глобальную переменную LANGUAGE; в ней же содержатся и названия румбов). См. также обратную функцию DEG_TO_DIR.

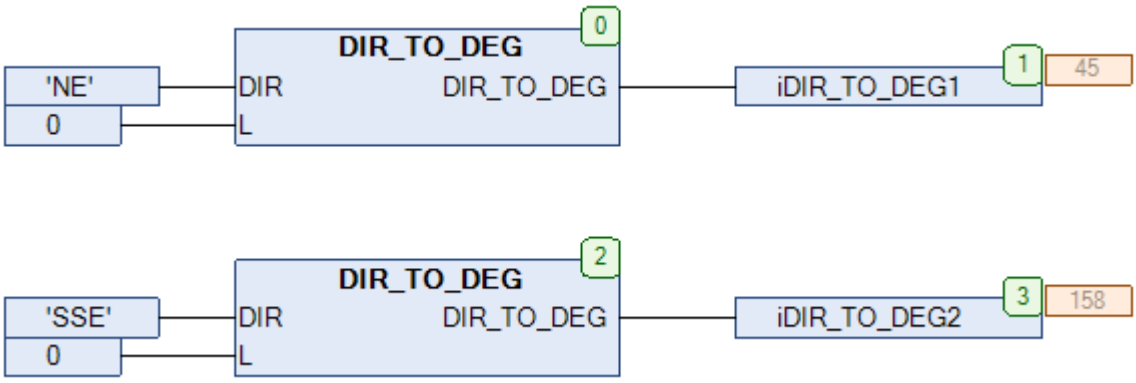


Рис. 22.14. Пример работы с функцией **DIR_TO_DEG** на языке CFC

22.7. ENERGY

Тип модуля: ФБ	Переменная	Тип	Описание
Входы	J	REAL	Исходное значение энергии, Дж.
	C	REAL	Исходное значение энергии, кал.
	WH	REAL	Исходное значение энергии, Вт·ч.
Выходы	YJ	REAL	Конвертированное значение энергии, Дж.
	YC	REAL	Конвертированное значение энергии, кал.
	YWH	REAL	Конвертированное значение энергии, Вт·ч.

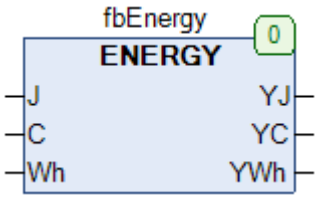


Рис. 22.15. Внешний вид ФБ **ENERGY** на языке CFC

Функциональный блок **ENERGY** конвертирует значение энергии из одних астрономических единиц измерения в другие. Подразумевается, что при каждом вызове блока используется только один вход, а остальные имеют нулевые значения. При этом на выходах отображаются конвертированные значения в соответствующих единицах измерения. Если при вызове ФБ есть значения на нескольких входах, то будет произведена конвертация суммы этих значений.

Соотношения между единицами:

- 1 кал = 4,1868 Дж = $1{,}163 \cdot 10^{-3}$ Вт·ч
- 1 Вт·ч = $3{,}6 \cdot 10^{3}$ Дж = 860 кал

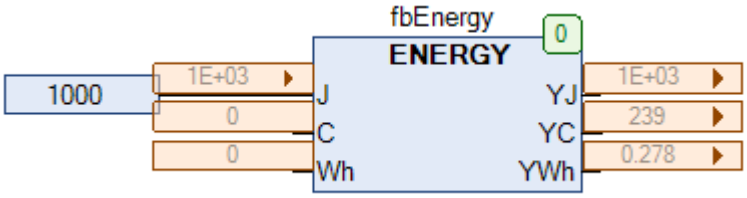


Рис. 22.16. Пример работы с ФБ **ENERGY** на языке CFC (конвертация одного значения)

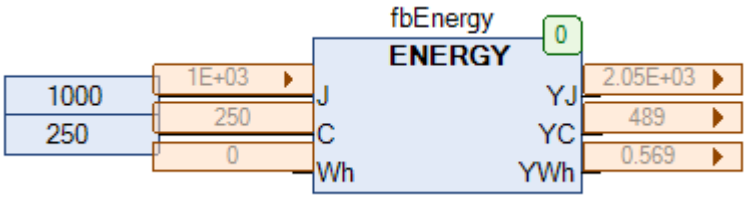


Рис. 22.17. Внешний вид ФБ **ENERGY** на языке CFC (конвертация суммы значений)

22.8. F_TO_C

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	fahrenheit	REAL	Температура в градусах по Фаренгейту.
Выходы	F_TO_C	REAL	Температура в градусах Цельсия.

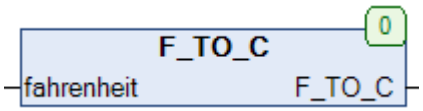


Рис. 22.18. Внешний вид функции **F_TO_C** на языке CFC

Функция C_TO_F конвертирует значение температуры из градусов Фаренгейта в градусы Цельсия. Преобразование выполняется с помощью формулы

$$\mathrm{F\_TO\_C} = \frac{5}{9} \cdot (\mathrm{fahrenheit} - 32)$$

См. также обратную функцию C_TO_F.

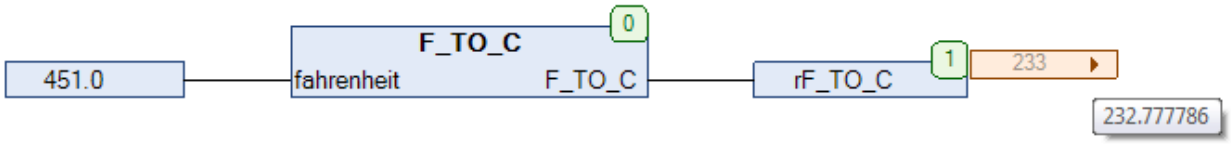


Рис. 22.19. Пример работы с функцией **F_TO_C** на языке CFC

22.9. F_TO_OM

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	F	REAL	Частота, Гц.
Выходы	F_TO_OM	REAL	Угловая частота, рад/с.

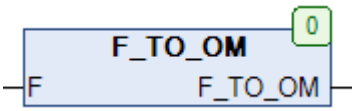


Рис. 22.20. Внешний вид функции **F_TO_OM** на языке CFC

Функция **F_TO_OM** возвращает значение угловой частоты для заданной частоты **F**. Вычисление выполняется по формуле

$$F\_TO\_OM = 2\pi F$$

См. также обратную функцию OM_TO_F.

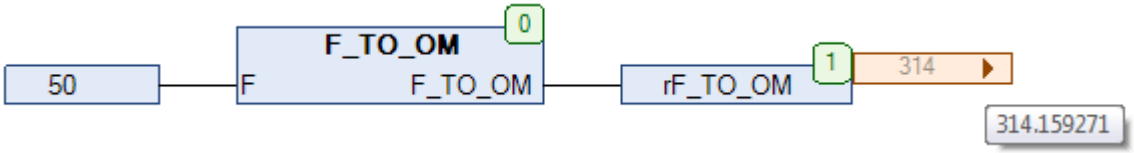


Рис. 22.21. Пример работы с функцией **F_TO_OM** на языке CFC

22.10. F_TO_PT

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	F	REAL	Частота колебаний, Гц.
Выходы	F_TO_PT	TIME	Период колебаний, с.

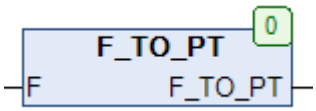


Рис. 22.22. Внешний вид функции **F_TO_PT** на языке CFC

Функция **F_TO_PT** возвращает значение периода колебаний для заданной частоты **F**. Вычисление выполняется по формуле

$$F\_TO\_PT = \frac{1}{F}$$

См. также обратную функцию PT_TO_F.

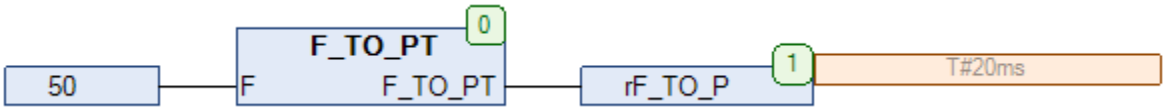


Рис. 22.23. Пример работы с функцией **F_TO_PT** на языке CFC

22.11. GEO_TO_DEG

Тип модуля: функция	Переменная	Тип	Описание
Входы	D	INT	Градусы.
	M	INT	Минуты.
	SEC	REAL	Секунды.
Выходы	GEO_TO_DEG	REAL	Десятичные градусы.

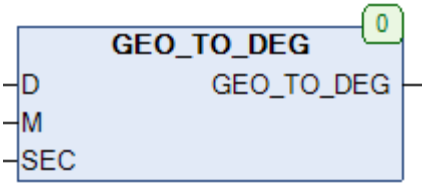


Рис. 22.24. Внешний вид функции **GEO_TO_DEG** на языке CFC

Функция **GEO_TO_DEG** конвертирует географические координаты из формата DMS (градусы **D**/минуты **M**/секунды **SEC**) в формат DD (десятичные градусы). Вычисление выполняется по формуле

$$\mathrm{GEO\_TO\_DEG} = \mathrm{D} + \frac{\mathrm{M}}{60} + \frac{\mathrm{SEC}}{3600}$$

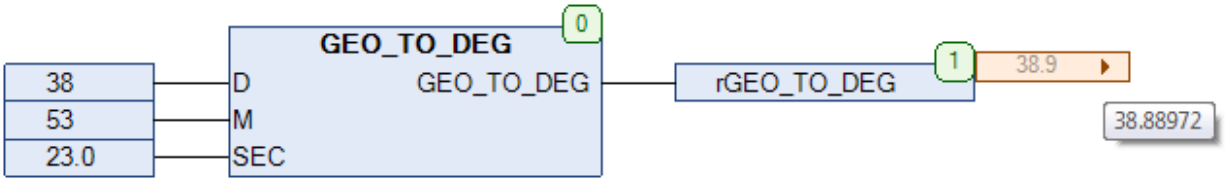


Рис. 22.25. Пример работы с функцией **GEO_TO_DEG** на языке CFC

22.12. K_TO_C

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	Kelvin	REAL	Температура в градусах Кельвина.
Выходы	K_TO_C	REAL	Температура в градусах Цельсия.

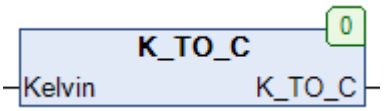


Рис. 22.26. Внешний вид функции **K_TO_C** на языке CFC

Функция **K_TO_C** конвертирует значение температуры из кельвинов в градусы Цельсия. Преобразование выполняется с помощью формулы

$$\mathrm{K\_TO\_C} = \mathrm{Kelvin} - 273{,}15$$

См. также обратную функцию C_TO_K.

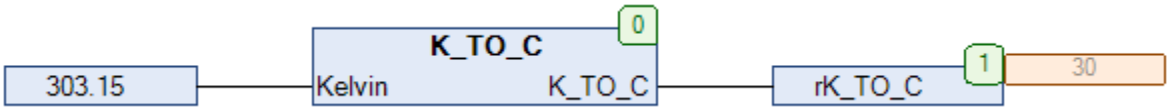


Рис. 22.27. Пример работы с функцией **K_TO_C** на языке CFC

22.13. KMH_TO_MS

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	kmh	REAL	Скорость, км/ч.
Выходы	KMH_TO_MS	REAL	Скорость, м/с.

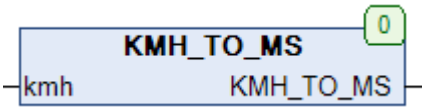


Рис. 22.28. Внешний вид функции **KMH_TO_MS** на языке CFC

Функция **KMH_TO_MS** конвертирует значение скорости из км/ч в м/с. См. также обратную функцию MS_TO_KMH.

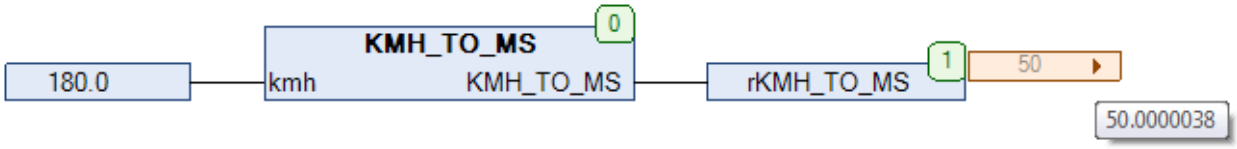


Рис. 22.29. Пример работы с функцией **KMH_TO_MS** на языке CFC

22.14. LENGTH

Тип модуля: ФБ	Переменная	Тип	Описание
Входы	m	REAL	Исходное значение длины, м.
	p	REAL	Исходное значение длины, типографские пункты.
	in	REAL	Исходное значение длины, дюймы.
	ft	REAL	Исходное значение длины, футы.
	yd	REAL	Исходное значение длины, ярды.
	mile	REAL	Исходное значение длины, мили.
	sm	REAL	Исходное значение длины, морские мили.
	fm	REAL	Исходное значение длины, морские сажени.
Выходы	Ym	REAL	Конвертированное значение длины, м.
	Yp	REAL	Конвертированное значение длины, типограф. пункты.
	Yin	REAL	Конвертированное значение длины, дюймы.
	Yft	REAL	Конвертированное значение длины, футы.
	Yyd	REAL	Конвертированное значение длины, ярды.
	Ymile	REAL	Конвертированное значение длины, мили.
	Ysm	REAL	Конвертированное значение длины, морские мили.
	Yfm	REAL	Конвертированное значение длины, морские сажени.

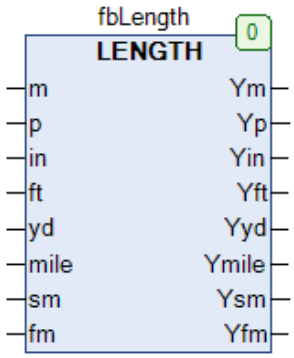


Рис. 22.30. Внешний вид ФБ **LENGTH** на языке CFC

Функциональный блок **LENGTH** конвертирует значение длины из одних единиц измерения в другие. Подразумевается, что при каждом вызове блока используется только один вход, а остальные имеют нулевые значения. При этом на выходах отображаются конвертированные значения в соответствующих единицах измерения. Если при вызове ФБ есть значения на нескольких входах, то будет произведена конвертация суммы этих значений.

Соотношения между единицами:

- 1 типографский пункт = 0,376065 мм
- 1 дюйм = 25,4 мм
- 1 фут = 0,3048 м
- 1 ярд = 0,9144 м
- 1 миля = 1609,344 м
- 1 морская миля = 1852 м
- 1 морская сажень = 1,829 м

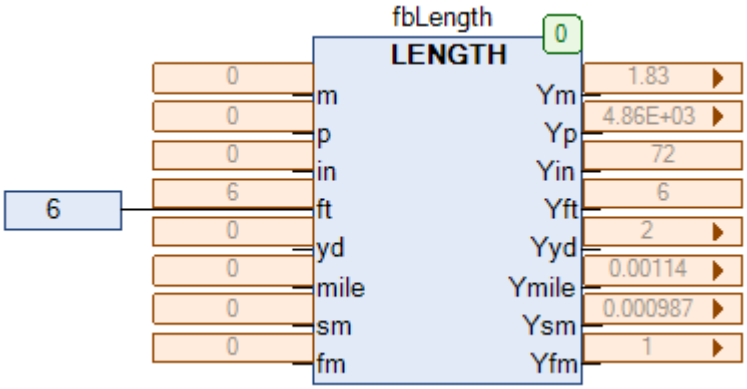


Рис. 22.31. Пример работы с ФБ **LENGTH** на языке CFC (конвертация одного значения)

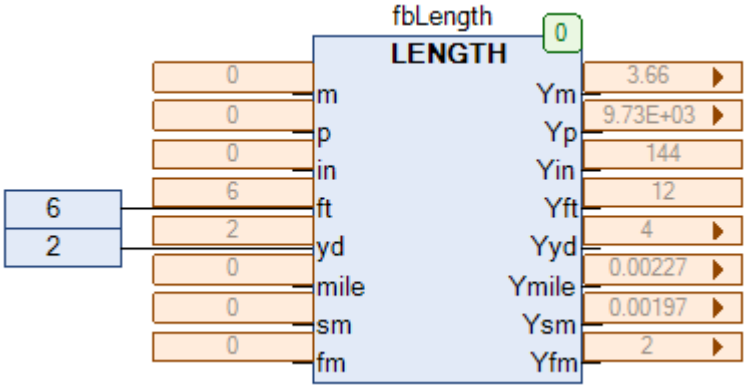


Рис. 22.32. Внешний вид ФБ **LENGTH** на языке CFC (конвертация суммы значений)

22.15. MS_TO_BFT

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	MS	REAL	Скорость ветра в м/с.
Выходы	MS_TO_BFT	INT	Скорость ветра в баллах Бофорта.

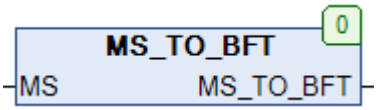


Рис. 22.33. Внешний вид функции **MS_TO_BFT** на языке CFC

Функция **MS_TO_BFT** конвертирует значение скорости ветра из метров в секунду в баллы шкалы Бофорта. Преобразование выполняется с помощью эмпирической формулы

$$\mathrm{MS\_TO\_BFT} = (1.196172 \cdot \mathrm{MS})^{\frac{2}{3}}$$

См. также обратную функцию BFT_TO_MS.

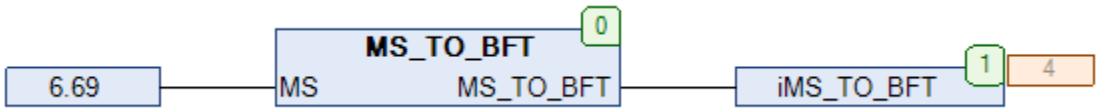


Рис. 22.34. Пример работы с функцией **MS_TO_BFT** на языке CFC

22.16. MS_TO_KMH

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	MS	REAL	Скорость, м/с.
Выходы	MS_TO_KMH	REAL	Скорость, км/ч.

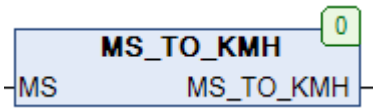


Рис. 22.35. Внешний вид функции **MS_TO_KMH** на языке CFC

Функция **MS_TO_KMH** конвертирует значение скорости из м/с в км/ч. См. также обратную функцию KMH_TO_MS.

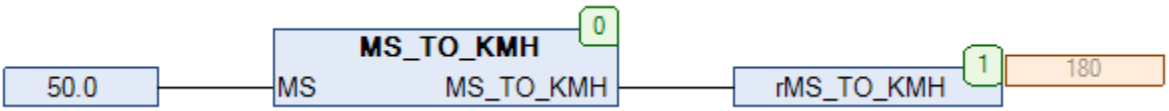


Рис. 22.36. Пример работы с функцией **MS_TO_KMH** на языке CFC

22.17. OM_TO_F

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	OM	REAL	Угловая частота, рад/с.
Выходы	OM_TO_F	REAL	Частота, Гц.

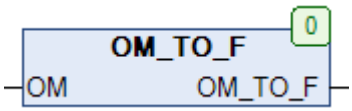


Рис. 22.37. Внешний вид функции **OM_TO_F** на языке CFC

Функция **OM_TO_F** возвращает значение частоты для заданной угловой частоты **OM**. Вычисление выполняется по формуле

$$\mathrm{OM\_TO\_F} = \frac{\mathrm{OM}}{2\pi}$$

См. также обратную функцию F_TO_OM.

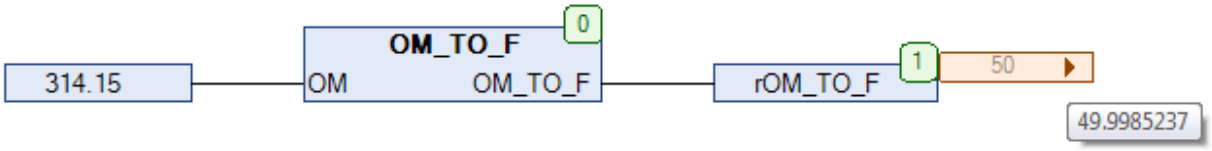


Рис. 22.38. Пример работы с функцией **OM_TO_F** на языке CFC

22.18. PRESSURE

Тип модуля: ФБ	Переменнная	Тип	Описание
Входы	mws	REAL	Исходное значение давления, метры водяного столба.
	torr	REAL	Исходное значение давления, мм ртутного столба.
	att	REAL	Исходное значение давления, технические атмосферы.
	atm	REAL	Исходное значение давления, физические атмосферы.
	pa	REAL	Исходное значение давления, Паскали.
	bar	REAL	Исходное значение давления, бары.
Выходы	Ymws	REAL	Конвертированное значение, метры водяного столба.
	Ytorr	REAL	Конвертированное значение я, мм ртутного столба.
	Yatt	REAL	Конвертированное значение, технические атмосферы.
	Yatm	REAL	Конвертированное значение, физические атмосферы.
	Ypa	REAL	Конвертированное значение, Паскали.
	Ybar	REAL	Конвертированное значение, бары.

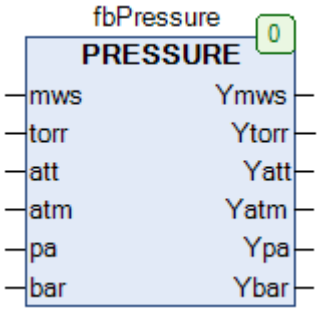


Рис. 22.39. Внешний вид ФБ **PRESSURE** на языке CFC

Функциональный блок **PRESSURE** конвертирует значение давления из одних единиц измерения в другие. Подразумевается, что при каждом вызове блока используется только один вход, а остальные имеют нулевые значения. При этом на выходах отображаются конвертированные значения в соответствующих единицах измерения. Если при вызове ФБ есть значения на нескольких входах, то будет произведена конвертация суммы этих значений.

Соотношения между единицами приведены в [этой статье].

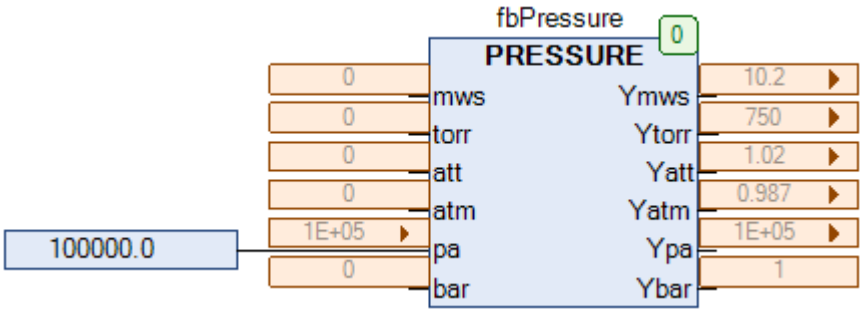


Рис. 22.40. Пример работы с ФБ **PRESSURE** на языке CFC (конвертация одного значения)

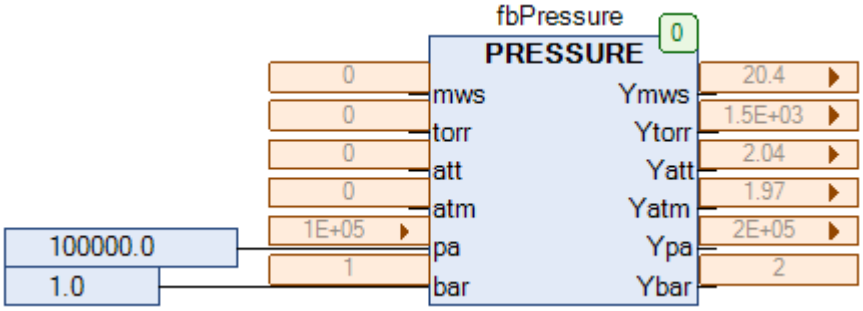


Рис. 22.41. Внешний вид ФБ **PRESSURE** на языке CFC (конвертация суммы значений)

22.19. PT_TO_F

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	PT	TIME	Период колебаний, с.
Выходы	PT_TO_F	REAL	Частота колебаний, Гц.

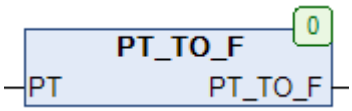


Рис. 22.42. Внешний вид функции **PT_TO_F** на языке CFC

Функция **PT_TO_F** возвращает значение частоты для заданного периода колебаний **PT**. Вычисление выполняется по формуле

$$\mathrm{PT\_TO\_F} = \frac{1}{\mathrm{PT}}$$

См. также обратную функцию F_TO_PT.

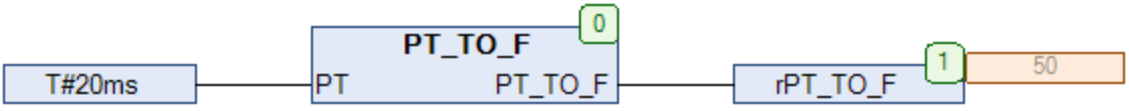


Рис. 22.43. Пример работы с функцией **PT_TO_F** на языке CFC

22.20. SPEED

Тип модуля: ФБ	Переменная	Тип	Описание
Входы	ms	REAL	Исходное значение скорости, м/с.
	kmh	REAL	Исходное значение скорости, км/ч.
	kh	REAL	Исходное значение скорости, узлов.
	mh	REAL	Исходное значение скорости, миль/ч.
Выходы	Yms	REAL	Конвертированное значение скорости, м/с.
	Ykmh	REAL	Конвертированное значение скорости, км/ч.
	Ykh	REAL	Конвертированное значение скорости, узлов.
	Ymh	REAL	Конвертированное значение скорости, миль/ч.

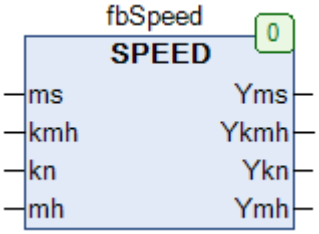


Рис. 22.44. Внешний вид ФБ **SPEED** на языке CFC

Функциональный блок **SPEED** конвертирует значение скорости из одних единиц измерения в другие. Подразумевается, что при каждом вызове блока используется только один вход, а остальные имеют нулевые значения. При этом на выходах отображаются конвертированные значения в соответствующих единицах измерения. Если при вызове ФБ есть значения на нескольких входах, то будет произведена конвертация суммы этих значений.

Соотношения между единицами:

- 1 км/ч = 0,278 м/с
- 1 узел = 1 морская миля/ч = 1,852 км/ч = 0,5144 м/с
- 1 миля/ч = 1,609 км/ч = 0,447 м/с

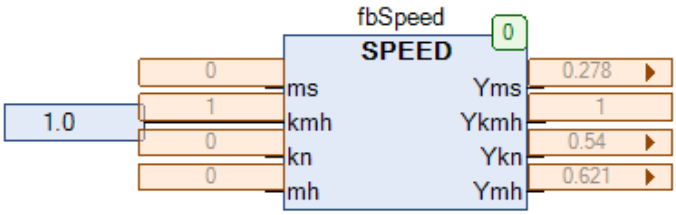


Рис. 22.45. Пример работы с ФБ **SPEED** на языке CFC (конвертация одного значения)

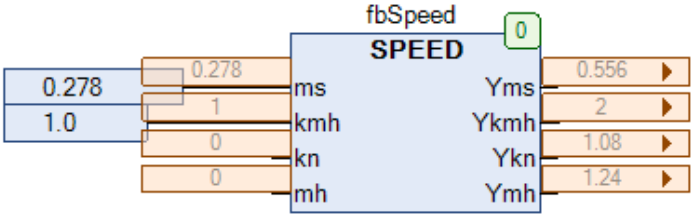


Рис. 22.46. Внешний вид ФБ **SPEED** на языке CFC (конвертация суммы значений)

22.21. TEMPERATURE

Тип модуля: ФБ	Переменная	Тип	Описание
Входы	K	REAL	Исходное значение температуры, кельвины.
	C	REAL	Исходное значение температуры, градусы Цельсия.
	F	REAL	Исходное значение температуры, градусы Фаренгейта.
	Re	REAL	Исходное значение температуры, градусы Реомюра.
	Ra	REAL	Исходное значение температуры, градусы Ранкина.
Выходы	YK	REAL	Конвертированное значение, кельвины.
	YC	REAL	Конвертированное значение, градусы Цельсия.
	YF	REAL	Конвертированное значение, градусы Фаренгейта.
	YRe	REAL	Конвертированное значение, градусы Реомюра.
	YRa	REAL	Конвертированное значение, градусы Ранкина.

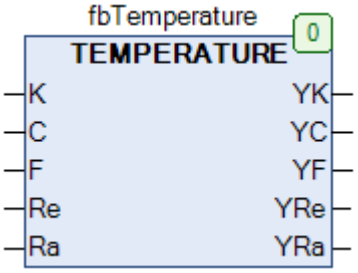


Рис. 22.47. Внешний вид ФБ **TEMPERATURE** на языке CFC

Функциональный блок **TEMPERATURE** конвертирует значение температуры из одних единиц измерения в другие. Подразумевается, что при каждом вызове блока используется только один вход, а остальные имеют нулевые значения. При этом на выходах отображаются конвертированные значения в соответствующих единицах измерения. Если при вызове ФБ есть значения на нескольких входах, то будет произведена конвертация суммы этих значений.

Соотношения между единицами:

- $T_K = T_C + 273{,}15$
- $T_C = T_K - 273{,}15$
- $T_F = \frac{9}{5} \cdot T_C + 32$
- $T_{RE} = (T_K - 273{,}15) \cdot 0{,}8$
- $T_{RA} = 1{,}8 \cdot T_K$

Следует помнить, что разные температурные шкалы имеют разные точки отсчета. По умолчанию входы блока имеют следующие значения:

- K = 0.0;
- C = -273,15;
- F = -459,67;
- Re = -218,52;
- Ra = 0.0.

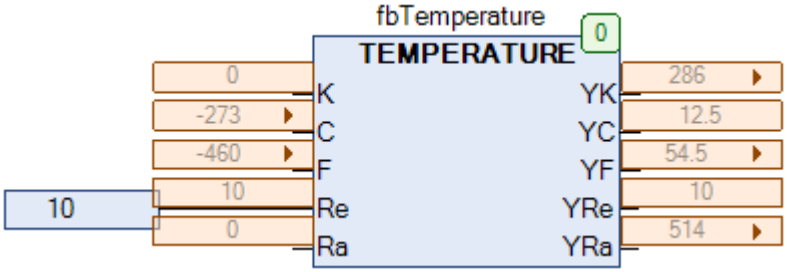


Рис. 22.45. Пример работы с ФБ **TEMPERATURE** на языке CFC (конвертация одного значения)

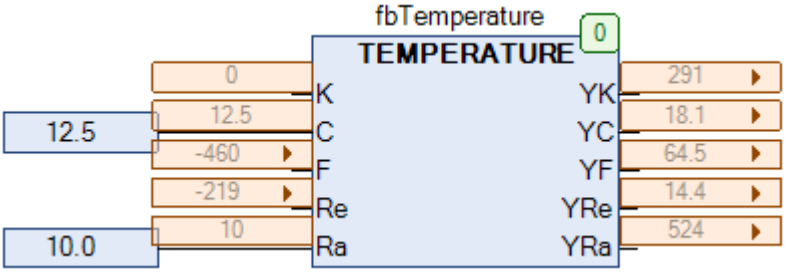


Рис. 22.46. Пример работы с ФБ **TEMPERATUR**E на языке CFC (конвертация суммы значений)

23. Регуляторы

23.1. Вступление

Данная глава содержит описание модулей управления (регуляторов), содержащихся в библиотеке **OSCAT**. Регулятором называется устройство, которое следит за функционированием объекта управления и, постоянно анализируя его состояние, вырабатывает определенное управляющее воздействие (сигнал управления). В тех случаях, когда это возможно, модули управления должны вызываться каждый цикл ПЛК. Преимуществом такого подхода является возможность в пределах одной задачи контролировать процессы, протекающие с разной скоростью. Альтернативный подход подразумевает вызов модулей с заданной частотой (например, по таймеру); его недостатком является возможное нарушение частоты вызова блоков в задачах с низким приоритетом. Величину цикла ПЛК должен определить программист, проанализировав требования и характеристики конкретного технологического процесса.

Более подробная информация о регуляторах в системах АСУТП доступна по ссылке.

Ниже описана структура и взаимосвязь основных модулей данной главы:

Модуль	Основан на	Описание
INTEGRATE	-	Блок интегрирования.
FT_DERIV	-	Блок дифференцирования.
CTRL_IN	-	Блок вычисления сигнала рассогласования.
CTRL_OUT	-	Блок вычисления управляющего воздействия.
FT_INT	INTEGRATE	Блок интегрирования с ограничением выхода.
FT_PI	FT_INT	Блок ПИ-регулирования.
FT_PIW	FT_INT	Блок ПИ-регулирования с компенсаций эффекта интегрального насыщения.
FT_PIWL	-	Блок ПИ-регулирования с компенсаций эффекта интегрального насыщения.
CTRL_PI	FT_PIWL, CTRL_IN, CTRL_OUT	ПИ-регулятор с обвязкой.
FT_PID	FT_INT, FT_DERIV	Блок ПИД-регулирования.
FT_PIDW	INTEGRATE, FT_DERIV	Блок ПИД-регулирования с компенсаций эффекта интегрального насыщения.
FT_PIDWL	FT_PIWL, FT_DERIV	Блок ПИД-регулирования с компенсаций эффекта интегрального насыщения.
CTRL_PID	FT_PIDWL, CTRL_IN, CTRL_OUT	ПИД-регулятор с обвязкой.
FT_PD	FT_DERIV	ПД-регулятор.
FT_PDT1	FT_DERIV, FT_PT1	ПД-регулятор с А-звеном.

23.2. BAND_B

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	X	BYTE	Входное значение.
	B	BYTE	Область ограничения.
Выходы	BAND_B	BYTE	Выходное значение.

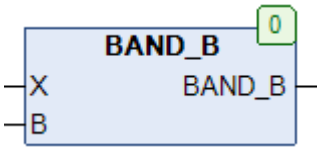


Рис. 23.1. Внешний вид функции **BAND_B** на языке CFC

Функция **BAND_B** ограничивает диапазон входного сигнала **X** по следующему принципу:

- BAND_B = 0, если X < B;
- BAND_B = X, если B ≤ X ≤ (255-B);
- BAND_B = 255, если X > (255-B).

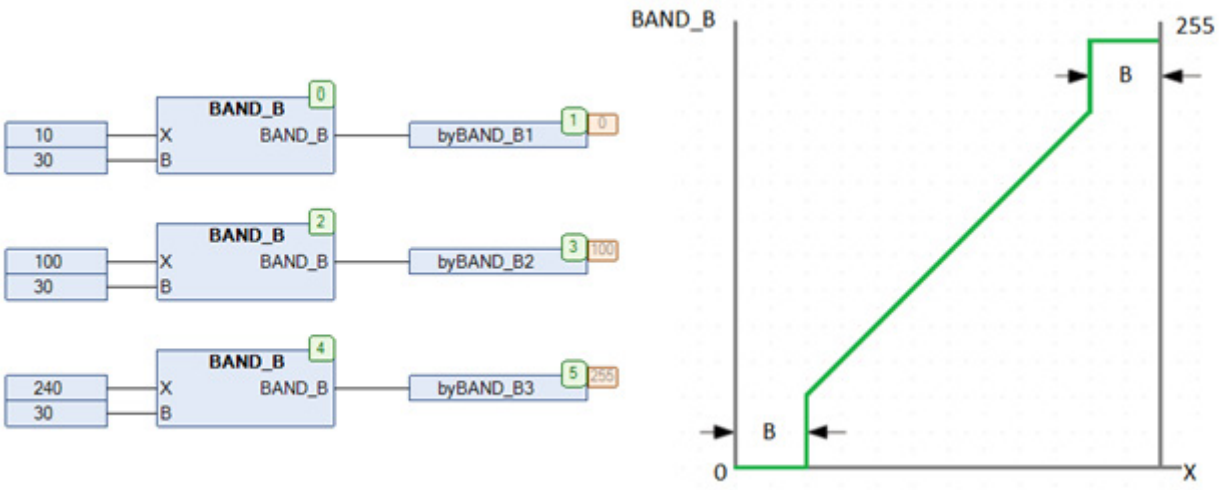


Рис. 23.2. Пример работы с функцией **BAND_B** на языке CFC

23.3. CONTROL_SET1

Тип модуля: ФБ	Переменная	Тип	Описание
Входы	Kt	REAL	Коэффициент усиления на границе устойчивости.
	Tt	REAL	Период установившихся колебаний.
	PI	BOOL	Режим «Настройки ПИ-регулятора».
	PID	BOOL	Режим «Настройки ПИД-регулятора».
Выходы	KP	REAL	Рассчитанный коэффициент усиления.
	TN	REAL	Постоянная времени интегрирования.
	TV	REAL	Постоянная времени дифференцирования.
	KI	REAL	Рассчитанный коэффициент интегрирования.
	KD	REAL	Рассчитанный коэффициент дифференцирования.
Параметры	P_K	REAL	Эмпирический коэффициент (0.5).
	PI_K	REAL	Эмпирический коэффициент (0.45).
	PI_TN	REAL	Эмпирический коэффициент (0.83).
	PID_K	REAL	Эмпирический коэффициент (0.6).
	PID_TN	REAL	Эмпирический коэффициент (0.5).
	PID_TV	REAL	Эмпирический коэффициент (0.125).

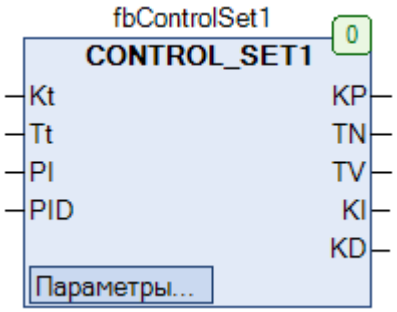


Рис. 23.3. Внешний вид ФБ **CONTROL_SET1** на языке CFC

Функциональный блок **CONTROL_SET1** используется для настройки регулятора методом Циглера-Никольса (*по частотным параметрам).* Для этого регулятор переводится в режим П-регулятора, коэффициент усиления которого увеличивается до тех пор, пока на выходе системы не установятся колебания с постоянной амплитудой (т.е. система выводится на границу устойчивости). Коэффициент усиления на границе устойчивости и период незатухающих колебаний передаются на входы блока **Kt** и **Tt**. Если вход **PI** имеет значение **TRUE**, то будет проведена настройка ПИ-регулятора; если вход **PID** имеет значение **TRUE**, то будет проведена настройка ПИД-регулятора. Если оба входа имеют значение **FALSE**, то будет проведена настройка П-регулятора. Если оба входа имеют значение **TRUE**, то настройка проведена не будет. Рассчитанные параметры подаются на выходы блока. Они вычисляются по следующим эмпирическим соотношениям:

Регулятор	PI	PID	KP	TN	TV	KI	KD
П	FALSE	FALSE	$P\_K \cdot Kt$				
ПИ	TRUE	FALSE	$PI\_K \cdot Kt$	$PI\_TN \cdot Tt$		$\frac{KP}{TN}$	
ПИД	FALSE	TRUE	$PID\_K \cdot Kt$	$PID\_TN \cdot Tt$	$PID\_TV \cdot Tt$		$KP \cdot TV$

23.4. CONTROL_SET2

Тип модуля: ФБ	Переменная	Тип	Описание
Входы	KS	REAL	Коэффициент усиления.
	TU	REAL	Время запаздывания.
	TG	REAL	Постоянная времени.
	PI	BOOL	Режим «Настройки ПИ-регулятора».
	PID	BOOL	Режим «Настройки ПИД-регулятора».
Выходы	KP	REAL	Рассчитанный коэффициент усиления.
	TN	REAL	Постоянная времени интегрирования.
	TV	REAL	Постоянная времени дифференцирования.
	KI	REAL	Рассчитанный коэффициент интегрирования.
	KD	REAL	Рассчитанный коэффициент дифференцирования.
Параметры	P_K	REAL	Эмпирический коэффициент (1.0).
	PI_K	REAL	Эмпирический коэффициент (0.9).
	PI_TN	REAL	Эмпирический коэффициент (3.33).
	PID_K	REAL	Эмпирический коэффициент (1.2).
	PID_TN	REAL	Эмпирический коэффициент (2.0).
	PID_TV	REAL	Эмпирический коэффициент (0.5).

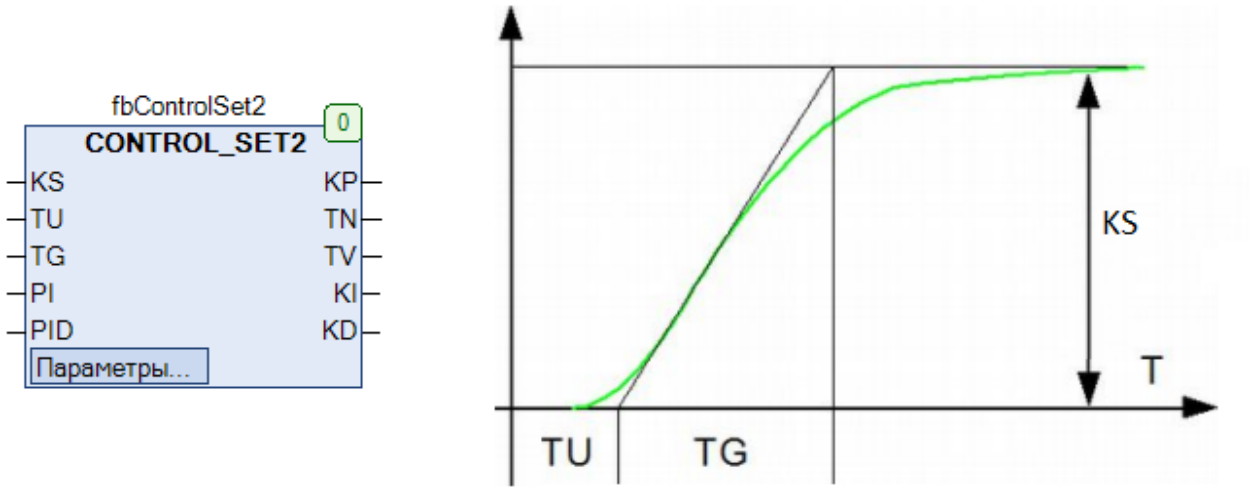


Рис. 23.4. Внешний вид ФБ **CONTROL_SET2** на языке CFC

Функциональный блок CONTROL_SET1 используется для настройки регулятора методом Циглера-Никольса (*по кривой разгона).* Для этого на объект управления подается ступенчатое воздействие и снимается кривая разгона (переходная характеристика). Графическим методом определяются параметры **KS** (коэффициент усиления), **TU** (время запаздывания) и **TG** (постоянная времени), которые подаются на соответствующие входы блока. Если вход **PI** имеет значение **TRUE**, то будет проведена настройка ПИ-регулятора; если вход **PID** имеет значение **TRUE**, то будет проведена настройка ПИД-регулятора. Если оба входа имеют значение **FALSE**, то будет проведена настройка П-регулятора. Если оба входа имеют значение **TRUE**, то настройка проведена не будет. Рассчитанные параметры подаются на выходы блока. Они вычисляются по следующим эмпирическим соотношениям:

Регулятор	PI	PID	KP	TN	TV	KI	KD
П	FALSE	FALSE	$\frac{P\_K \cdot KS \cdot TG}{TU}$				
ПИ	TRUE	FALSE	$\frac{PI\_K \cdot KS \cdot TG}{TU}$	$PI\_TN \cdot TU$		$\frac{KP}{TN}$	
ПИД	FALSE	TRUE	$\frac{PID\_K \cdot KS \cdot TG}{TU}$	$PID\_TN \cdot TU$	$PID\_TV \cdot TU$	$\frac{KP}{TN}$	$KP \cdot TV$

23.5. CTRL_IN

Тип модуля: функция	Переменная	Тип	Описание
Входы	SET_POINT	REAL	Значение уставки.
	ACTUAL	REAL	Значение регулируемой величины.
	NOISE	REAL	Зона нечувствительности.
Выходы	CTRL_IN	REAL	Сигнал рассогласования.
Используемые модули	DEAD_ZONE		

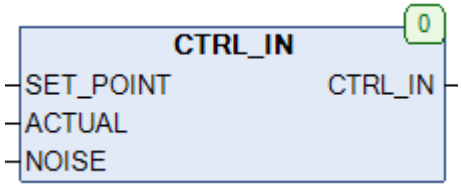


Рис. 23.5. Внешний вид функции **CTRL_IN** на языке CFC

Функция **CTRL_IN** возвращает сигнал рассогласования между значением уставки **SET_POINT** и измеренным значением регулируемой величины **ACTUAL** с учетом зоны нечувствительности **NOISE**. Если |SET_POINT-ACTUAL| < NOISE, то функция возвращает **0**.

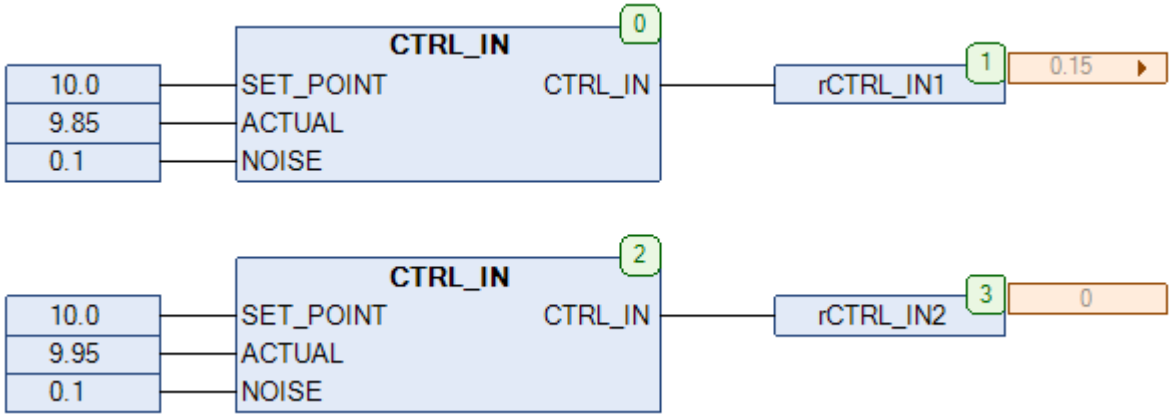


Рис. 23.6. Пример работы с функцией **CTRL_IN** на языке CFC

23.6. CTRL_OUT

Тип модуля: ФБ	Переменная	Тип	Описание
Входы	CI	REAL	Выход регулятора.
	OFFSET	REAL	Смещение.
	MAN_IN	REAL	Сигнал ручного ввода.
	LIM_L	REAL	Нижний предел управляющего сигнала.
	LIM_H	REAL	Верхний предел управляющего сигнала.
	MANUAL	BOOL	Режим управления.
Выходы	Y	REAL	Величина управляющего воздействия.
	LIM	BOOL	Флаг «достигнут предел для Y».

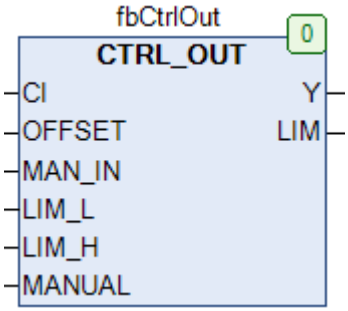


Рис. 23.7. Внешний вид ФБ **CTRL_OUT** на языке CFC

Функциональный блок **CTRL_OUT** используется для генерации управляющего воздействия. Если вход **MANUAL** имеет значение **FALSE**, система работает в автоматическом режиме управления, и на выход **Y** подается сумма значения выходного сигнала регулятора **CI** и смещения **OFFSET**; если вход **MANUAL** имеет значение **TRUE**, то система работает в режиме ручного управления и на выход **Y** подается сумма значения сигнала ручного ввода **MAN_IN** и смещения **OFFSET**. В обоих случаях значение выхода **Y** ограничивается значениями входов **LIM_L** (минимум) и **LIM_H** (максимум). При достижении любой из границ выход **LIM** принимает значение **TRUE**.

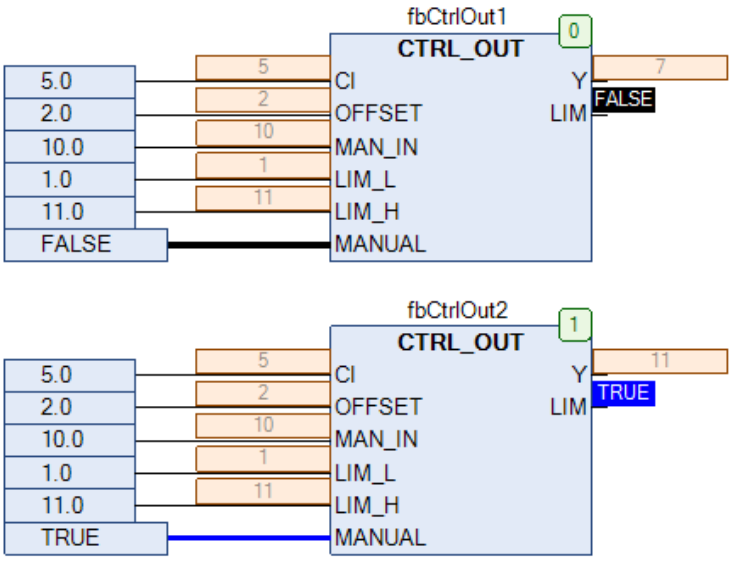


Рис. 23.8. Пример работы с ФБ **CTRL_OUT** на языке CFC

23.7. CTRL_PI

Тип модуля: ФБ	Переменная	Тип	Описание
Входы	ACT	REAL	Значение регулируемой величины.
	SET	REAL	Значение уставки.
	SUP	REAL	Зона нечувствительности.
	OFS	REAL	Смещение.
	M_I	REAL	Сигнал ручного ввода.
	MAN	BOOL	Режим управления.
	RST	BOOL	Сигнал обнуления И-составляющей.
	KP	REAL	Коэффициент усиления П-составляющей.
	KI	REAL	Коэффициент усиления И-составляющей.
	LL	REAL	Нижний предел управляющего сигнала.
	LH	REAL	Верхний предел управляющего сигнала.
Выходы	Y	REAL	Величина управляющего воздействия.
	DIFF	REAL	Сигнал рассогласования.
	LIM	BOOL	Флаг «достигнут предел для Y».
Используемые модули	CTRL_IN, FT_PIWL, CTRL_OUT		

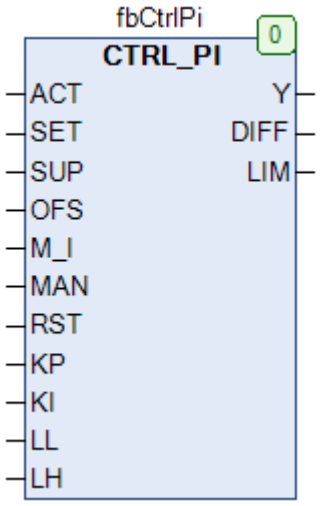


Рис. 23.9. Внешний вид ФБ **CTRL_PI** на языке CFC

Функциональный блок **CTRL_PI** представляет собой ПИ-регулятор с автоматической компенсацией эффекта интегрального насыщения и возможностью перехода на ручное управление. Значение управляющего воздействия **Y** вычисляется по формуле:

$$Y = OFS + KP \cdot DIFF + KI \cdot \sum_{i=0}^{n} DIFF_i \cdot \Delta t_{изм}, \qquad где$$

$$DIFF = SET - ACT, \quad \Delta t_{изм} - время\ между\ двумя\ измерениями$$

На вход **ACT** подается измеренное значение регулируемой величины, на вход **SET** – уставка регулятора. Вход **SUP** определяет зону нечувствительности для регулятора, которая используется, чтобы избежать перерегулирования. Вход **OFS** задает смещение управляющего сигнала. Входы **KP**

и **KI** определяют коэффициенты усиления пропорциональной (П) и интегральной (И) составляющей регулятора. Входы **LL** и L**H** определяют нижний и верхний пределы управляющего сигнала **Y**. При достижении любой из границ выход **LIM** принимает значение **TRUE**, и значение выхода **Y** фиксируется. На выход **DIFF** подается сигнал рассогласования, вычисленный с помощью ФБ CTRL_OUT. По переднему фронту на входе **RST** происходит обнуление накопленной интегральной составляющей блока.

Если вход **MAN** имеет значение **TRUE,** то система работает в режиме ручного управления и на выход **Y** подается сумма значения сигнала ручного ввода **M_I** и смещения **OFS**.

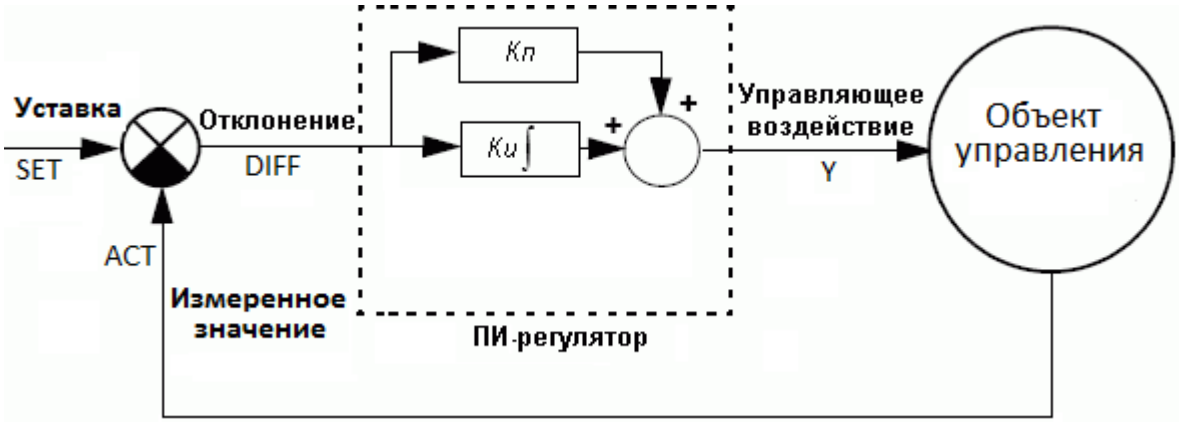


Рис. 23.10. Структурная схема ПИ-регулятора

23.8. CTRL_PID

Тип модуля: ФБ	Переменная	Тип	Описание
Входы	ACT	REAL	Значение регулируемой величины.
	SET	REAL	Значение уставки.
	SUP	REAL	Зона нечувствительности.
	OFS	REAL	Смещение.
	M_I	REAL	Сигнал ручного ввода.
	MAN	BOOL	Режим управления.
	RST	BOOL	Сигнал обнуления И-составляющей.
	KP	REAL	Коэффициент усиления П-составляющей.
	TN	REAL	Постоянная времени интегрирования.
	TV	REAL	Постоянная времени дифференцирования.
	LL	REAL	Нижний предел управляющего сигнала.
	LH	REAL	Верхний предел управляющего сигнала.
Выходы	Y	REAL	Величина управляющего воздействия.
	DIFF	REAL	Сигнал рассогласования.
	LIM	BOOL	Флаг «достигнут предел для Y».
Используемые модули	CTRL_IN, FT_PIDWL, CTRL_OUT		

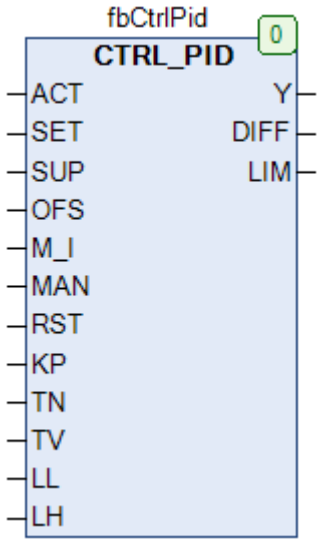


Рис. 23.11. Внешний вид ФБ **CTRL_PID** на языке CFC

Функциональный блок **CTRL_PID** представляет собой ПИД-регулятор с компенсацией эффекта интегрального насыщения и возможностью перехода на ручное управление. Значение управляющего воздействия Y вычисляется по формуле:

$$Y = OFS + KP \cdot DIFF + \frac{1}{TN} \cdot \sum_{i=0}^{n} DIFF_i \cdot \Delta t_{изм} + TV \cdot \frac{\Delta DIFF_i}{\Delta t_{изм}}, \quad \text{где}$$

$$DIFF = SET - ACT, \quad \Delta t_{изм} - \text{время между двумя измерениями}$$

На вход **ACT** подается измеренное значение регулируемой величины, на вход **SET** – уставка регулятора. Вход **SUP** определяет зону нечувствительности для регулятора, которая используется,

чтобы избежать перерегулирования. Вход **OFS** задает смещение управляющего сигнала. Вход **KP** и определяет коэффициент усиления пропорциональной (П) составляющей регулятора, входы **TN** и **TV** – постоянные времени интегрирования и дифференцирования (***обратите внимание***, они не могут быть равны **0** – это приведет к исключению в работе ПЛК). Входы **LL** и **LH** определяют нижний и верхний пределы управляющего сигнала **Y**. При достижении любой из границ выход **LIM** принимает значение **TRUE**, и значение выхода **Y** фиксируется. На выход **DIFF** подается сигнал рассогласования, вычисленный с помощью ФБ CTRL_OUT. По переднему фронту на входе **RST** происходит обнуление накопленной интегральной составляющей блока.

Если вход **MAN** имеет значение **TRUE**, то система работает в режиме ручного управления и на выход **Y** подается сумма значения сигнала ручного ввода **M_I** и смещения **OFS**.

Блок использует в качестве параметров постоянные времени интегрирования и дифференцирования **TV** и **TN**. При необходимости можно перейти от них к коэффициентам усиления:

$$\mathrm{KI} = \frac{\mathrm{KP}}{\mathrm{TN}} \qquad \mathrm{KD} = \mathrm{KP} \cdot \mathrm{TV}$$

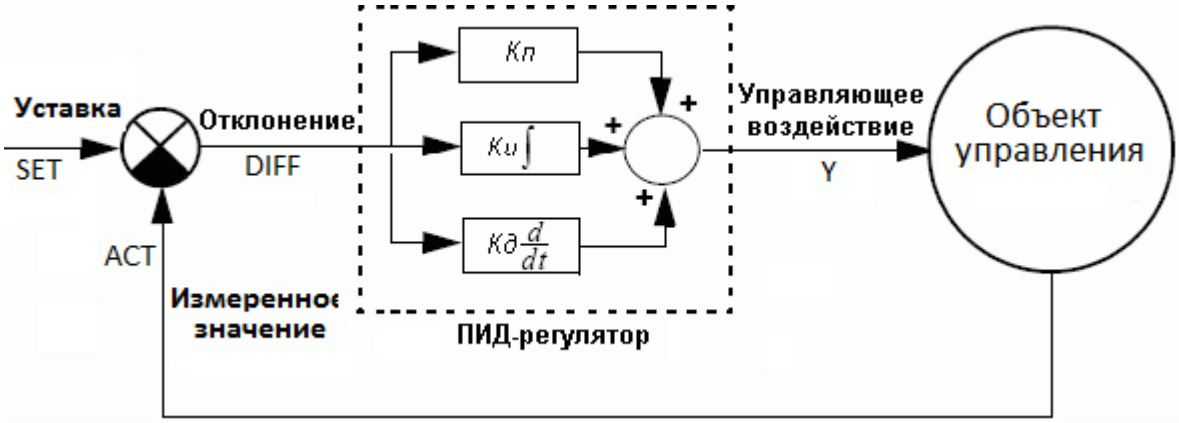


Рис. 23.12. Структурная схема ПИД-регулятора

23.9. CTRL_PWM

Тип модуля: ФБ	Переменная	Тип	Описание
Входы	CI	REAL	Скважность сигнала ШИМ (0…1.0).
	MAN_IN	BOOL	Сигнал ручного ввода.
	MANUAL	BOOL	Режим управления.
	F	REAL	Частота сигнала ШИМ.
Выходы	Q	BOOL	Сигнал ШИМ.
Используемые модули	PWM_DC		

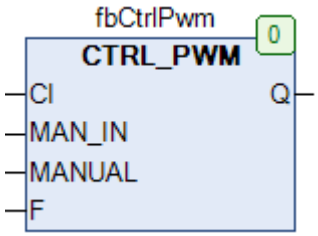


Рис. 23.13 Внешний вид ФБ **CTRL_PWM** на языке CFC

Функциональный блок **CTRL_PWM** представляем собой широтно-импульсный модулятор (ШИМ) с возможностью ручного управления. Если вход **MANUAL** имеет значение **FALSE**, то ШИМ находится в работе и на выход **Q** подается широтно-модулированный сигнал с частотой **F** и скважностью **CI** (0…1.0); если вход **MANUAL** имеет значение **TRUE**, то блок работает в режиме ручного управления и на выход **Q** подается значение входа **MAN_IN**.

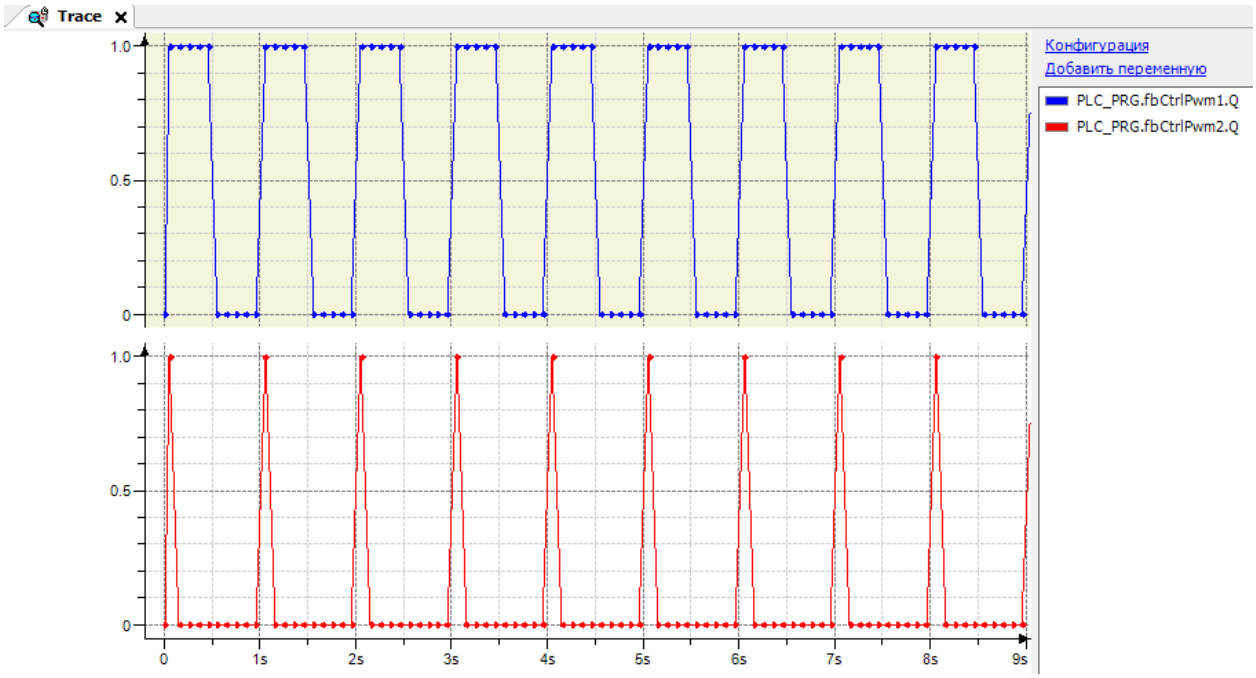


Рис. 23.14. Трассировка работы ФБ CTRL_PWM при F=1 Гц, CI=0.5 и F=1 Гц, CI=0.1

23.10. DEAD_BAND

Тип модуля: функция	Переменная	Тип	Описание
Входы	X	REAL	Входное значение.
	L	REAL	Смещение.
Выходы	DEAD_BAND	REAL	Выходное значение.

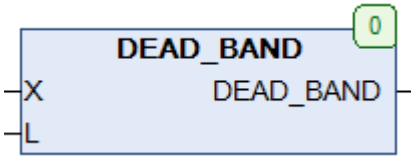


Рис. 23.15. Внешний вид функции **DEAD_BAND** на языке CFC

Функция **DEAD_BAND** ограничивает значение входа X по следующему принципу:

- DEAD_BAND = (X-L), если L < X;
- DEAD_BAND = (X+L), если X < (-L);
- DEAD_BAND = 0.0, если -L ≤ X ≤ L.

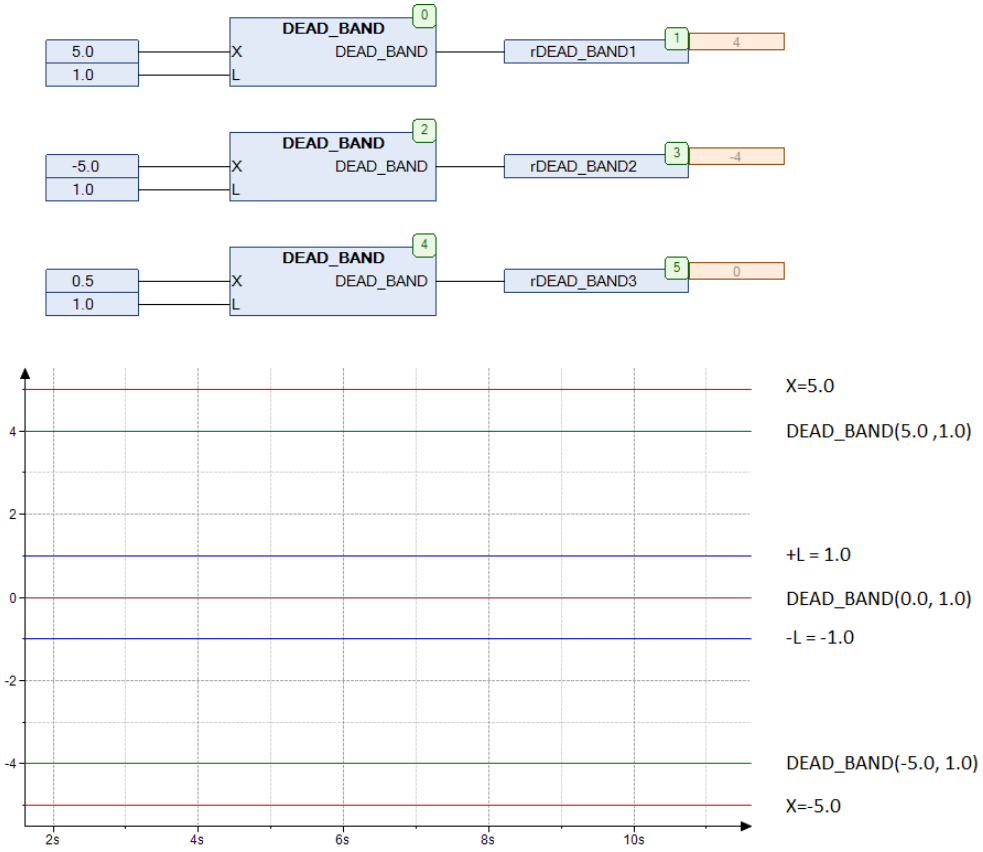


Рис. 23.16. Пример работы с функцией **DEAD_BAND** на языке CFC

23.11. DEAD_BAND_A

Тип модуля: ФБ	Переменная	Тип	Описание
Входы	X	REAL	Входное значение.
	T	TIME	Постоянная времени фильтра.
	KL	REAL	Коэффициент усиления фильтра.
	LM	REAL	Максимальная амплитуда высокочастотной составляющей сигнала.
Выходы	Y	REAL	Отфильтрованное значение.
	L	REAL	Смещение сигнала.
Используемые модули	FT_PT1, MULTIME		

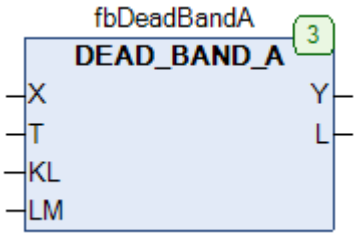


Рис. 23.17. Внешний вид ФБ **DEAD_BAND_A** на языке CFC

Функциональный блок **DEAD_BAND_A** ограничивает значение входа **X** по следующему принципу:

- DEAD_BAND_A = (X-L), если L < X;
- DEAD_BAND_A = (X+L), если X < (-L);
- DEAD_BAND_A = 0.0, если -L ≤ X ≤ L.

Блок содержит фильтр нижних частот для входного сигнала **X** с постоянной времени **T**. Значение **L** определяется как произведение коэффициента усиления фильтра **KL** и текущей амплитуды высокочастотной составляющей входного сигнала; при этом максимальное значение **L** ограничено значением **LM**. Коэффициент **KL** определяет чувствительность фильтра; его значение рекомендуется выбирать из интервала (**1.0…5.0**).

Блок может использоваться в системах регулирования для фильтрации шума входного сигнала, что может являться причиной постоянной работы и как следствие исполнительных механизмов.

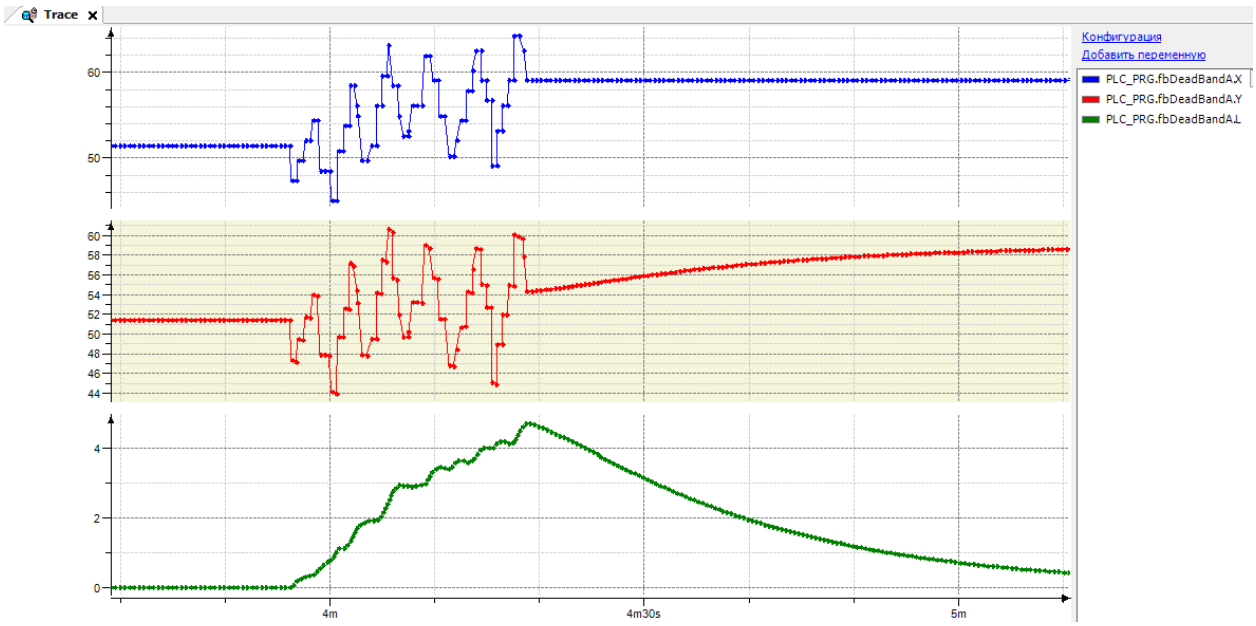


Рис. 23.18. Трассировка работы ФБ **DEAD_BAND_A** (T=T#5s, KL=2.0, LM=5.0)

23.12. DEAD_ZONE

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	X	REAL	Входное значение.
	L	REAL	Зона нечувствительности.
Выходы	DEAD_ZONE	REAL	Выходное значение.

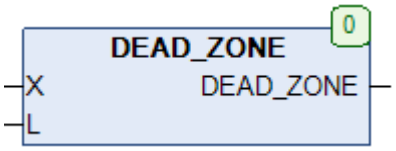


Рис. 23.19. Внешний вид функции **DEAD_ZONE** на языке CFC

Функция **DEAD_ZONE** представляет собой передаточную функцию с зоной нечувствительности. Значение входа **X** ограничивается по следующему принципу:

- DEAD_ZONE = X, если L < |X|;
- DEAD_ZONE = 0.0, если |X| ≤ L.

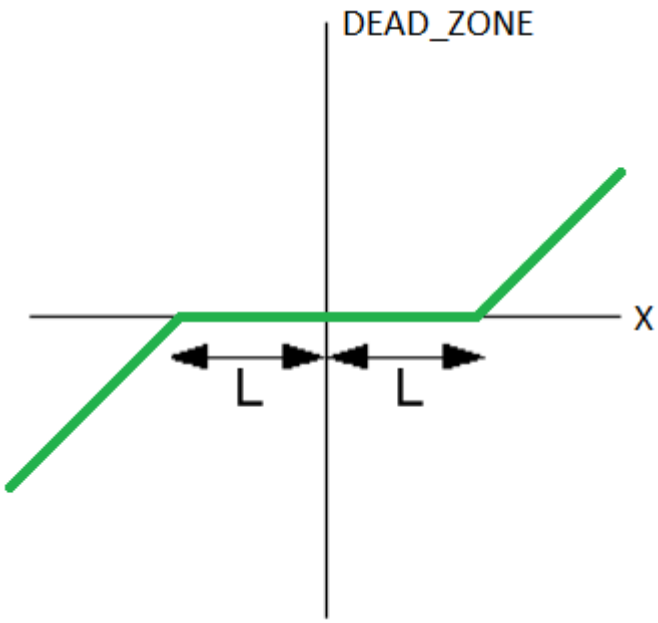


Рис. 23.20. Принцип работы функции **DEAD_ZONE**

23.13. DEAD_ZONE2

Тип модуля: ФБ	Переменная	Тип	Описание
Входы	X	REAL	Входное значение.
	L	REAL	Зона нечувствительности.
Выходы	Y	REAL	Выходное значение.

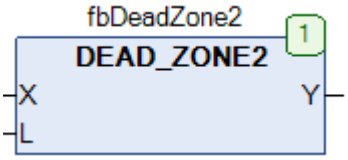


Рис. 23.21. Внешний вид ФБ **DEAD_ZONE2** на языке CFC

Функциональный блок **DEAD_ZONE2** представляет собой передаточную функцию с зоной нечувствительности и гистерезисом. Значение входа **X** ограничивается по следующему принципу:

- DEAD_ZONE = X, если L < |X|;
- DEAD_ZONE = L, если 0 ≤ |X| ≤ L;
- DEAD_ZONE = -L, если -L ≤ |X| ≤ 0.

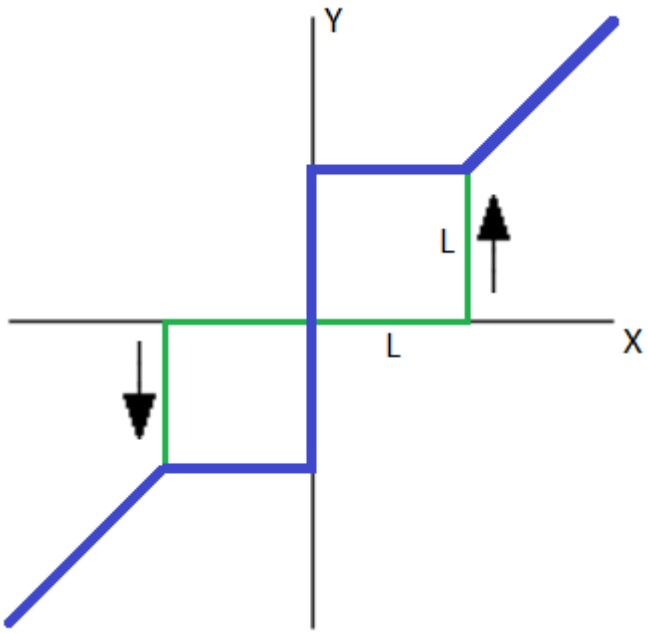


Рис. 23.22. Принцип работы ФБ **DEAD_ZONE2**

23.14. FT_DERIV

Тип модуля: ФБ	Переменная	Тип	Описание
Входы	in	REAL	Входное значение.
	K	REAL	Коэффициент передачи.
	run	BOOL	Сигнал управление блоком.
Выходы	out	REAL	Производная входного сигнала.
Используемые модули	T_PLC_US		

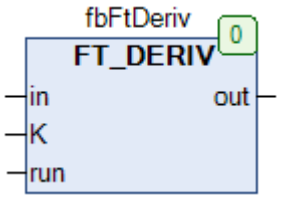


Рис. 23.23. Внешний вид ФБ **FT_DERIV** на языке CFC

Функциональный блок **FT_DERIV** представляет собой Д-звено. Если вход **run** имеет значение **TRUE**, то на выход **out** поступает производная входного сигнала **X** с коэффициентом усиления **K**.

$$\text{out} = \text{K} \cdot \frac{\Delta \text{in}}{\Delta \text{T}}, \quad \text{где } \Delta \text{T} - \text{время цикла ПЛК секундах}$$

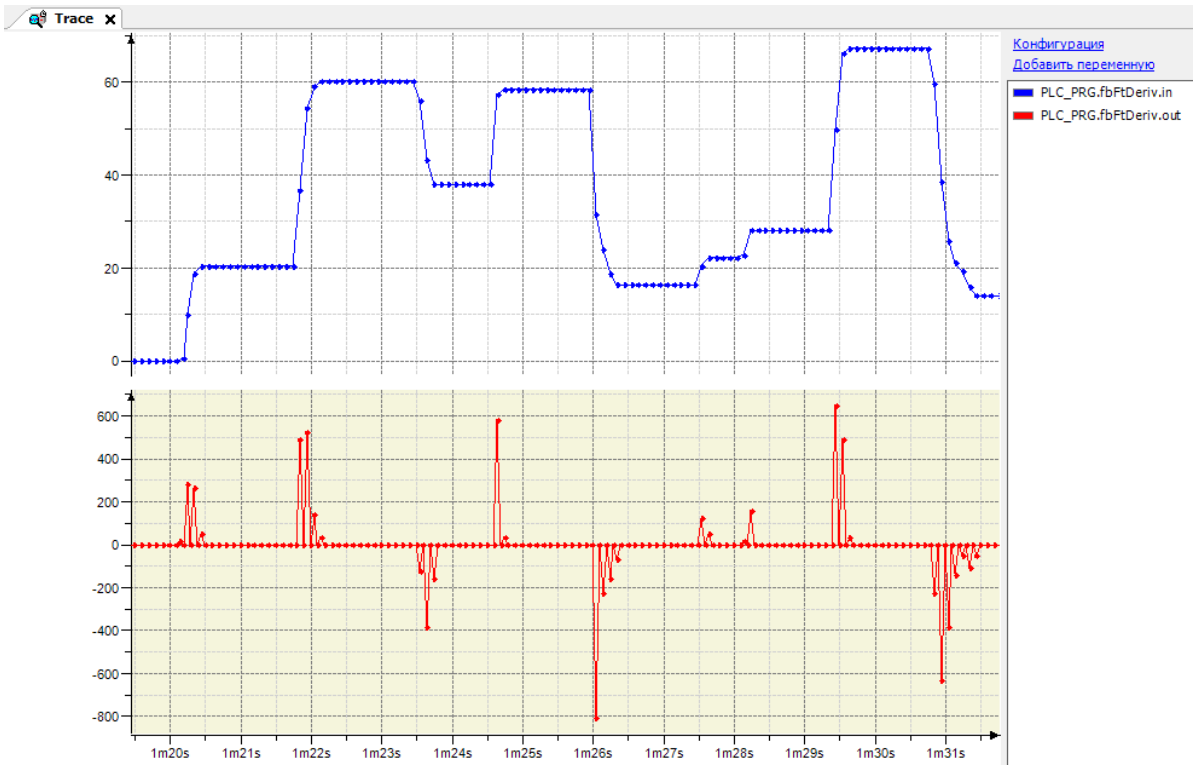


Рис. 23.24. Трассировка работы ФБ **FT_DERIV** (время цикла ПЛК = 50 мс, K=1.5)

23.15. FT_IMP

Тип модуля: ФБ	Переменная	Тип	Описание
Входы	in	REAL	Входное значение.
	T	TIME	Постоянная времени.
	K	REAL	Коэффициент усиления.
Выходы	out	REAL	Отфильтрованное значение.
Используемые модули	FT_PT1		

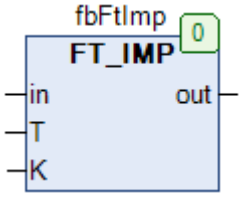


Рис. 23.25. Внешний вид ФБ **FT_IMP** на языке CFC

Функциональный блок **FT_IMP** представляет собой фильтр верхних частот с коэффициентом усиления **K** и постоянной времени **T**.

После изменения входного значения **IN**, выход **out** изменяется по следующему принципу:

- out(0)=K·IN;
- out(T)≈0.37· K·IN;
- out(3T)≈0.05· K·IN;
- out(…)=0.

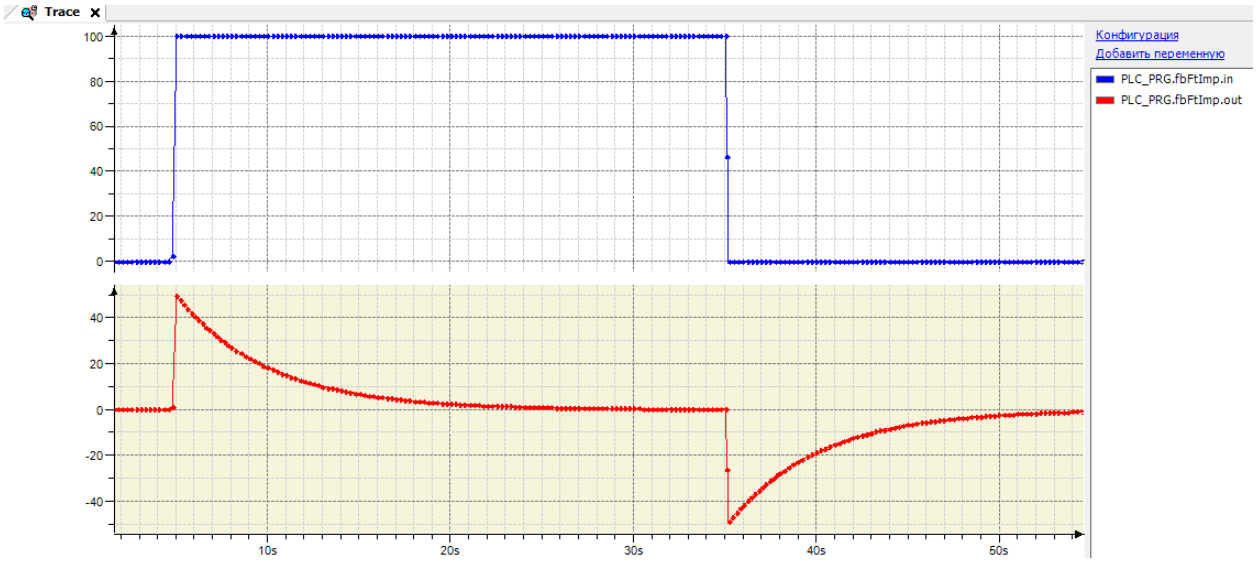


Рис. 23.26. Трассировка работы ФБ **FT_IMP** (K=0.5, T=T#5s)

23.16. FT_INT

Тип модуля: ФБ	Переменная	Тип	Описание
Входы	IN	REAL	Входное значение.
	K	REAL	Коэффициент передачи.
	RUN	BOOL	Сигнал управление блоком.
	RST	BOOL	Сигнал сброса блока.
	OUT_MIN	REAL	Нижний предел выходного сигнала.
	OUT_MAX	REAL	Верхний предел выходного сигнала.
Выходы	OUT	REAL	Значение интеграла.
	LIM	BOOL	Флаг «достигнут предел».
Используемые модули	INTEGRATE		

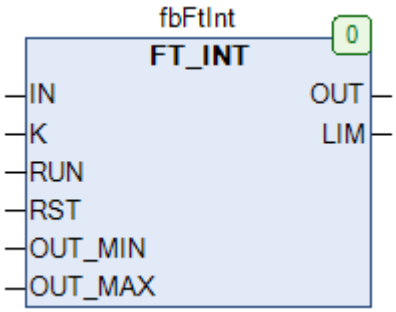


Рис. 23.27. Внешний вид ФБ **FT_INT** на языке CFC

Функциональный блок **FT_INT** представляет собой И-звено. Если вход **RUN** имеет значение **TRUE**, то на выход **out** поступает интеграл входного сигнала **IN** с коэффициентом усиления **K**. Для расчета интеграла используется метод трапеций:

$$\mathrm{OUT(n)} = \mathrm{OUT(n-1)} + \mathrm{K} \cdot \frac{\mathrm{IN(n)} + \mathrm{IN(n-1)}}{2} \cdot \Delta\mathrm{T},$$

где n – номер текущего цикла ПЛК, $\Delta\mathrm{T}$ – время цикла ПЛК секундах

Входы **OUT_MIN** и **OUT_MAX** определяют нижний и верхний пределы выхода **OUT**. При достижении любой из границ выход **LIM** принимает значение **TRUE**. По переднему фронту на входе **RST** происходит обнуление выходов блока.

Типичной проблемой интегратора является его точность. Точность переменных типа **REAL**, используемых в данном блоке, ограничена 7-8 знаками после запятой; в результате, если среднее значение входа **X** и его изменение отличаются более чем на 7 порядков (например, X=1000000, ΔX=0.0001), значение интеграла не будет меняться. См. блок FT_INT2, использующий числа двойной точности.

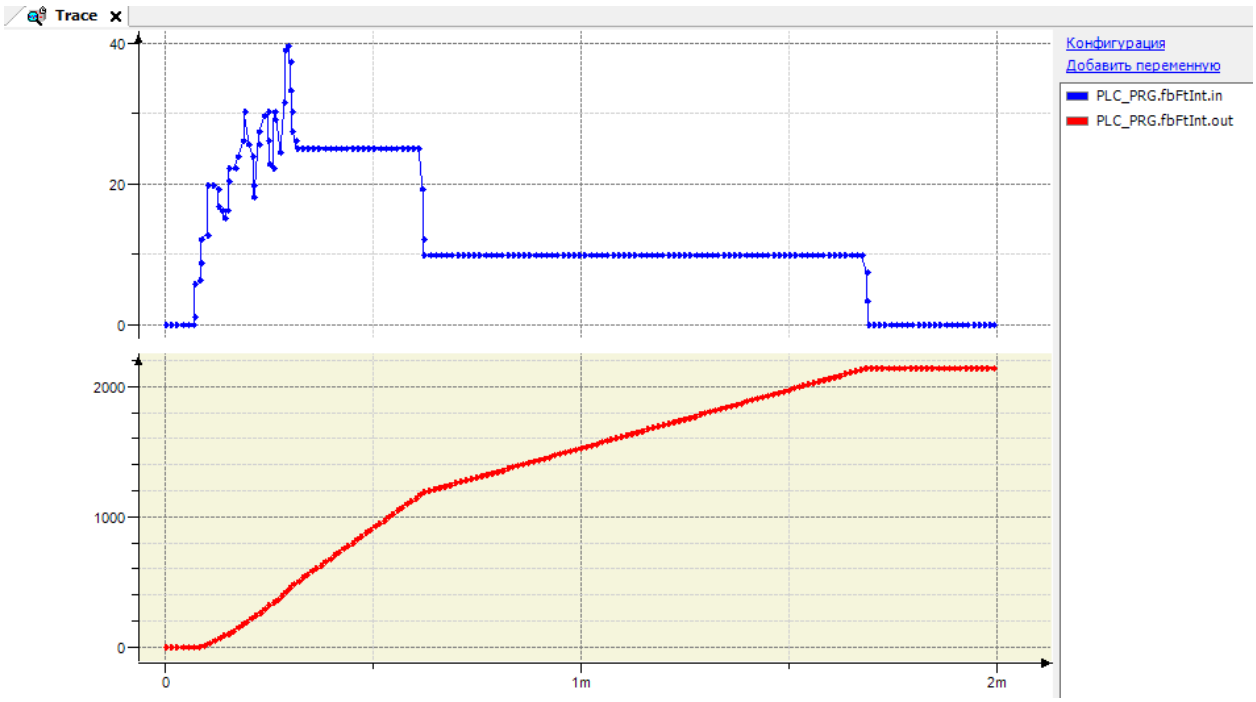


Рис. 23.28. Трассировка работы ФБ **FT_INT** (время цикла ПЛК = 50 мс, K=1.5)

23.17. FT_INT2

Тип модуля: ФБ	Переменная	Тип	Описание
Входы	IN	REAL	Входное значение.
	K	REAL	Коэффициент передачи.
	RUN	BOOL	Сигнал управление блоком.
	RST	BOOL	Сигнал сброса блока.
	OUT_MIN	REAL	Нижний предел выходного сигнала.
	OUT_MAX	REAL	Верхний предел выходного сигнала.
Выходы	OUT	REAL	Значение интеграла.
	LIM	BOOL	Флаг «достигнут предел».
Используемые модули	INTEGRATE, R2_SET, R2_ADD		

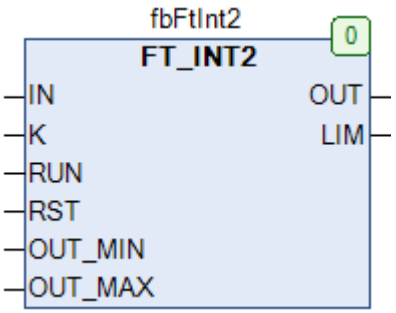


Рис. 23.29. Внешний вид ФБ **FT_INT2** на языке CFC

Функциональный блок **FT_INT2** представляет собой И-звено. Принцип работы блока полностью соответствует блоку FT_INT, единственным отличием является использование чисел двойной точности для промежуточных вычислений (более подробно см. в описании блока FT_INT).

23.18. FT_PD

Тип модуля: ФБ	**Переменная**	**Тип**	**Описание**
Входы	IN	REAL	Сигнал рассогласования.
	KP	REAL	Коэффициент передачи.
	TV	REAL	Постоянная времени дифференцирования.
Выходы	Y	REAL	Величина управляющего воздействия.
Используемые модули	FT_DERIV		

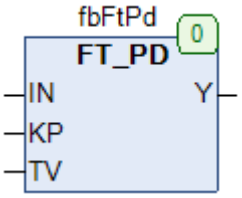


Рис. 23.30. Внешний вид ФБ **FT_PD** на языке CFC

Функциональный блок **FT_PD** представляет собой блок ПД-регулирования. Значение управляющего воздействия **Y** вычисляется по формуле:

$$Y = KP \cdot (IN + FT\_DERIV.OUT(IN, TV)),$$

Блок может использоваться совместно с модулями CTRL_IN и CTRL_OUT для реализации ПД-регулятора.

23.19. FT_PDT1

Тип модуля: ФБ	**Переменная**	**Тип**	**Описание**
Входы	IN	REAL	Сигнал рассогласования.
	KP	REAL	Коэффициент передачи.
	TV	REAL	Постоянная времени дифференцирования.
	T1	REAL	Запаздывание.
Выходы	Y	REAL	Величина управляющего воздействия.
Используемые модули	FT_DERIV, FT_PT1		

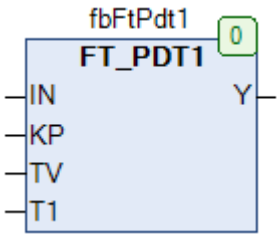


Рис. 23.31. Внешний вид ФБ **FT_PDT1** на языке CFC

Функциональный блок **FT_PDT1** представляет собой блок ПД-регулирования с апериодическим звеном 1-го порядка. Значение управляющего воздействия **Y** вычисляется по формуле:

$$Y = KP \cdot (IN + FT\_PT1.OUT(FT\_DERIV.OUT(IN, TV), T1))),$$

Блок может использоваться совместно с модулями CTRL_IN и CTRL_OUT для реализации ПД-регулятора.

23.20. FT_PI

Тип модуля: ФБ	Переменная	Тип	Описание
Входы	IN	REAL	Сигнал рассогласования.
	KP	REAL	Коэффициент усиления П-составляющей.
	KI	REAL	Коэффициент усиления И-составляющей.
	ILIM_L	REAL	Нижний предел зоны накопления интеграла.
	ILIM_H	REAL	Верхний предел зоны накопления интеграла.
	IEN	BOOL	Сигнал управления интегратором.
	RST	BOOL	Сигнал обнуления И-составляющей.
Выходы	Y	REAL	Величина управляющего воздействия.
	LIM	BOOL	Флаг «достигнут предел».
Используемые модули	FT_INT		

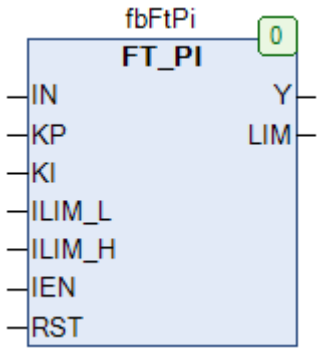


Рис. 23.32. Внешний вид ФБ **FT_PI** на языке CFC

Функциональный блок **FT_PI** представляет собой блок ПИ-регулирования. Значение управляющего воздействия **Y** вычисляется по формуле:

$$Y = KP \cdot IN + KI \cdot \sum_{i=0}^{n} IN_i \cdot \Delta t_{изм}, \qquad где$$

$$\Delta t_{изм} - время\ между\ двумя\ измерениями$$

На вход **IN** подается сигнал рассогласования. Входы **KP** и **KI** определяют коэффициенты усиления пропорциональной (П) и интегральной (И) составляющей регулятора. Входы **ILIM_L** и **ILIM_H** определяют нижний и верхний пределы зоны накопления интеграла. При достижении любой из границ выход **LIM** принимает значение **TRUE**, и значение интегратора перестает увеличиваться. Кроме того, интегратор в любой момент может быть отключен с помощью присвоения значения **FALSE** входу **IEN**. По переднему фронту на входе **RST** происходит обнуление накопленной интегральной составляющей блока.

23.21. FT_PID

Тип модуля: ФБ	Переменная	Тип	Описание
Входы	IN	REAL	Сигнал рассогласования.
	KP	REAL	Коэффициент усиления П-составляющей.
	TN	REAL	Постоянная времени интегрирования.
	TV	REAL	Постоянная времени дифференцирования.
	ILIM_L	REAL	Нижний предел зоны накопления интеграла.
	ILIM_H	REAL	Верхний предел зоны накопления интеграла.
	IEN	BOOL	Сигнал управления интегратором.
	RST	BOOL	Сигнал обнуления И-составляющей.
Выходы	Y	REAL	Величина управляющего воздействия.
	LIM	BOOL	Флаг «достигнут предел».
Используемые модули	FT_INT, FT_DERIV		

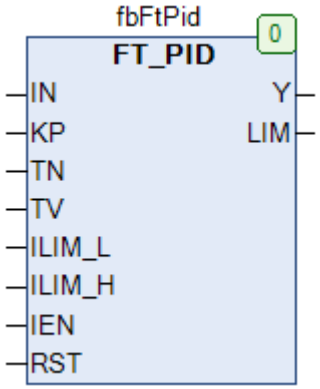


Рис. 23.33. Внешний вид ФБ **FT_PID** на языке CFC

Функциональный блок **FT_PID** представляет собой блок ПИД-регулирования. Значение управляющего воздействия **Y** вычисляется по формуле:

$$Y = KP \cdot IN + \frac{1}{TN} \cdot \sum_{i=0}^{n} IN_i \cdot \Delta t_{изм} + TV \cdot \frac{\Delta IN_i}{\Delta t_{изм}}, \qquad \text{где}$$

$$\Delta t_{изм} - \text{время между двумя измерениями}$$

На вход **IN** подается сигнал рассогласования. Вход **KP** и определяет коэффициент усиления пропорциональной (П) составляющей регулятора, входы **TN** и **TV** – постоянные времени интегрирования и дифференцирования (***обратите внимание***, они не могут быть равны **0** – это приведет к исключению в работе ПЛК). Входы **ILIM_L** и **ILIM_H** определяют нижний и верхний пределы зоны накопления интеграла. При достижении любой из границ выход **LIM** принимает значение **TRUE**, и значение интегратора перестает увеличиваться. Кроме того, интегратор в любой момент может быть отключен с помощью присвоения значения **FALSE** входу **IEN**. По переднему фронту на входе **RST** происходит обнуление накопленной интегральной составляющей блока.

23.22. FT_PIDW

Тип модуля: ФБ	Переменная	Тип	Описание
Входы	IN	REAL	Сигнал рассогласования.
	KP	REAL	Коэффициент усиления П-составляющей.
	TN	REAL	Постоянная времени интегрирования.
	TV	REAL	Постоянная времени дифференцирования.
	LIM_L	REAL	Нижний предел управляющего сигнала.
	LIM_H	REAL	Верхний предел управляющего сигнала.
	RST	BOOL	Сигнал обнуления И-составляющей.
Выходы	Y	REAL	Величина управляющего воздействия.
	LIM	BOOL	Флаг «достигнут предел».
Используемые модули	INTEGRATE, FT_DERIV		

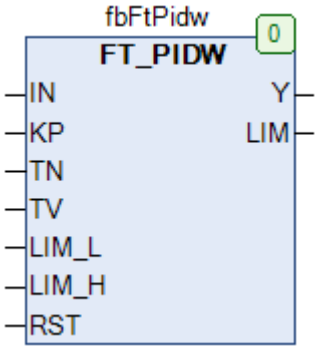


Рис. 23.34. Внешний вид ФБ **FT_PIDW** на языке CFC

Функциональный блок **FT_PIDW** представляет собой блок ПИД-регулирования с компенсацией эффекта интегрального насыщения. Значение управляющего воздействия **Y** вычисляется по формуле:

$$Y = KP \cdot IN + \frac{1}{TN} \cdot \sum_{i=0}^{n} IN_i \cdot \Delta t_{изм} + TV \cdot \frac{\Delta IN_i}{\Delta t_{изм}}, \qquad \text{где}$$

$$\Delta t_{изм} - \text{время между двумя измерениями}$$

На вход **IN** подается сигнал рассогласования. Вход **KP** и определяет коэффициент усиления пропорциональной (П) составляющей регулятора, входы **TN** и **TV** – постоянные времени интегрирования и дифференцирования (***обратите внимание***, они не могут быть равны **0** – это приведет к исключению в работе ПЛК). Входы **LIM_L** и **LIM_H** определяют нижний и верхний пределы управляющего сигнала. При достижении любой из границ выход **LIM** принимает значение **TRUE**, и значение выхода **Y** фиксируется. По переднему фронту на входе **RST** происходит обнуление накопленной интегральной составляющей блока.

23.23. FT_PIDWL

Тип модуля: ФБ	Переменная	Тип	Описание
Входы	IN	REAL	Сигнал рассогласования.
	KP	REAL	Коэффициент усиления П-составляющей.
	TN	REAL	Постоянная времени интегрирования.
	TV	REAL	Постоянная времени дифференцирования.
	LIM_L	REAL	Нижний предел управляющего сигнала.
	LIM_H	REAL	Верхний предел управляющего сигнала.
	RST	BOOL	Сигнал обнуления И-составляющей.
Выходы	Y	REAL	Величина управляющего воздействия.
	LIM	BOOL	Флаг «достигнут предел».
Используемые модули	FT_PIWL, FT_DERIV		

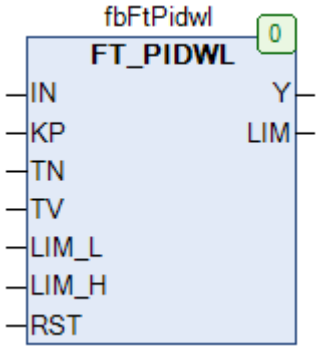


Рис. 23.35. Внешний вид ФБ **FT_PIDWL** на языке CFC

Функциональный блок **FT_PIDWL** представляет собой блок ПИД-регулирования с компенсацией эффекта интегрального насыщения. Значение управляющего воздействия **Y** вычисляется по формуле:

$$Y = KP \cdot IN + \frac{1}{TN} \cdot \sum_{i=0}^{n} IN_i \cdot \Delta t_{изм} + TV \cdot \frac{\Delta IN_i}{\Delta t_{изм}}, \qquad где$$

$$\Delta t_{изм} - время\ между\ двумя\ измерениями$$

Принцип работы блока полностью соответствует блоку FT_PIDW. Единственным отличием является набор модулей, использованных для реализации блока.

23.24. FT_PIW

Тип модуля: ФБ	Переменная	Тип	Описание
Входы	IN	REAL	Сигнал рассогласования.
	KP	REAL	Коэффициент усиления П-составляющей.
	KI	REAL	Коэффициент усиления И-составляющей.
	LIM_L	REAL	Нижний предел управляющего сигнала.
	LIM_H	REAL	Верхний предел управляющего сигнала.
	RST	BOOL	Сигнал обнуления И-составляющей.
Выходы	Y	REAL	Величина управляющего воздействия.
	LIM	BOOL	Флаг «достигнут предел».
Используемые модули	FT_INT		

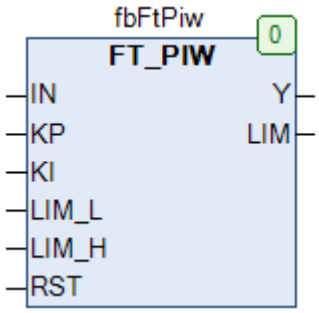


Рис. 23.36. Внешний вид ФБ **FT_PIW** на языке CFC

Функциональный блок **FT_PIW** представляет собой блок ПИ-регулирования с компенсацией эффекта интегрального насыщения. Значение управляющего воздействия **Y** вычисляется по формуле:

$$Y = KP \cdot IN + KI \cdot \sum_{i=0}^{n} IN_i \cdot \Delta t_{изм}, \qquad \text{где}$$

$$\Delta t_{изм} - \text{время между двумя измерениями}$$

На вход **IN** подается сигнал рассогласования. Входы **KP** и **KI** определяют коэффициенты усиления пропорциональной (П) и интегральной (И) составляющей регулятора. Входы **LIM_L** и **LIM_H** определяют нижний и верхний пределы управляющего сигнала. При достижении любой из границ выход **LIM** принимает значение **TRUE**, и значение выхода **Y** фиксируется. По переднему фронту на входе **RST** происходит обнуление накопленной интегральной составляющей блока.

23.25. FT_PIWL

Тип модуля: ФБ	Переменная	Тип	Описание
Входы	IN	REAL	Сигнал рассогласования.
	KP	REAL	Коэффициент усиления П-составляющей.
	KI	REAL	Коэффициент усиления И-составляющей.
	LIM_L	REAL	Нижний предел управляющего сигнала.
	LIM_H	REAL	Верхний предел управляющего сигнала.
	RST	BOOL	Сигнал обнуления И-составляющей.
Выходы	Y	REAL	Величина управляющего воздействия.
	LIM	BOOL	Флаг «достигнут предел».
Используемые модули	T_PLC_US		

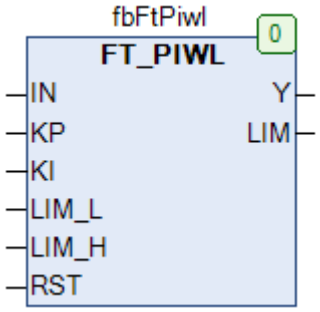


Рис. 23.36. Внешний вид ФБ **FT_PIWL** на языке CFC

Функциональный блок **FT_PIWL** представляет собой блок ПИ-регулирования с компенсацией эффекта интегрального насыщения. Значение управляющего воздействия **Y** вычисляется по формуле:

$$Y = KP \cdot IN + KI \cdot \sum_{i=0}^{n} IN_i \cdot \Delta t_{изм}, \qquad \text{где}$$

$$\Delta t_{изм} - \text{время между двумя измерениями}$$

Принцип работы блока полностью соответствует блоку FT_PIW. Отличием является внутренняя реализация блока, в которой не используются другие модули регуляторов.

23.26. FT_PT1

Тип модуля: ФБ	Переменная	Тип	Описание
Входы	IN	REAL	Входное значение.
	T	TIME	Постоянная времени.
	K	REAL	Коэффициент усиления.
Выходы	out	REAL	Выходное значение.
Используемые модули	T_PLC_US		

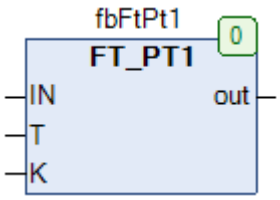


Рис. 23.37. Внешний вид ФБ **FT_PT1** на языке CFC

Функциональный блок **FT_PT1** представляет собой апериодическое звено 1-го порядка с коэффициентом усиления **K** и постоянной времени **T**.

После изменения входного значения **IN**, выход **out** изменяется по следующему принципу:

- out(0)=0;
- out(T)≈0.63· K·IN;
- out(3T)≈0.95· K·IN;
- out(…)=K·IN.

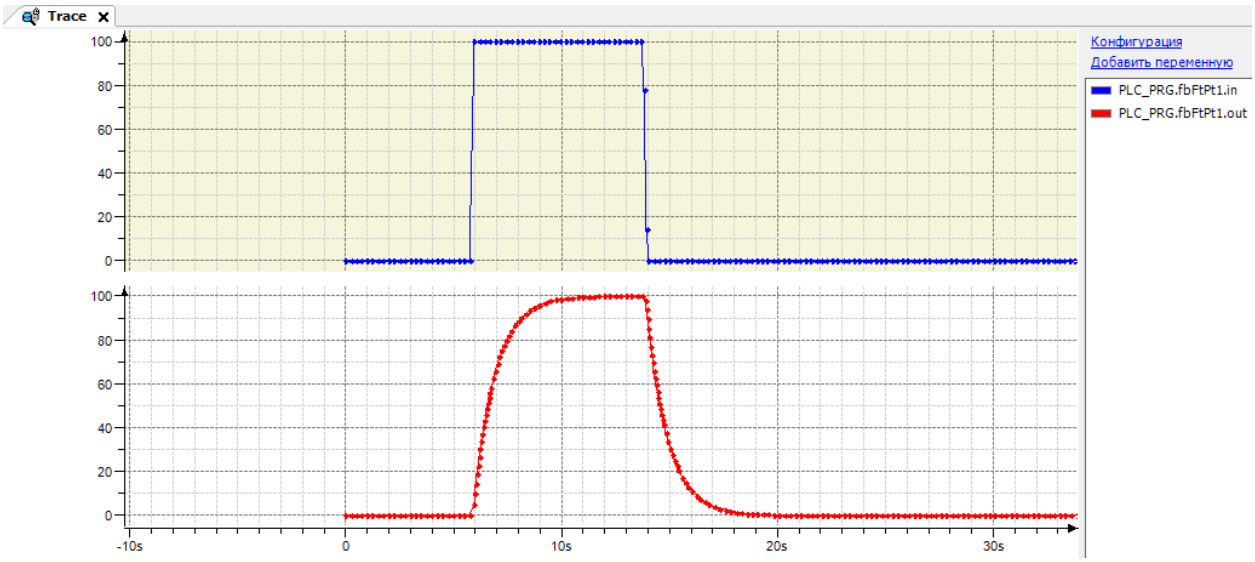


Рис. 23.38. Трассировка работы ФБ **FT_PT1** (K=1.0, T=T#1s)

23.27. FT_PT2

Тип модуля: ФБ	Переменная	Тип	Описание
Входы	IN	REAL	Входное значение.
	T	TIME	Постоянная времени.
	D	REAL	Коэффициент затухания.
	K	REAL	Коэффициент усиления.
Выходы	out	REAL	Выходное значение.
Используемые модули	INTEGRATE		

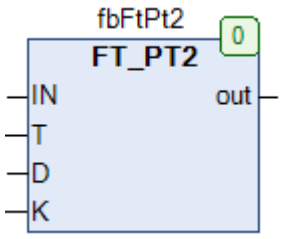


Рис. 23.39. Внешний вид ФБ **FT_PT2** на языке CFC

Функциональный блок **FT_PT2** представляет собой апериодическое звено 2-го порядка с коэффициентом усиления **K**, коэффициентом затухания **D** и постоянной времени **T**.

После изменения входного значения **IN**, выход **out** изменяется по следующему принципу:

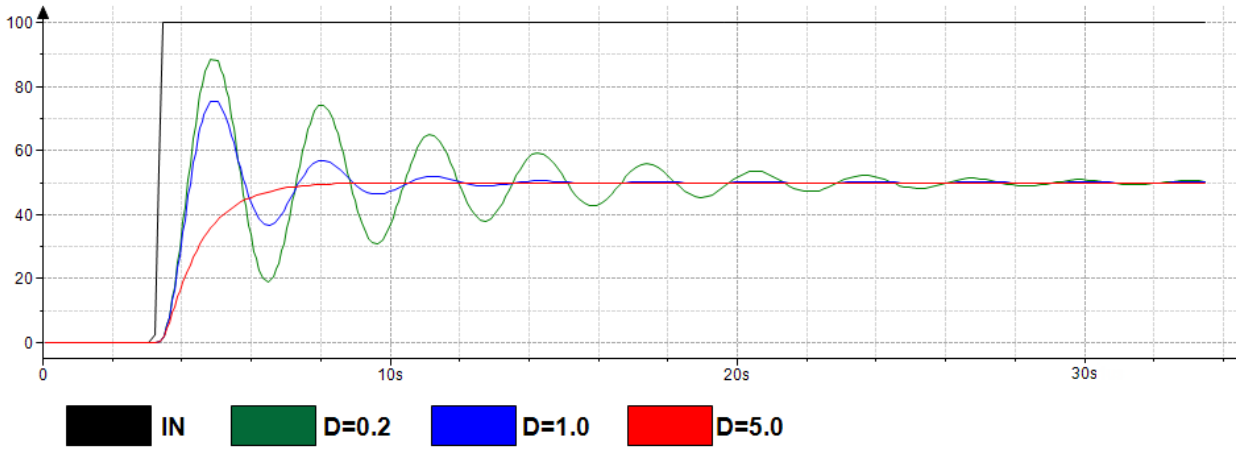


Рис. 23.40. Трассировка работы ФБ **FT_PT2** (K=0.5, T=T#500ms)

23.28. FT_TN16

Тип модуля: ФБ	Переменная	Тип	Описание
Входы	IN	REAL	Входное значение.
	T	TIME	Время задержки.
Выходы	out	REAL	Выходное значение.
	trig	BOOL	Флаг «сохранение значения».
Используемые модули	T_PLC_MS		

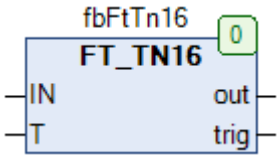


Рис. 23.41. Внешний вид ФБ **FT_TN16** на языке CFC

Функциональный блок **FT_TN16** представляет собой модуль задержки сигнала. Входное значение **IN** сохраняется с интервалом **T/16**, при каждом сохранении выход **out** принимает значение **TRUE** на один цикл ПЛК. Блок сохраняет 16 значений переменной **IN**. Эти значения подается на выход **out** спустя время задержки **T**. См. также блоки FT_TN64 и FT_TN8, которые отличаются частотой дискретизации входного значения.

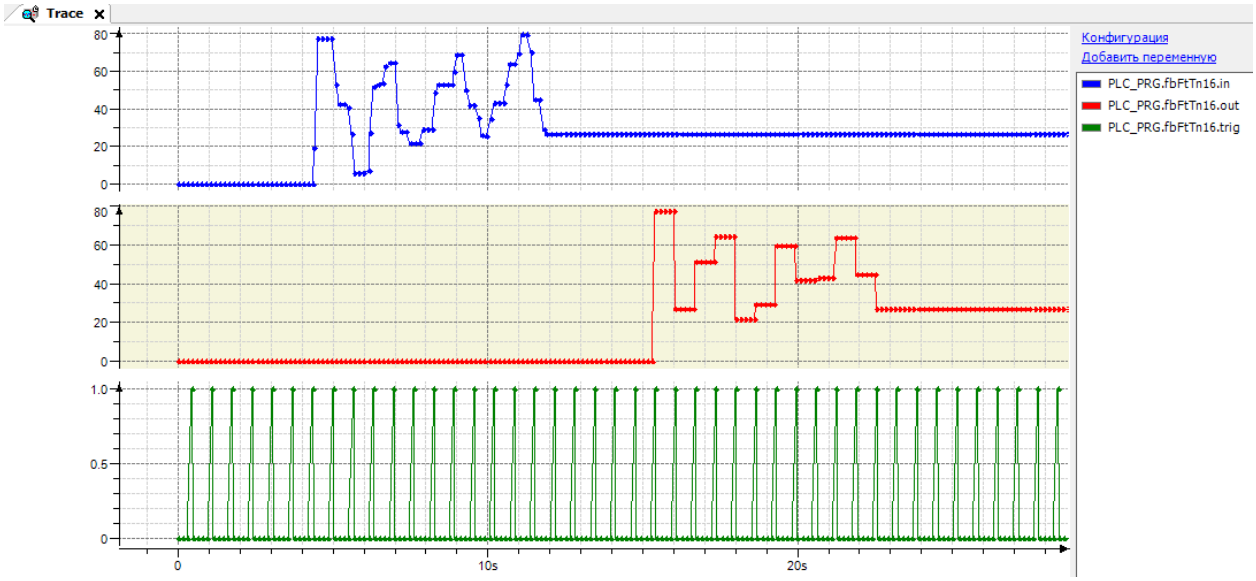


Рис. 23.42. Трассировка работы ФБ **FT_TN16** (T=T#10s)

23.29. FT_TN64

Тип модуля: ФБ	Переменная	Тип	Описание
Входы	IN	REAL	Входное значение.
	T	TIME	Время задержки.
Выходы	out	REAL	Выходное значение.
	trig	BOOL	Флаг «сохранение значения».
Используемые модули	T_PLC_MS		

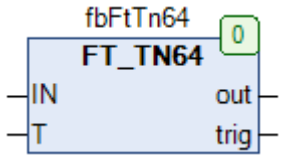


Рис. 23.43. Внешний вид ФБ **FT_TN64** на языке CFC

Функциональный блок **FT_TN64** представляет собой модуль задержки сигнала. Входное значение **IN** сохраняется с интервалом **T/64**, при каждом сохранении выход **out** принимает значение **TRUE** на один цикл ПЛК. Блок сохраняет 64 значений переменной **IN**. Эти значения подается на выход out спустя время задержки **T** (см. рис. 23.42). См. также блоки FT_TN16 и FT_TN8, которые отличаются частотой дискретизации входного значения.

23.30. FT_TN8

Тип модуля: ФБ	Переменная	Тип	Описание
Входы	IN	REAL	Входное значение.
	T	TIME	Время задержки.
Выходы	out	REAL	Выходное значение.
	trig	BOOL	Флаг «сохранение значения».
Используемые модули	T_PLC_MS		

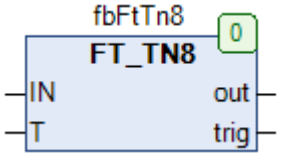


Рис. 23.44. Внешний вид ФБ **FT_TN8** на языке CFC

Функциональный блок **FT_TN8** представляет собой модуль задержки сигнала. Входное значение **IN** сохраняется с интервалом **T/8**, при каждом сохранении выход **out** принимает значение **TRUE** на один цикл ПЛК. Блок сохраняет 8 значений переменной **IN**. Эти значения подается на выход **out** спустя время задержки **T** (см. рис. 23.42). См. также блоки FT_TN16 и FT_TN64, которые отличаются частотой дискретизации входного значения.

23.31. HYST

Тип модуля: ФБ	Переменная	Тип	Описание
Входы	IN	REAL	Входное значение.
	ON	REAL	Порог включения ИМ.
	OFF	REAL	Порог отключения ИМ.
Выходы	Q	BOOL	Сигнал управления.
	win	BOOL	Флаг «значение в допустимых границах».

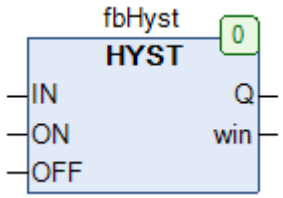


Рис. 23.45. Внешний вид ФБ **HYST** на языке CFC

Функциональный блок **HYST** представляет собой двухпозиционный регулятор. На вход **IN** подается измеренное значение, входы **ON** и **OFF** определяют порог включения и отключения исполнительного механизма (ИМ). Значение выхода **Q** определяется по следующему принципу:

1. ON ≥ OFF

- Q=FALSE, если IN < OFF;
- Q=TRUE, если IN > ON.

2. ON < OFF

- Q=FALSE, если IN > OFF;
- Q=TRUE, если IN < ON.

Выход **WIN** принимает значение **TRUE**, если **IN** принадлежит интервалу [**OFF, ON**].

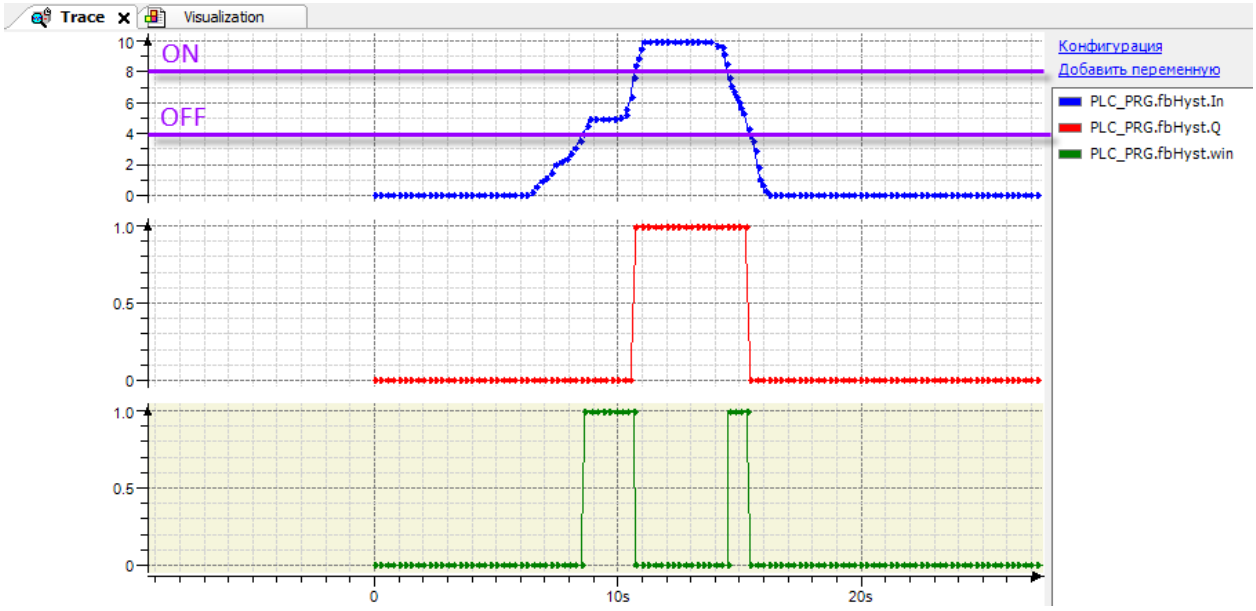


Рис. 23.46. Трассировка работы ФБ **HYST** (ON=8.0, OFF=4.0)

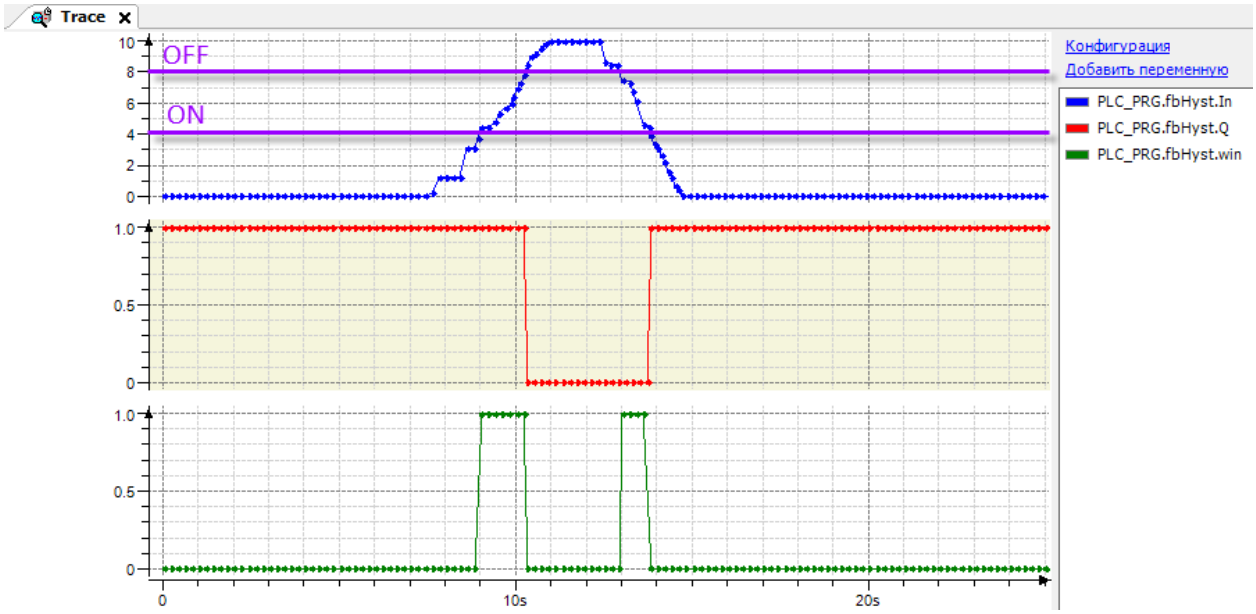


Рис. 23.47. Трассировка работы ФБ **HYST** (ON=4.0, OFF=8.0)

23.32. HYST_1

Тип модуля: ФБ	Переменная	Тип	Описание
Входы	IN	REAL	Входное значение.
	high	REAL	Порог включения ИМ.
	low	REAL	Порог отключения ИМ.
Выходы	Q	BOOL	Сигнал управления.
	win	BOOL	Флаг «значение в допустимых границах».

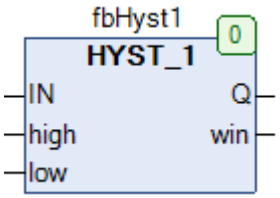


Рис. 23.48. Внешний вид ФБ **HYST_1** на языке CFC

Функциональный блок **HYST_1** представляет собой двухпозиционный регулятор. На вход **IN** подается измеренное значение, входы **high** и **low** определяют порог включения и отключения исполнительного механизма (при этом значение **high** должно быть больше значения **low**). Значение выхода **Q** определяется по следующему принципу:

- Q=FALSE, если IN < low;
- Q=TRUE, если IN > high.
- Q сохраняет предыдущее значение, если low ≤ IN ≤ high.

Выход **WIN** принимает значение **TRUE**, если **IN** принадлежит интервалу [**OFF, ON**].

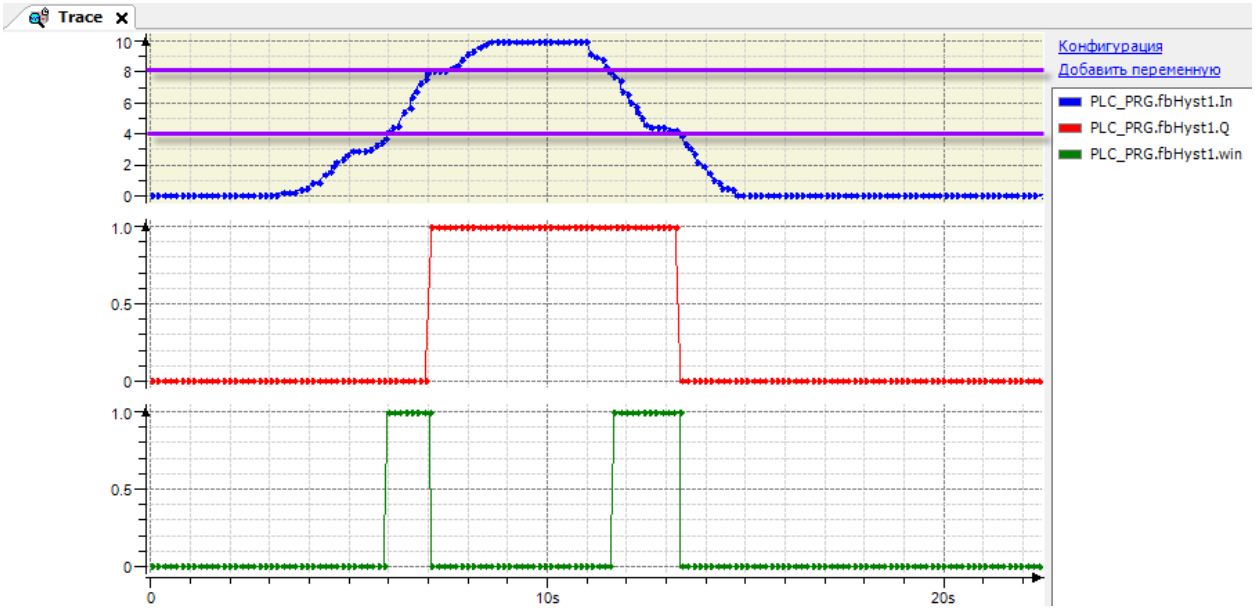


Рис. 23.49. Трассировка работы ФБ **HYST_1** (high=8.0, low=4.0)

23.33. HYST_2

Тип модуля: ФБ	Переменная	Тип	Описание
Входы	IN	REAL	Входное значение.
	VAL	REAL	Граница гистерезиса.
	HYS	REAL	Ширина гистерезиса.
Выходы	Q	BOOL	Сигнал управления.
	win	BOOL	Флаг «значение в допустимых границах».

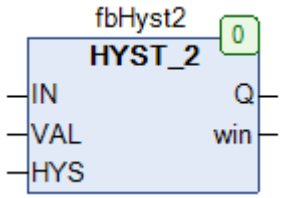


Рис. 23.50. Внешний вид ФБ **HYST_2** на языке CFC

Функциональный блок **HYST_2** представляет собой двухпозиционный регулятор с зоной гистерезиса. На вход **IN** подается измеренное значение, входы **VAL** и **HYS** определяют границу гистерезиса и его ширину. Значение выхода **Q** определяется по следующему принципу:

- Q=FALSE, если IN < (VAL – 0.5·HYS);
- Q=TRUE, если IN > (VAL + 0.5·HYS);
- Q сохраняет предыдущее значение, если (VAL – 0.5·HYS) ≤ IN ≤ (VAL + 0.5·HYS).

Выход **WIN** принимает значение **TRUE**, если **IN** принадлежит интервалу [**VAL – 0.5·HYS, VAL + 0.5·HYS**].

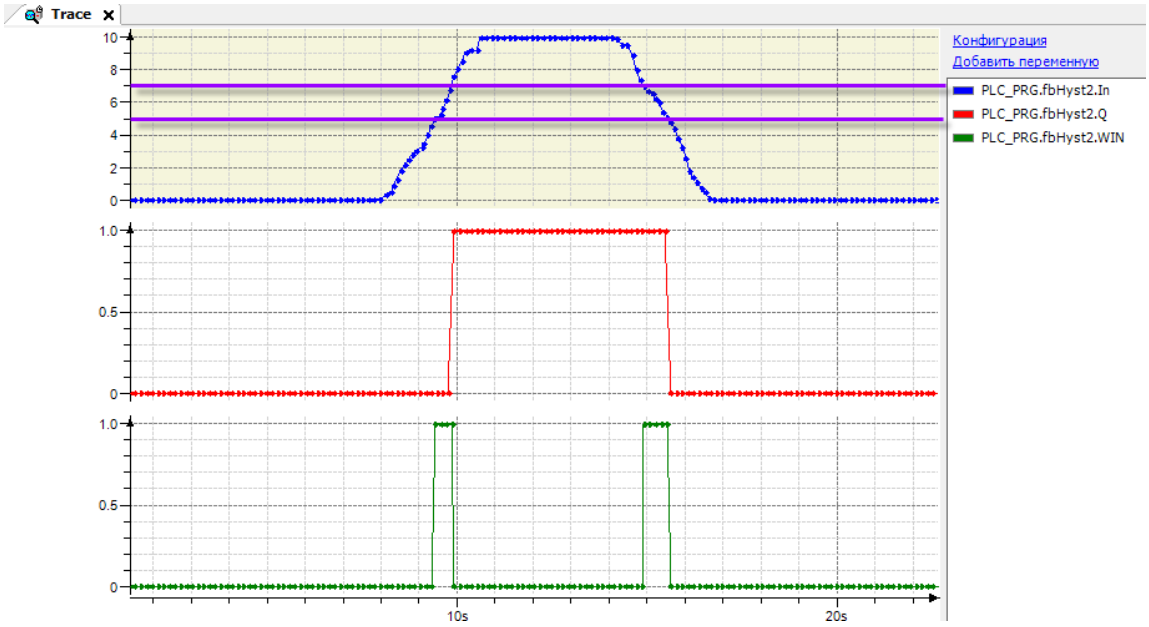


Рис. 23.51. Трассировка работы ФБ **HYST_2** (VAL=6.0, HYS=2.0)

23.34. HYST_3

Тип модуля: ФБ	Переменная	Тип	Описание
Входы	IN	REAL	Входное значение.
	hyst	REAL	Ширина гистерезиса.
	val1	REAL	Граница гистерезиса 1.
	val2	REAL	Граница гистерезиса 2.
Выходы	Q1	BOOL	Сигнал управления «ход вправо».
	Q2	BOOL	Сигнал управления «ход влево».

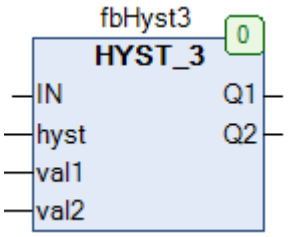


Рис. 23.52. Внешний вид ФБ **HYST_3** на языке CFC

Функциональный блок **HYST_3** представляет собой трехпозиционный регулятор с двумя зонами гистерезиса. На вход **IN** подается измеренное значение, входы **val1**, **val2** и **hyst** определяют границы гистерезиса и его ширину. Значение выходов **Q1** и **Q2** определяется по следующему принципу:

- Q1=TRUE, если IN < (val1 – 0.5·hyst);
- Q1=FALSE, если IN > (val1 + 0.5·hyst);
- Q2=FALSE, если IN < (val2 – 0.5·hyst);
- Q2=TRUE, если IN > (val2 + 0.5·hyst).

Пояснить принцип работы блока можно на следующем примере. Пусть **IN** – измеренное значение температуры, **U1** и **U2** – нижняя и верхняя уставки температуры, **Q1** и **Q2** – реле управления задвижкой (**Q1** – открытие, **Q2** – закрытие). Значения **U1** и **U2** определяются следующим образом:

- U1 = (val1 – 0.5·hyst);
- U2 = (val2 + 0.5·hyst).

Если текущее значение температуры **IN** меньше уставки **U1**, то замыкается реле **Q1**, заставляя задвижку открываться и увеличивать количество теплоносителя. Как только значение температуры превысит (**U1+hyst**), реле **Q1** разомкнется и задвижка останется в том промежуточном положении, которое успело достигнуть за время **t1**. Если затем значение температуры по какой-либо причине превысит уставку **U2**, сработает реле **U2** и задвижка начнет закрываться до тех пор, пока температура не вернется в диапазон межу значениями [**U1+hyst, U2-hyst**].

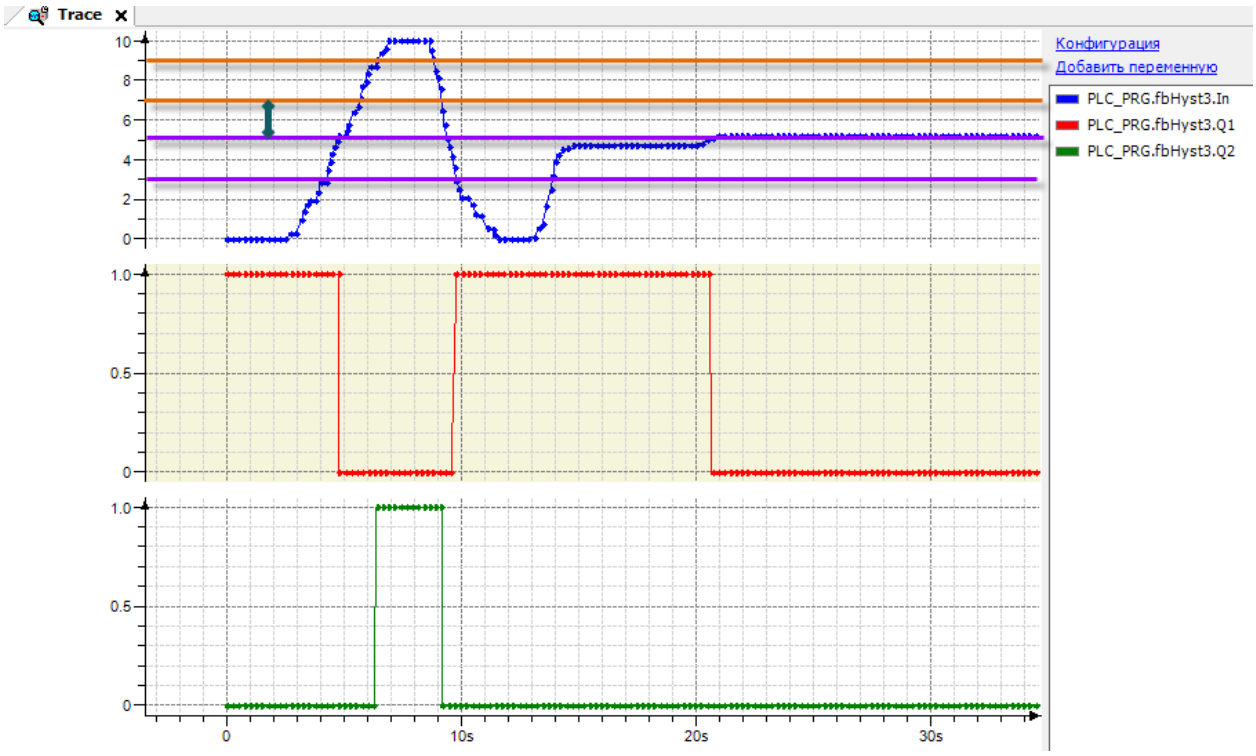


Рис. 23.53. Трассировка работы ФБ **HYST_3** (val1=4.0, val2=6.0, hyst=2.0)

23.35. INTEGRATE

Тип модуля: ФБ	Переменная	Тип	Описание
Входы	E	BOOL	Сигнал управление блоком.
	X	REAL	Входное значение.
	K	REAL	Коэффициент передачи.
Входы-выходы	Y	REAL	Значение интеграла.
Используемые модули	T_PLC_MS		

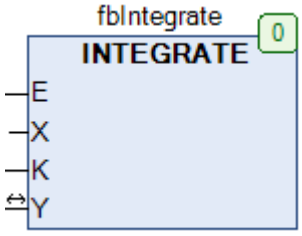


Рис. 23.54. Внешний вид ФБ **INTEGRATE** на языке CFC

Функциональный блок **INTEGRATE** представляет собой И-звено. Если вход **E** имеет значение **TRUE**, то на выход **Y** поступает интеграл входного сигнала **X** с коэффициентом усиления **K**. Для расчета интеграла используется метод трапеций:

$$\mathrm{OUT(n)} = \mathrm{OUT(n-1)} + \mathrm{K} \cdot \frac{\mathrm{IN(n)} + \mathrm{IN(n-1)}}{2} \cdot \Delta \mathrm{T},$$

где n – номер текущего цикла ПЛК, $\Delta \mathrm{T}$ – время цикла ПЛК секундах

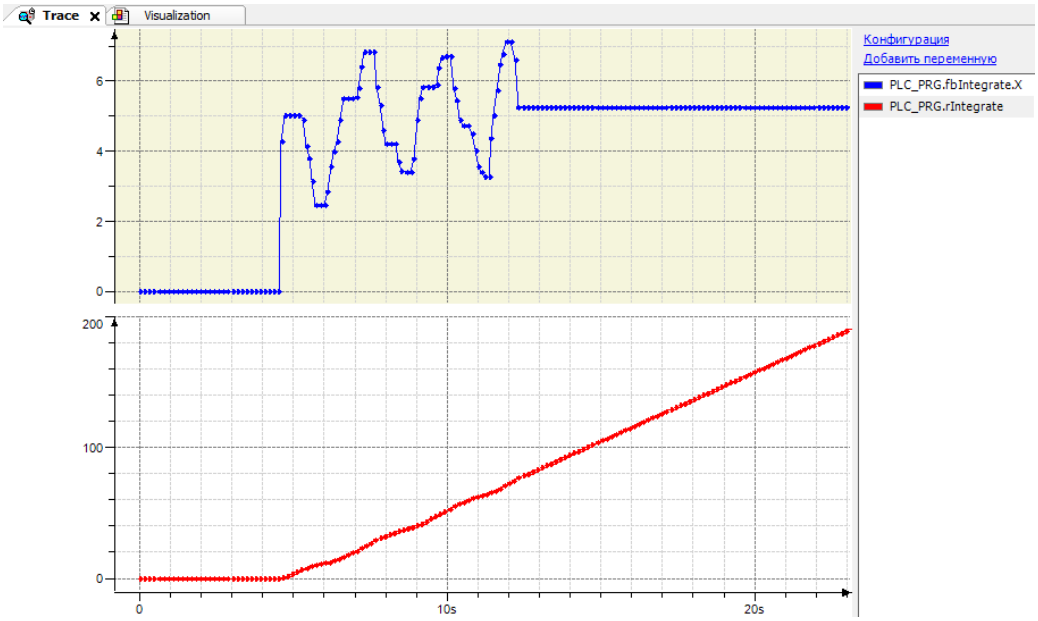


Рис. 23.55. Трассировка работы ФБ **INTEGRATE** (K=2.0)

23.36. Комментарий об отсутствующих пунктах

В англо- и германоязычной документации на библиотеку **OSCAT Basic** версии **3.33** в данной главе присутствуют пункты **23.36 – 23.57.** Фактически же модули, описываемые в данных пунктах, не входят в эту версию библиотеки; они были вынесены в библиотеку **OSCAT Building**. В связи с этим, их описание будет приведено в документации на соответствующую библиотеку.

24. Модули управления

24.1. DRIVER_1

Тип модуля: ФБ	**Переменная**	**Тип**	**Описание**
Входы	SET	BOOL	Ручное управление выходом.
	IN	BOOL	Сигнал нажатия на кнопку.
	RST	BOOL	Переключение выхода в FALSE.
Выходы	Q	BOOL	Выход кнопки.
Параметры	Toggle_Mode	BOOL	Режим работы кнопки.
	Timeout	TIME	Таймаут сброса выхода.

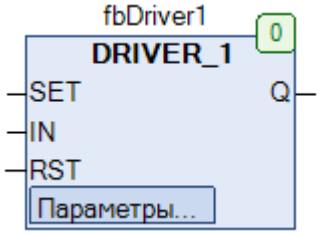


Рис. 24.1. Внешний вид ФБ **DRIVER_1** на языке CFC

Функциональный блок **DRIVER_1** представляет собой модуль обработки нажатия кнопки. Если параметр **Toggle_Mode** имеет значение **FALSE**, то по переднему фронту на входе **IN** выход **Q** принимает значение **TRUE** на время **Timeout**, после чего выход будет автоматически сброшен в состояние **FALSE**. В течение этого времени вход **IN** не обрабатывается. Если параметр **Toggle_Mode** имеет значение **TRUE**, то по переднему фронту на входе **IN** значение выхода **Q** инвертируется. Параметр **Timeout** определяет время, через которое выход **Q** будет автоматически сброшен в состояние **FALSE** (если выход имеет значение **TRUE**). Для отключения таймаута необходимо задать **Timeout=T#0ms**.

Вход **SET** позволяет управлять выходом **Q** независимо от состояния входа **IN**. По переднему фронту на входе **RST** выход **Q** принимает значение **FALSE** независимо от состояния входов **SET** и **IN**. См. также ФБ DRIVER_4, позволяющий обрабатывать нажатия 4-х кнопок.

24.2. DRIVER_4

Тип модуля: ФБ	**Переменная**	**Тип**	**Описание**
Входы	SET	BOOL	Ручное управление выходом.
	IN0…IN3	BOOL	Сигнал нажатия на кнопку 0…3.
	RST	BOOL	Переключение выхода в FALSE.
Выходы	Q0…Q3	BOOL	Выход кнопки 0…3.
Параметры	Toggle_Mode	BOOL	Режим работы кнопки.
	Timeout	TIME	Таймаут сброса выходов.

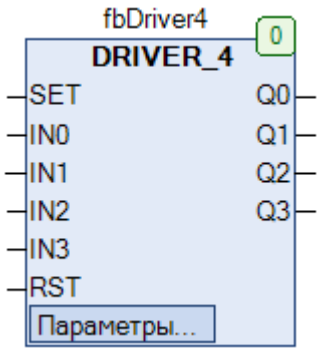


Рис. 24.2. Внешний вид ФБ **DRIVER_4** на языке CFC

Функциональный блок **DRIVER_4** представляет собой модуль обработки нажатий для 4-х кнопок. Принцип работы блока полностью соответствует ФБ DRIVER_1, единственным отличием является число обрабатываемых кнопок.

24.3. DRIVER_4C

Тип модуля: ФБ	Переменная	Тип	Описание
Входы	IN	BOOL	Сигнал переключения выходов.
	RST	BOOL	Сигнал сброса блока.
Выходы	SN	INT	Номер текущего состояния.
	Q0…Q3	BOOL	Выходы блока.
Параметры	Timeout	TIME	Таймаут переключения в состояние 0.
	SX	ARRAY [1..7] OF BYTE	Массив битовых масок выходов.

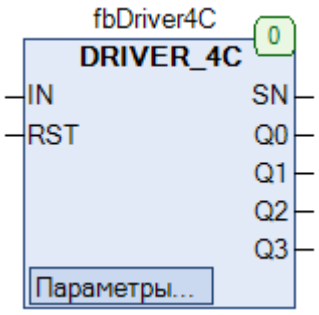


Рис. 24.3. Внешний вид ФБ **DRIVER_4C** на языке CFC

Функциональный блок **DRIVER_4C** представляет собой модуль управления с 7-ю заданными состояниями выходов, которые устанавливаются в параметре **SX** в виде битовых масок. Выход **SN** определяет номер текущего состояния. По умолчанию модуль находится в состоянии **0**, при этом **SN=0**, а битовая маска выходов = **2#0000**. По переднему фронту на входе **IN** происходит переключение выходов в состояние **SN=1**, при котором битовая маска выходов = **SX[1]** (где **Q0** – младший бит битовой маски). При последующих передних фронтах на входе **IN** модуль будет последовательно переключаться в состояния **2…7**, и после 7-ого состояния вернется в нулевое. Если одна из битовых масок массива **SX** равна **2#0000**, то по ее достижению модуль вернется в состояние **SN=0**. Значение параметра **Timeout** ограничивает время, которое модуль может провести в состояниях, отличных от нулевого. По истечении этого времени модуль вернется в состояние **SN=0**. Для отключения таймаута необходимо задать **Timeout=T#0ms**. По переднему фронту на входе **RST** происходит принудительное переключение модуля в состояние **SN=0**.

Рассмотрим пример работы с модулем. Пусть Timeout=T#0ms, SX=[2#0001, 2#0011, 2#0111, 2#0000, 2#1111, 2#1100, 2#1110].

Сигнал управления	SN	Q0	Q1	Q2	Q3
-	0	FALSE	FALSE	FALSE	FALSE
Передний фронт по IN	1	TRUE	FALSE	FALSE	FALSE
Передний фронт по IN	2	TRUE	TRUE	FALSE	FALSE
Передний фронт по IN	3	TRUE	TRUE	TRUE	FALSE
Передний фронт по IN	0	FALSE	FALSE	FALSE	FALSE
Передний фронт по IN	1	TRUE	FALSE	FALSE	FALSE
Передний фронт по RST	0	FALSE	FALSE	FALSE	FALSE

24.4. FLOW_CONTROL

Тип модуля: ФБ	Переменная	Тип	Описание
Входы	IN	BOOL	Сигнал ручного управления.
	REQ	BOOL	Сигнал автоматического управления.
	ENQ	BOOL	Разрешение управления.
	RST	BOOL	Сигнал сброса.
Выходы	Q	BOOL	Сигнал управления клапаном.
	STATUS	BYTE	ESR-код.
Параметры	T_AUTO	TIME	Время активности выхода Q
	T_DELAY	TIME	Время блокировки управления REQ.
Используемые модули	TP_1D		

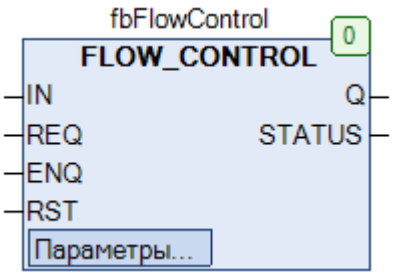


Рис. 24.4. Внешний вид ФБ **FLOW_CONTROL** на языке CFC

Функциональный блок **FLOW_CONTROL** используется для управления клапаном. Если вход **ENQ** имеет значение **TRUE**, то блок находится в работе и разрешает управление выходом **Q** через входы **IN** и **REQ**. Вход **IN** позволяет вручную управлять выходом **Q**. По переднему фронту на входе **REQ** выход **Q** принимает значение **TRUE** на время **T_AUTO**, после чего выход сбрасывается **FALSE** и управление через вход **REQ** блокируется на время **T_DELAY**. В течение времени блокировки можно управлять выходом через вход **IN**. В течение времени **T_AUTO** выход может быть сброшен в состояние **FALSE** с помощью переднего фронта на входе **RST**.

Выход **Status** определяет состояние блока и совместим с ESR-модулями:

Значение выхода Status	Описание
100	Блок готов к работе.
101	Выход включен с помощью входа IN.
102	Выход включен с помощью входа REQ.
103	Выход отключен с помощью входа RST.

24.5. FT_PROFILE

Тип модуля: ФБ	Переменная	Тип	Описание
Входы	K	REAL	Коэффициент масштабирования.
	O	REAL	Смешение.
	M	REAL	Коэффициент масштаба времени.
	E	BOOL	Сигнал запуска блока.
Выходы	Y	REAL	Выход генератора.
	RUN	BOOL	Флаг «блок в работе».
	ET	TIME	Время, прошедшее с начала работы.
Параметры	value_0, value_1, value_2, value_3, value_10, value_11, value_12, value_13	REAL	Значение уставки.
	time_1, time_2, time_3, time_10, time_11, time_12, time_13	TIME	Время достижения уставки.
Используемые модули	T_PLC_MS, MULTIME		

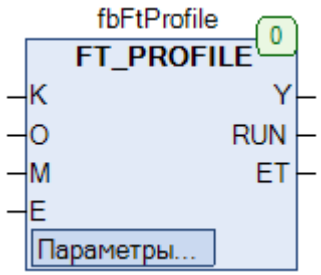


Рис. 24.5. Внешний вид ФБ **FT_PROFILE** на языке CFC

Функциональный блок **FT_PROFILE** используется для изменения величины по заданной зависимости «уставка – время достижения» (например, для температурного профиля печи). Профиль разбивается на 7 участков (с номерами 1, 2, 3, 10, 11, 12, 13), продолжительности которых определяют соответствующие параметры **time_x**. Следует заметить, что все интервалы **time_x** отсчитываются относительно момента запуска блока, поэтому каждое последующее значение времени должно быть больше предыдущего. Значения крайних точек участков определяются параметрами **value_x**. Переходы между участками происходят по линейной зависимости. Входы **K** и **O** определяют коэффициент масштабирования и смещения выхода **Y**; вход **M** определяет коэффициент масштаба времени. Значение выхода **Y** на участке **b** определяется следующей формулой:

$$\mathrm{Y(b)} = \mathrm{K} \cdot (\mathrm{value\_b} - \mathrm{value\_a}) \cdot \frac{\text{время с начала отработки участка } b}{M \cdot (\mathrm{time\_b} - \text{время с начала отработки участка } b)} + \mathrm{O}, \text{ где a=b-1}$$

Блок запускается в работу импульсом по переднему фронту на входе **E**; при этом участки 11-13 будут обработаны только в том случае, если к началу времени их обработки **E** уже имеет значение **FALSE**. Выход **RUN** имеет значение **TRUE**, пока блок находится в работе; выход **ET** отображает время, прошедшее с начала работы блока. После завершения обработки всех участков выход **RUN** принимает значение **FALSE**, а на выходе **ET** сохраняется общее время работы блока.

Каждый импульс по переднему фронту на входе **IN** запускает работу блока заново.

На рис. 24.6 приведена трассировка блока для случая: K=M=1.0, O=0.0.

value_0	value_1	value_2	value_3	value_10	value_11	value_12	value_13
10.0	15.0	25.0	-10.0	5.0	15.0	25.0	50.0
	time_1	**time_1**	**time_1**	**time_1**	**time_1**	**time_1**	**time_13**
	T#10s	T#30s	T#50s	T#55s	T#80s	T#90s	T#120s

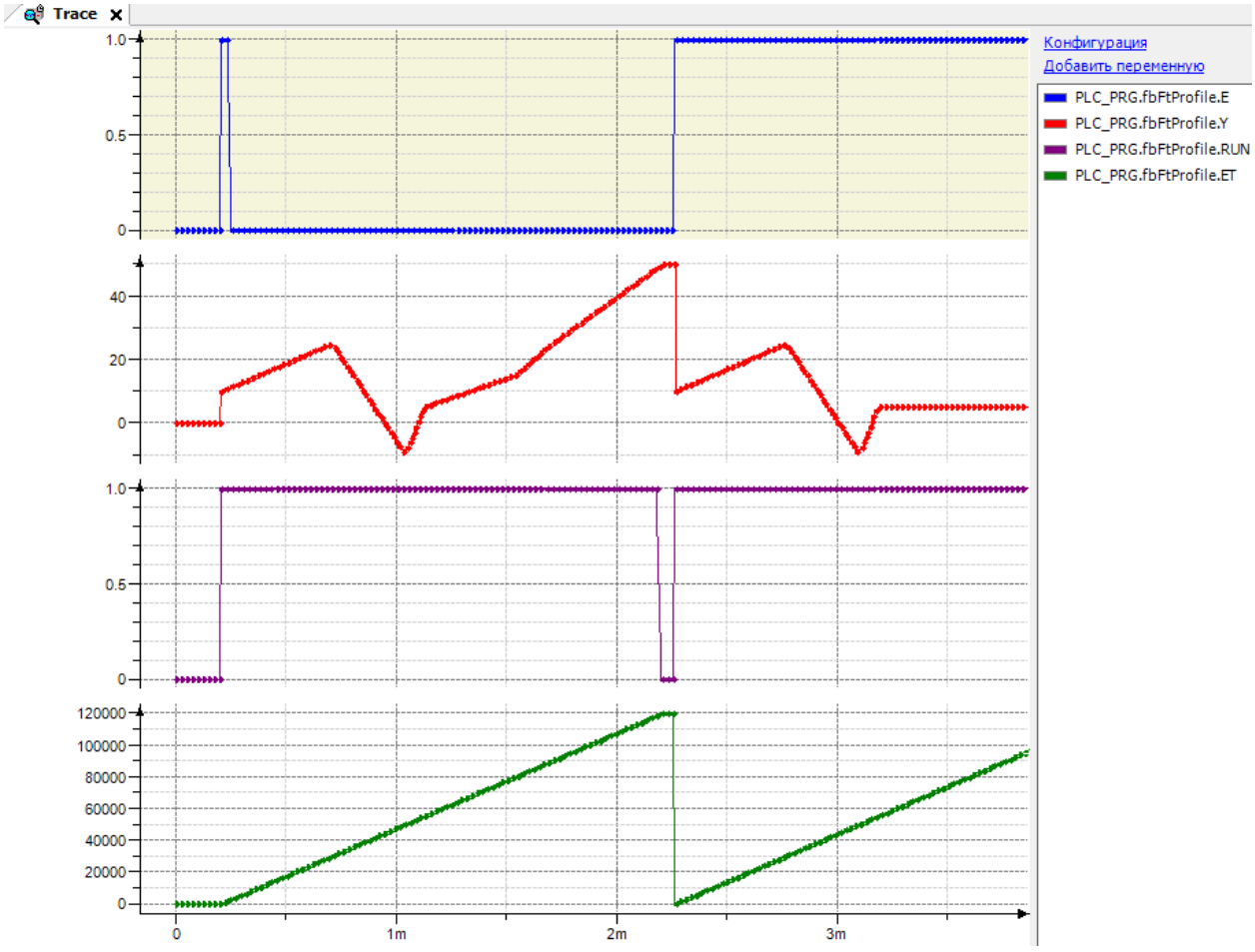


Рис. 24.6. Трассировка ФБ **FT_PROFILE**

Время **ET** по оси **Y** измеряется в мс. ***Обратите внимание***, на трассировке показано два запуска блока: первый из них – импульсом (и тогда участки 11-13 обрабатываются), второй – уровнем (и тогда участки 11-13 не обрабатываются).

24.6. INC_DEC

Тип модуля: ФБ	Переменная	Тип	Описание
Входы	CHa	BOOL	Канал A.
	CHb	BOOL	Канал B.
	RST	BOOL	Сигнал сброса.
Выходы	dir	BOOL	Направление вращения.
	cnt	INT	Счетчик фронтов.

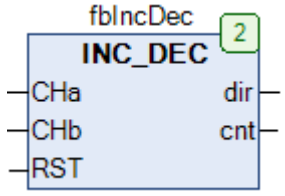


Рис. 24.7. Внешний вид ФБ **INC_DEC** на языке CFC

Функциональный блок **INT_DEC** представляет собой модуль инкрементального квадратурного энкодера. На входы **CHa** и **CHb** должны быть поданы сигналы каналов **A** и **B** энкодера. Выход **dir** определяет направление вращения (**TRUE** – прямое, **FALSE** – обратное). Выход **cnt** содержит сумму подсчитанных фронтов (как передних, так и задних) по обоим каналов. По переднему фронту на входе **RST** счетчик обнуляется.

Ниже приведен пример работы с блоком, в котором ФБ GEN_BIT используется для эмуляции энкодера.

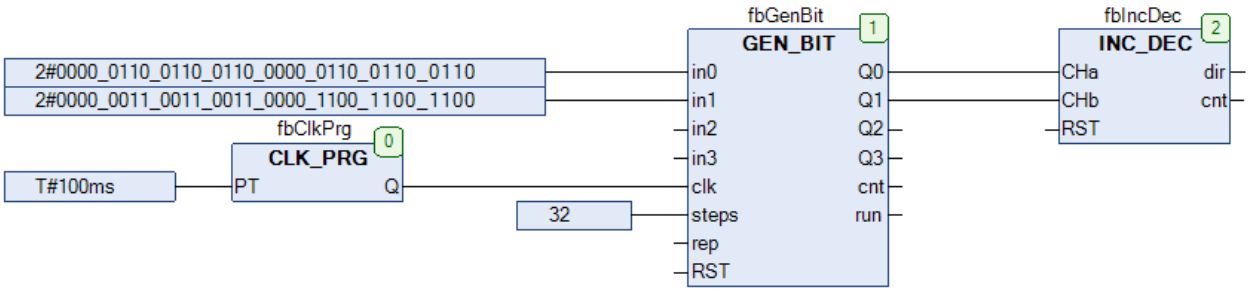


Рис. 24.8. Пример работы с ФБ **INC_DEC** на языке CFC

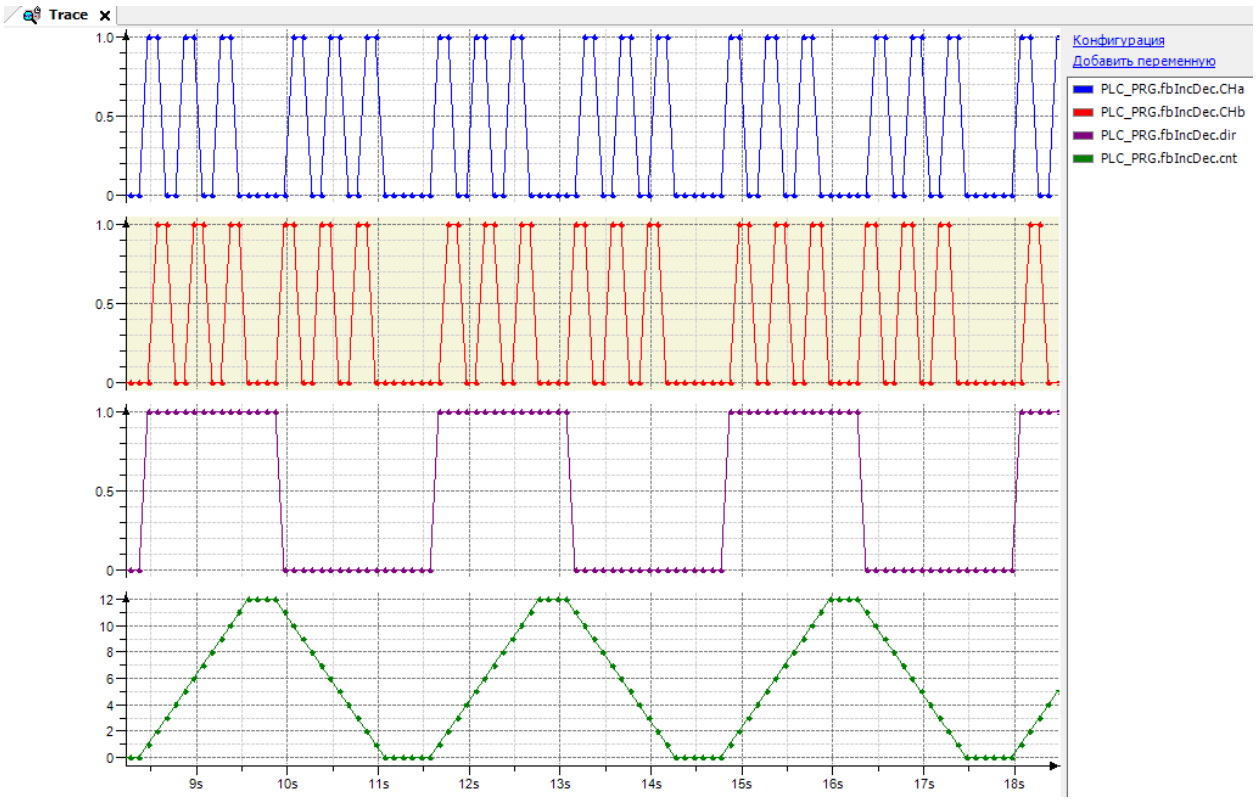


Рис. 24.9. Трассировка ФБ **INC_DEC** (см. рис. 24.8)

24.7. INTERLOCK

Тип модуля: ФБ	Переменная	Тип	Описание
Входы	I1	BOOL	Вход 1.
	I2	BOOL	Вход 2.
	TL	TIME	Время блокировки.
Выходы	Q1	BOOL	Выход 1.
	Q2	BOOL	Выход 2.

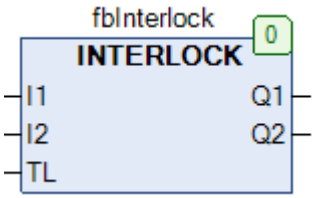


Рис. 24.10. Внешний вид ФБ **INTERLOCK** на языке CFC

Функциональный блок **INTERLOCK** содержит два входа (**I1** и **I2**), которые управляют соответствующими выходами (**Q1** и **Q2**). В каждый момент времени только один из выходов может быть активен. Ниже приведена таблица состояний входов и выходов.

I1	I2	Q1	Q2
FALSE	FALSE	FALSE	FALSE
FALSE	TRUE	FALSE	TRUE
TRUE	FALSE	TRUE	FALSE
TRUE	TRUE	FALSE	FALSE

Вход **TL** определяет время блокировки между отключением одного выхода и включением другого (т.е. после того, как выход **Q1** примет значение **FALSE**, выход **Q2** не сможет принять значение **TRUE** до истечения времени **TL**).

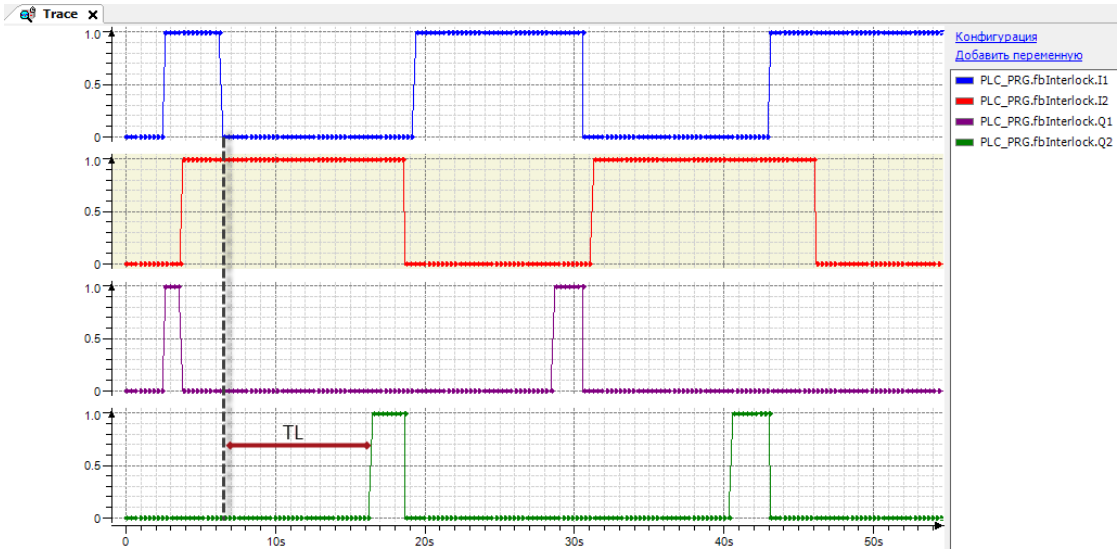


Рис. 24.11. Трассировка ФБ **INTERLOCK** (TL=T#10s)

24.8. INTERLOCK_4

Тип модуля: ФБ	Переменная	Тип	Описание
Входы	I0	BOOL	Вход 0.
	I1	BOOL	Вход 1.
	I2	BOOL	Вход 2.
	I3	BOOL	Вход 3.
	E	BOOL	Сигнал управления блоком.
	MODE	INT	Режим работы блока.
Выходы	OUT	BYTE	Выход блока.
	TP	BOOL	Флаг «значение выхода изменилось».

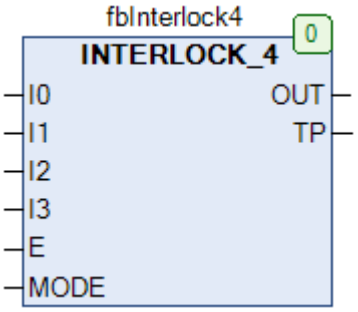


Рис. 24.12. Внешний вид ФБ **INTERLOCK_4** на языке CFC

Функциональный блок **INTERLOCK_4** содержит четыре входа типа BOOL (**I0…I3**), которые управляют выходом типа BYTE (**OUT**). Если вход **E** имеет значение **FALSE**, то выход **OUT** имеет значение **0**. Если вход **E** имеет значение **TRUE**, то значение выхода **OUT** определяется значениями входов **I0…I3** и режимом работы **MODE**:

Mode	Описание
0	Все входы транслируются на выход (I0 – младший бит).
1	На выход транслируется только старший активный бит.
2	На выход транслируется активный бит, изменившийся последним.
3	На выход транслируется активный бит, причем до его отключения остальные входы не обрабатываются.

Выход **TP** принимает значение **TRUE** на 1 цикл ПЛК при каждом изменении выхода **OUT**.

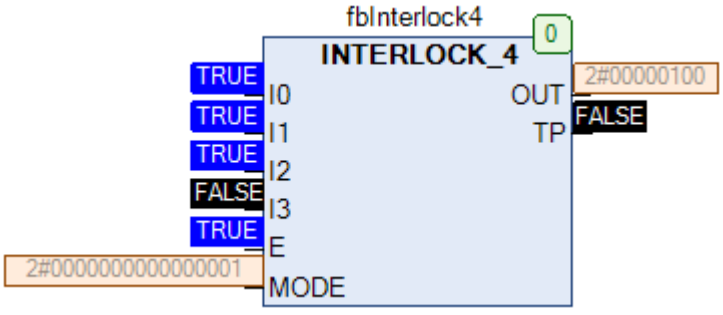


Рис. 24.13. Пример работы с ФБ **INTERLOCK_4** на языке CFC

24.9. MANUAL

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	IN	BOOL	Сигнал дистанционного управления.
	ON	BOOL	Сигнал местного управления «Включить».
	OFF	BOOL	Сигнал местного управления «Выключить».
Выходы	MANUAL	BOOL	Выход функции.

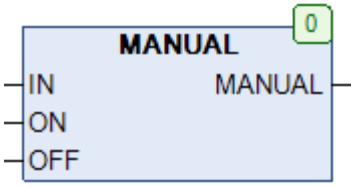


Рис. 24.14. Внешний вид функции **MANUAL** на языке CFC

Функция **MANUAL** представляет собой модуль местного/дистанционного управления. Выход функции определяется значением входа дистанционного управления **IN** и входами локального управления **ON/OFF** («включить»/»выключить»), причем сигналы управления имеют следующие приоритеты:

IN	**ON**	**OFF**	**MANUAL**
FALSE	FALSE	FALSE	FALSE
TRUE	FALSE	FALSE	TRUE
не имеет значения	не имеет значения	TRUE	FALSE
не имеет значения	TRUE	FALSE	TRUE

Таким образом, местное управление имеет приоритет по сравнению с дистанционным, а сигнал местного управления «Выключить» - над сигналом «Включить».

24.10. MANUAL_1

Тип модуля: ФБ	**Переменная**	**Тип**	**Описание**
Входы	IN	BOOL	Сигнал дистанционного управления.
	MAN	BOOL	Режим управления.
	M_I	BOOL	Сигнал местного управления.
	SET	BOOL	Сигнал принудительного включения.
	RST	BOOL	Сигнал принудительного отключения.
Выходы	Q	BOOL	Выход блока.
	STATUS	BYTE	ESR-код.

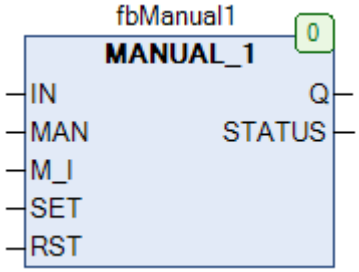


Рис. 24.15. Внешний вид ФБ **MANUAL_1** на языке CFC

Функциональный блок **MANUAL_1** представляет собой модуль местного/дистанционного управления. Если вход **MAN** имеет значение **FALSE**, то значение выхода **Q** определяется значением входа **IN**. Если вход **MAN** имеет значение **TRUE**, то значение выхода **Q** определяется значением входа **M_I** и, кроме того, может быть принудительно изменено на **TRUE/FALSE** по переднему фронту входа **SET/RST**. После импульса на входе **SET** или **RST** значение входа **M_I** перестает влиять на значение выхода; чтобы вернуться к управлению через вход **M_I** необходимо переключить вход **MAN** в состояние **FALSE**, после чего опять вернуть его в **TRUE**.

Выход **Status** определяет состояние блока и совместим с ESR-модулями:

Значение выхода Status	**Описание**
100	Блок в режиме дистанционного управления (MAN=FALSE).
101	Выход принудительно взведен в TRUE (через SET).
102	Выход принудительно сброшен в FALSE (через RST).
103	Блок в режиме местного управления (MAN=TRUE).

24.11. MANUAL_2

Тип модуля: ФБ	Переменная	Тип	Описание
Входы	IN	BOOL	Сигнал дистанционного управления.
	ENA	BOOL	Сигнал управления блоком.
	ON	BOOL	Сигнал принудительного включения.
	OFF	BOOL	Сигнал принудительного отключения.
	MAN	BOOL	Сигнал местного управления.
Выходы	Q	BOOL	Выход блока.
	STATUS	BYTE	ESR-код.

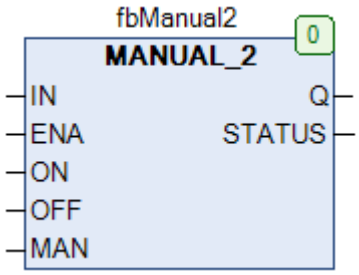


Рис. 24.16. Внешний вид ФБ **MANUAL_2** на языке CFC

Функциональный блок **MANUAL_2** представляет собой модуль местного/дистанционного управления. Если вход **ENA** имеет значение **FALSE**, то блок отключен и выход **Q** имеет значение **FALSE**. Если вход **ENA** имеет значение **TRUE**, то блок находится в работе и значение выхода **Q** определяется состояниями входов **IN**, **ON**, **OFF** и **MAN** («-» – состояние входа не имеет значения, «x» – совпадающие состояния входов и выходов). Выход **Status** определяет состояние блока и совместим с ESR-модулями.

ENA	IN	ON	OFF	MAN	Q	STATUS	Описание
FALSE	-	-	-	-	FALSE	104	Блок отключен.
TRUE	x	FALSE	FALSE	-	x	100	Блок в режиме дист. управления.
TRUE	-	TRUE	FALSE	-	TRUE	101	Выход принудительно взведен в TRUE.
TRUE	-	FALSE	TRUE	-	FALSE	102	Выход принудительно сброшен в FALSE.
TRUE	-	TRUE	TRUE	x	x	103	Блок в режиме местного управления.

24.12. MANUAL_4

Тип модуля: ФБ	Переменная	Тип	Описание
Входы	I0…I3	BOOL	Сигналы дистанционного управления.
	MAN	BOOL	Режим управления.
	STP	BOOL	Сигнал пошагового переключения.
	M0…M3	BOOL	Сигналы местного управления
Выходы	Q0…Q3	BOOL	Выходы блока.
	STATUS	BYTE	ESR-код.
Использованные модули	INC		

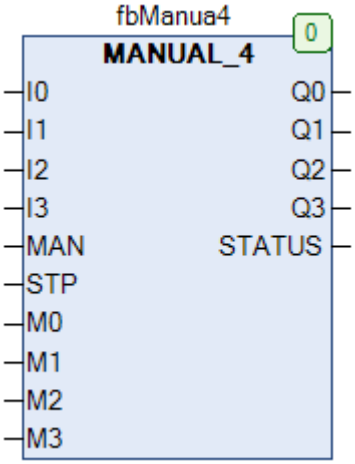


Рис. 24.17. Внешний вид ФБ **MANUAL_4** на языке CFC

Функциональный блок **MANUAL_4** представляет собой 4-х канальный модуль местного/дистанционного управления. Если вход **MAN** имеет значение **FALSE**, то значения выходов **Q0…Q3** определяются значениями входов **I0…I3**. Если вход **MAN** имеет значение **TRUE**, то значения выходов **Q0…Q3** определяются значениями входов **M0…M3**. Если вход **MAN** имеет значение **TRUE**, и на входе **STP** детектируется импульс по переднему фронту, то блок переходит в режим пошагового переключения выходов – при этом выход **Q0** принимает значение **TRUE**, а все остальные выходы – **FALSE**. Каждый следующий импульс на входе **STP** будет переключать активный выход блока:

Действие	Q0	Q1	Q2	Q3
Импульс по переднему фронту на входе STP	TRUE	FALSE	FALSE	FALSE
Импульс по переднему фронту на входе STP	FALSE	TRUE	FALSE	FALSE
Импульс по переднему фронту на входе STP	FALSE	FALSE	TRUE	FALSE
Импульс по переднему фронту на входе STP	FALSE	FALSE	FALSE	TRUE
Импульс по переднему фронту на входе STP	TRUE	FALSE	FALSE	FALSE
…	…	…	…	…

Выход **Status** определяет состояние блока и совместим с ESR-модулями:

STATUS	Описание
100	Блок в режиме дистанционного управления (MAN=FALSE).
101	Блок в режиме местного управления (MAN=TRUE).
110/111/112/113	Блок в режиме пошагового переключения, активен выход Q0/Q1/Q2/Q3.

24.13. PARSET

Тип модуля: ФБ	Переменная	Тип	Описание
Входы	A0	BOOL	Сигнал переключения выходов 0.
	A1	BOOL	Сигнал переключения выходов 1.
Выходы	P1…P4	REAL	Выходы блока.
Параметры	X01…X04, X11…X14, X21…X24, X31…X34	REAL	Возможные значения выходов.
	TC	TIME	Время перехода к новому значению.
Использованные модули	T_PLC_MS		

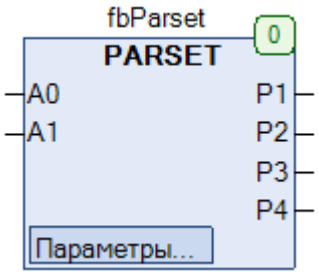


Рис. 24.18. Внешний вид ФБ **PARSET** на языке CFC

Функциональный блок **PARSET** передает на выходы **P1…P4** одну из 4-х заранее подготовленных групп значения (X01…X04/X11…X14/X21…X24/X31…X34) в зависимости от состояний входов **A0** и **A1**:

A0	A1	P1	P2	P3	P4
FALSE	FALSE	X01	X02	X03	X03
FALSE	TRUE	X11	X12	X13	X14
TRUE	FALSE	X21	X22	X23	X24
TRUE	TRUE	X31	X32	X33	X34

Если параметр **TC** имеет значение **T#0ms**, то при переключении выходов значения меняются мгновенно; если значение **TC** отлично от нуля, то изменение значения выходов происходит плавно, по линейной зависимости за время **TC**.

24.14. PARSET2

Тип модуля: ФБ	Переменная	Тип	Описание
Входы	X	REAL	Входное значение.
Выходы	P1…P4	REAL	Выходы блока.
Параметры	X01…X04, X11…X14, X21…X24, X31…X34	REAL	Возможные значения выходов.
	L1…L3	REAL	Граничные значения для X.
	TC	TIME	Время перехода к новому значению.
Использованные модули	PARSET		

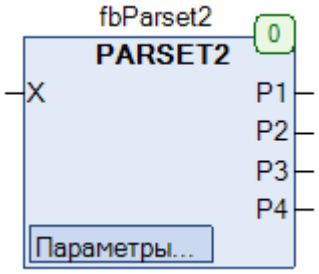


Рис. 24.19. Внешний вид ФБ **PARSET2** на языке CFC

Функциональный блок **PARSET2** передает на выходы **P1…P4** одну из 4-х заранее подготовленных групп значения (X01…X04/X11…X14/X21…X24/X31…X34) в зависимости от значения входа **X** и заданных для него граничных значений **L1…L3**:

X	P1	P2	P3	P4
X < L1	X01	X02	X03	X03
L1 < X < L2	X11	X12	X13	X14
L2 < X < L3	X21	X22	X23	X24
L3 ≤ X	X31	X32	X33	X34

Если параметр **TC** имеет значение **T#0ms**, то при переключении выходов значения меняются мгновенно; если значение **TC** отлично от нуля, то изменение значения выходов происходит плавно, по линейной зависимости за время **TC**.

24.15. SIGNAL

Тип модуля: ФБ	Переменная	Тип	Описание
Входы	IN	BOOL	Сигнал управления блоком.
	SIG	BYTE	Последовательность бит.
	TS	TIME	Период переключения выхода.
Выходы	Q	BOOL	Выход генерации последовательности бит.
Использованные модули	T_PLC_MS		

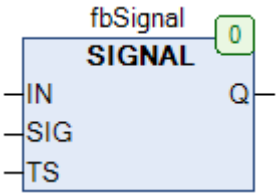


Рис. 24.20. Внешний вид ФБ **SIGNAL** на языке CFC

Функциональный блок **SIGNAL** используется для циклической генерации заданной последовательности бит. Пока вход **IN** имеет значение **TRUE**, на выходе **Q** генерируется последовательность бит, определенная значением входа **SIG**, при этом изменение выхода происходит с периодичностью **TS**. Генерация сигнала начинается с произвольного бита последовательности.

На рис. 24.21 приведена трассировка работы блока для случая SIG=2#1101_0110, TS=T#1s (т.е. время полной последовательности = 8 секунд).

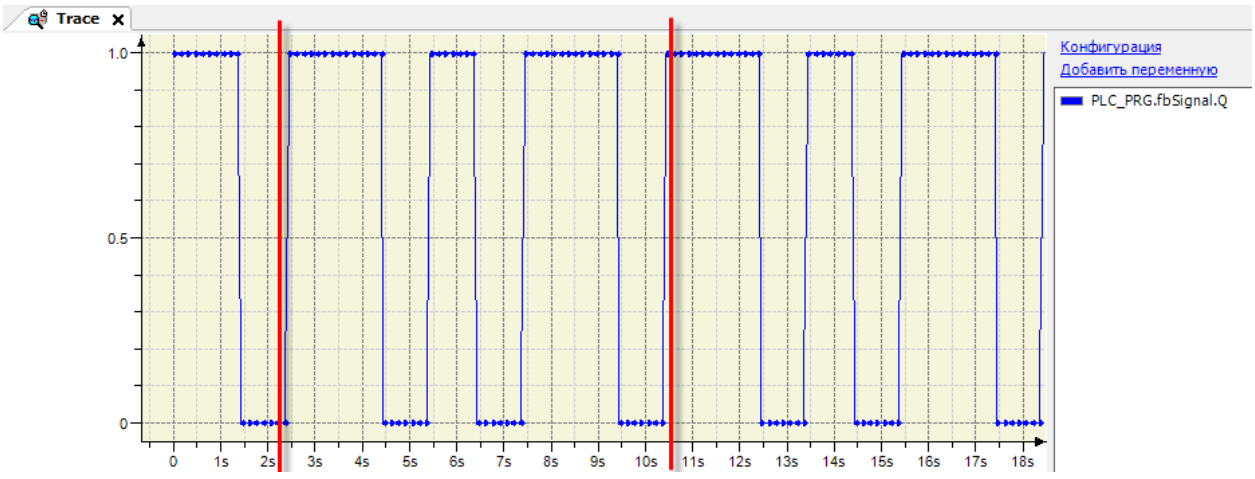


Рис. 24.21. Трассировка ФБ **SIGNAL**

24.16. SIGNAL_4

Тип модуля: ФБ	Переменная	Тип	Описание
Входы	IN1…IN4	BOOL	Сигнал выбора последовательности.
	TS	TIME	Период переключения выхода.
Выходы	Q	BOOL	Выход генерации последовательности бит.
Параметры	S1…S4	BYTE	Заданные последовательности бит.
Использованные модули	SIGNAL		

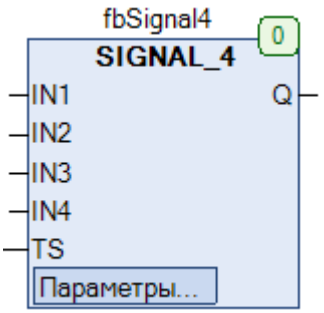


Рис. 24.22. Внешний вид ФБ **SIGNAL_4** на языке CFC

Функциональный блок **SIGNAL_4** используется для циклической генерации одной из 4-х заданных последовательностей бит. Пока один из входов **IN1…IN4** имеет значение **TRUE**, на выходе **Q** генерируется соответствующая последовательность бит (**S1…S4**), при этом изменение выхода происходит с периодичностью **TS**. Генерация сигнала начинается с произвольного бита последовательности. Если одновременно активны несколько входов, то на выходе генерируется последовательность с наименьшим номером.

См. также описание ФБ SIGNAL.

24.17. SRAMP

Тип модуля: ФБ	Переменная	Тип	Описание
Входы	X	REAL	Входной сигнал.
	A_UP	REAL	Ускорение нарастания выходного сигнала.
	A_DN	REAL	Ускорение спада выходного сигнала.
	VU_MAX	REAL	Макс. скорость нарастания выходного сигнала.
	VD_MAX	REAL	Макс. скорость спада выходного сигнала.
	LIMIT_HIGH	REAL	Верхний предел выходного сигнала.
	LIMIT_LOW	REAL	Нижний предел выходного сигнала.
	RST	BOOL	Сигнал сброса блока.
Выходы	Y	REAL	Выходной сигнал.
	V	REAL	Скорость изменения выходного сигнала.
Используемые модули	TC_S		

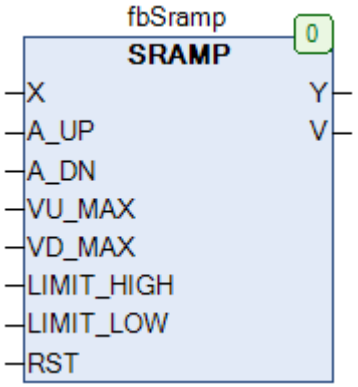


Рис. 24.23. Внешний вид функции **SRAMP** на языке CFC

Функциональный блок **SRAMP** используется для плавного изменения выходного сигнала при изменении входного. Значения входов блока ограничивают скорость нарастания и спада выходного сигнала **Y** (**VU_MAX**, **VD_MAX**), ускорение его нарастания и спада (**A_UP**, **A_DN**) и максимальное/минимальное допустимое значение (**LIMIT_HIGH**, **LIMIT_LOW**). Значения **VU_MAX** и **A_UP** должны быть положительными, а **VD_MAX** и **A_DN** – отрицательными. На выход **V** подается текущая скорость изменения выходного сигнала. По переднему фронту на входе **RST** происходит обнуление выходов блока.

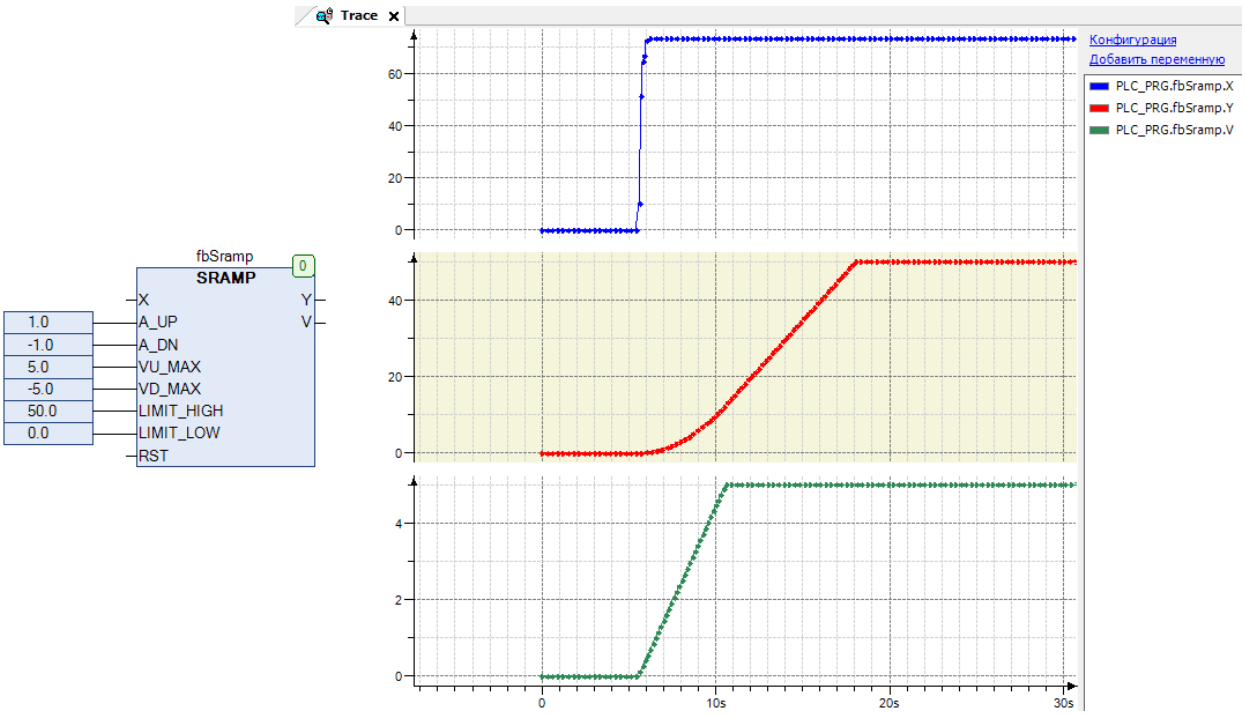


Рис. 24.24. Трассировка ФБ **SRAMP**

24.18. TUNE

Тип модуля: ФБ	Переменная	Тип	Описание
Входы	SET	BOOL	Сигнал установки значения SET_val.
	SU	BOOL	Сигнал увеличения значения.
	SD	BOOL	Сигнал уменьшения значения.
	RST	BOOL	Сигнал установки значения RST_val.
Выходы	Y	REAL	Изменяемое значение.
Параметры	SS	REAL	Шаг изменения значения по нажатию кнопок SU/SD.
	Limit_H	REAL	Верхний предел значения выхода.
	Limit_L	REAL	Нижний предел значения выхода.
	SET_val	REAL	Значение для сигнала SET.
	RST_val	REAL	Значение для сигнала RST.
	T1	TIME	Время зажатия кнопки для S2.
	T2	TIME	Время зажатия кнопки для S2.
	S1	REAL	Шаг изменения значения при зажатии кнопок SU/SD на время T1, ед/с.
	S2	REAL	Шаг изменения значения при зажатии кнопок SU/SD на время T2, ед/с.
Использованные модули	T_PLC_MS		

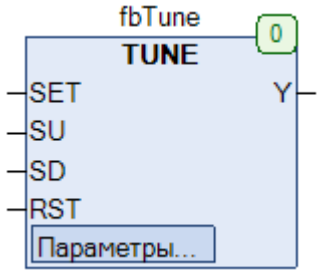


Рис. 24.25. Внешний вид ФБ **TUNE** на языке CFC

Функциональный блок **TUNE** представляет собой модуль управления значением. По переднему фронту на входе **SU/SD** значение выхода **Y** увеличивается/уменьшается на величину **SS**. Таким образом, входы **SU** и **SD** могут быть привязаны к кнопкам «больше»/ «меньше». Если вход **SU/SD** сохраняет значение **TRUE** в течение времени **T1/T2** (т.е. кнопка зажата), то значение **Y** начинается каждую секунду увеличиваться/уменьшаться на величину **S1/S2**. По переднему фронту на входе **SET/RST** выход **Y** принимает значение **SET_val/RST_val**. Значение **Y** ограничено диапазоном [**Limit_L…Limit_H**].

24.19. TUNE2

Тип модуля: ФБ	Переменная	Тип	Описание
Входы	SET	BOOL	Сигнал установки значения SET_val.
	SU	BOOL	Сигнал увеличения значения (SS).
	SD	BOOL	Сигнал уменьшения значения (SS).
	FU	BOOL	Сигнал увеличения значения (FS).
	FD	BOOL	Сигнал уменьшения значения (FS).
	RST	BOOL	Сигнал установки значения RST_val.
Выходы	Y	REAL	Изменяемое значение.
Параметры	SS	REAL	Шаг изменения значения по нажатию кнопок SU/SD.
	FS	REAL	Шаг изменения значения по нажатию кнопок FU/FD.
	Limit_H	REAL	Верхний предел значения выхода.
	Limit_L	REAL	Нижний предел значения выхода.
	SET_val	REAL	Значение для сигнала SET.
	RST_val	REAL	Значение для сигнала RST.
	TR	TIME	Время зажатия кнопки для S2.
	S1	REAL	Шаг изменения значения при зажатии кнопок SU/SD на время TR, ед/с.
	S2	REAL	Шаг изменения значения при зажатии кнопок FU/FD на время TR, ед/с.
Использованные модули	T_PLC_MS		

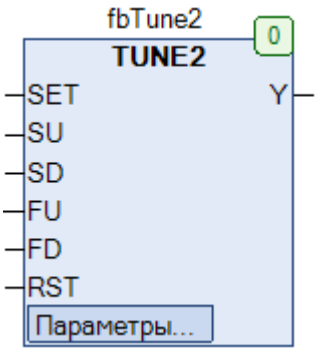


Рис. 24.26. Внешний вид ФБ **TUNE2** на языке CFC

Функциональный блок **TUNE2** представляет собой модуль управления значением. По переднему фронту на входе **SU/SD** значение выхода **Y** увеличивается/уменьшается на величину **SS**. По переднему фронту на входе **FU/FD** значение выхода **Y** увеличивается/уменьшается на величину **FS**. Таким образом, входы **SU** и **SD** могут быть привязаны к кнопкам «больше»/«меньше», а входы **FD/FS** – к кнопкам «значительно больше»/«значительно меньше». Если вход **SU/SD** сохраняет значение **TRUE** в течение времени **TR** (т.е. кнопка зажата), то значение **Y** начинается каждую секунду увеличиваться/уменьшаться на величину **S1**. Если вход **FU/FD** сохраняет значение **TRUE** в течение времени **TR**, то значение Y начинается каждую секунду увеличиваться/уменьшаться на величину **S2**. По переднему фронту на входе **SET/RST** выход **Y** принимает значение **SET_val/RST_val**. Значение **Y** ограничено диапазоном [**Limit_L…Limit_H**].

25. Работа с буфером

25.0. Вступление

Функции, описанные в данной главе, используются для операций с буфером. Буфер представляет собой массив байт. Входными переменными для каждой функции являются указатель на массив (**PT**) и размер массива (**SIZE**). В большинстве случаев представляется удобным использовать операторы **ADR** и **SIZEOF**, которые возвращают адрес и размер массива в байтах соответственно. Тогда вызов функции для обработки буфера **abyBuffer** можно представить следующим образом:

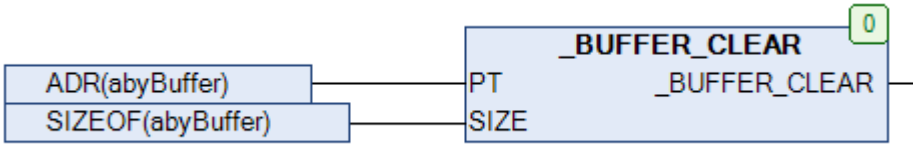


Рис. 25.1. Вызов функции работы с буфером на языке CFC

Обработка массива, полученного по указателю, производится путем прямых манипуляций с памятью ПЛК. Этот тип обработки является крайне эффективным, так как не требует копирования содержимого массива.

25.1. _BUFFER_CLEAR

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	PT	POINTER TO BYTE	Указатель на буфер.
	SIZE	UINT	Размер буфера.
Выходы	_BUFFER_CLEAR	BOOL	Флаг окончания обработки буфера.

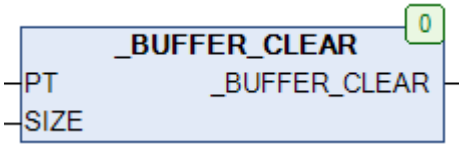


Рис. 25.2. Внешний вид функции **_BUFFER_CLEAR** на языке CFC

Функция **_BUFFER_CLEAR** присваивает каждому из элементов буфера значение **0**. Результаты записываются в тот же буфер по указателю. После окончания обработки выход функции принимает значение **TRUE**.

Пример: буфер [2, 12, 4, 8, 6] после обработки функцией примет вид [0, 0, 0, 0, 0].

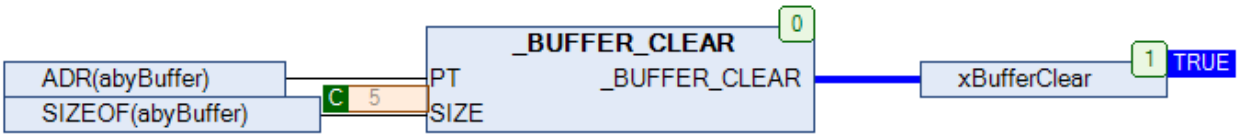


Рис. 25.3. Пример работы с функцией **_BUFFER_CLEAR** на языке CFC

25.2. _BUFFER_INIT

Тип модуля: функция	Переменная	Тип	Описание
Входы	PT	POINTER TO BYTE	Указатель на буфер.
	SIZE	UINT	Размер буфера.
	INIT	BYTE	Присваиваемое значение.
Выходы	_BUFFER_INIT	BOOL	Флаг окончания обработки буфера.

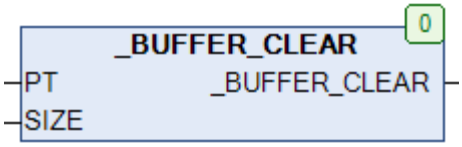


Рис. 25.4. Внешний вид функции **_BUFFER_INIT** на языке CFC

Функция **_BUFFER_INIT** присваивает каждому из элементов буфера значение **INIT**. Результаты записываются в тот же буфер по указателю. После окончания обработки выход функции принимает значение **TRUE**.

Пример: буфер [2, 12, 4, 8, 6] после обработки функцией примет вид [3, 3, 3, 3, 3].

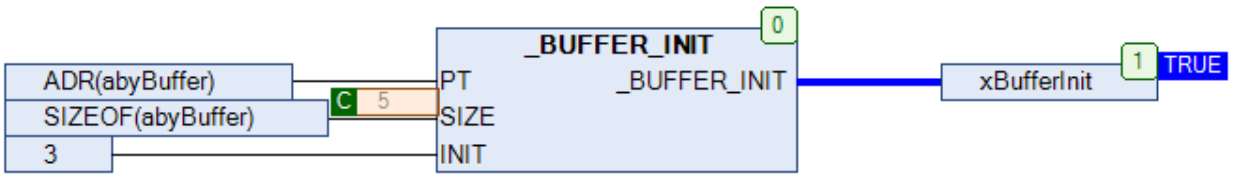


Рис. 25.5. Пример работы с функцией **_BUFFER_INIT** на языке CFC

25.3. _BUFFER_INSERT

Тип модуля: функция	Переменная	Тип	Описание
Входы	STR	STRING	Записываемая строка.
	POS	INT	Позиция для записи.
	PT	POINTER TO ARRAY [0..32767] OF BYTE	Указатель на буфер.
	SIZE	UINT	Размер буфера.
Выходы	_BUFFER_INSERT	INT	Позиция в буфере после записи.
Используемые модули	_STRING_TO_BUFFER		

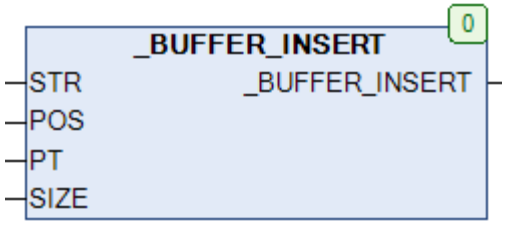


Рис. 25.6. Внешний вид функции _BUFFER_INSERT на языке CFC

Функция **_BUFFER_INSERT** записывает строку **STR** (в виде ASCII-кодов) в буфер с позиции **POS** (где **0** – позиция первого элемента буфера), смещая оставшуюся часть данных в буфере на длину строки. После окончания операции на выход функции транслируется позиция буфера после вставленной строки.

Пример: буфер [2, 12, 4, 8, 6, 0, 0, 0, 0, 0] после вызова функции в соответствии с рис 25.7 примет вид [2, 12, 116, 101, 115, 116, 4, 8, 6, 0], где 116/101/115/116 – это ASCII-коды (DEC) букв t/e/s/t соответственно.

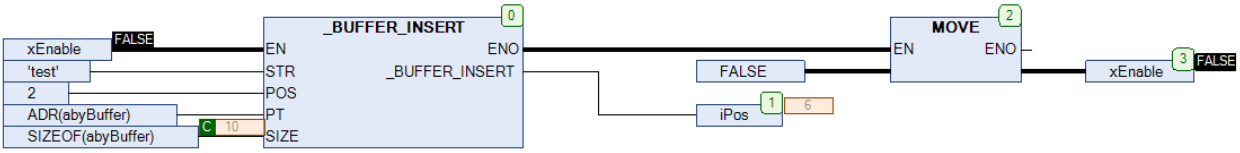


Рис. 25.7. Пример работы с функцией **_BUFFER_INSERT** на языке CFC

См. также функцию _STRING_TO_BUFFER, которая перезаписывает данные буфера, а не смещает их.

25.4. _BUFFER_UPPERCASE

Тип модуля: функция	Переменная	Тип	Описание
Входы	PT	POINTER TO BYTE	Указатель на буфер.
	SIZE	INT	Размер буфера.
Выходы	_BUFFER_UPPERCASE	BOOL	Флаг окончания обработки буфера.
Используемые модули	TO_UPPER		

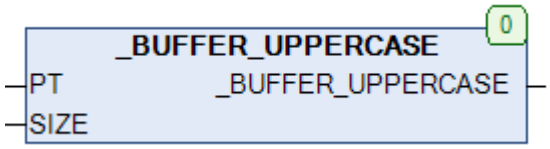


Рис. 25.8. Внешний вид функции **_BUFFER_UPPERCASE** на языке CFC

Функция **_BUFFER_UPPERCASE** интерпретирует буфер как набор ASCII-кодов, и конвертирует ASCII-коды букв в коды тех же букв в верхнем регистре.

Пример: буфер [242, 229, 241, 242] после обработки функцией примет вид [210, 197, 209, 210], где 242/229/241/242 – это ASCII-коды (DEC) букв т/е/с/т, а 210/197/209/210 – это ASCII-коды (DEC) букв Т/Е/С/Т.

25.5. _STRING_TO_BUFFER

Тип модуля: функция	Переменная	Тип	Описание
Входы	STR	STRING	Записываемая строка.
	POS	INT	Позиция для записи.
	PT	POINTER TO ARRAY [0..32767] OF BYTE	Указатель на буфер.
	SIZE	UINT	Размер буфера.
Выходы	_STRING_TO_BUFFER	INT	Позиция в буфере после записи.

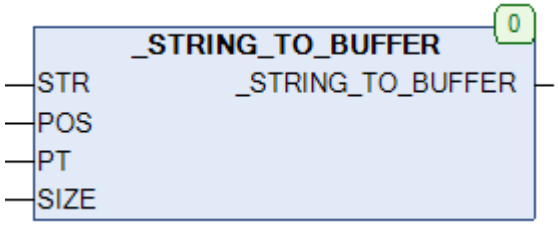


Рис. 25.9. Внешний вид функции **_STRING_TO_BUFFER** на языке CFC

Функция **_STRING_TO_BUFFER** записывает строку **STR** (в виде ASCII-кодов) в буфер с позиции **POS** (где **0** – позиция первого элемента буфера), перезаписывая хранящиеся в буфере данные. После окончания операции на выход функции транслируется позиция буфера после вставленной строки.

Пример: буфер [2, 12, 4, 8, 6, 0, 0, 0, 0, 0] после вызова функции в соответствии с рис 25.10 примет вид [2, 12, 116, 101, 115, 116, 0, 0, 0, 0], где 116/101/115/116 – это ASCII-коды (DEC) букв t/e/s/t соответственно.

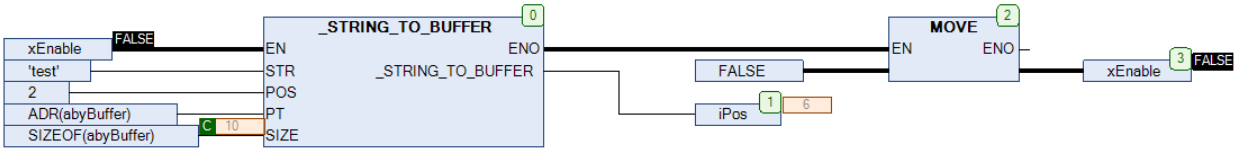


Рис. 25.10. Пример работы с функцией **_STRING_TO_BUFFER** на языке CFC

См. также функцию _BUFFER_INSERT, которая смещает данные буфера, а не перезаписывает их.

25.6. BUFFER_COMP

Тип модуля: функция	Переменная	Тип	Описание
Входы	PT1	POINTER TO ARRAY [0..32767] OF BYTE	Указатель на буфер 1.
	SIZE1	INT	Размер буфера 1.
	PT2	POINTER TO ARRAY [0..32767] OF BYTE	Указатель на буфер 2.
	SIZE2	INT	Размер буфера 2.
	START	INT	Начальная позиция для поиска буфера 2 в буфере 1.
Выходы	BUFFER_COMP	INT	Начальная позиция буфера 2 в буфере 1.

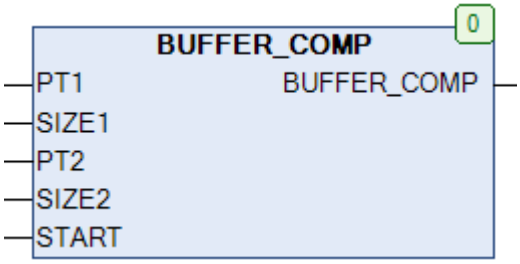


Рис. 25.11. Внешний вид функции **BUFFER_COMP** на языке CFC

Функция **BUFFER_COMP** производит поиск содержимого буфера, расположенного по адресу **PT2**, в буфере, расположенном по адресу **PT1**, начиная с элемента номер **START (**нулевому элементу буфера **1** соответствует **START=0**). Функция возвращает номер элемента буфера *1*, начиная с которого в нем было найдено первое вхождение буфера ***2***. Если буфер не был найден, то функция возвращает **-1**.

Поясним вышесказанное на примере. Пусть объявлены два буфера **abyBuffer1** и **abyBuffer2**:

```
abyBuffer1: ARRAY [0..9] OF BYTE:=[2, 12, 4, 8, 6, 2, 12, 4, 8, 6];
abyBuffer2: ARRAY [0..1] OF BYTE:=[12, 4];
```

Как можно заметить, содержимое буфера 2 два раза встречается в буфере 1: это элементы 1-2 и элементы 6-7. После вызова (рис. 25.12) функция вернет начальную позицию первого вхождения буфера 2 в буфер 1 – т.е. 1.

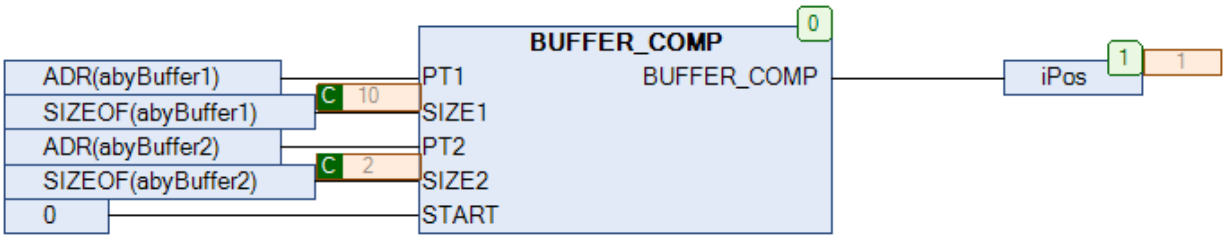


Рис. 25.12. Пример работы с функцией **BUFFER_COMP** на языке CFC

25.7. BUFFER_SEARCH

Тип модуля: функция	Переменная	Тип	Описание
Входы	PT	POINTER TO ARRAY [0..32767] OF BYTE	Указатель на буфер.
	SIZE	INT	Размер буфера.
	STR	STRING	Искомая строка.
	POS	INT	Начальная позиция для поиска.
	IGN	BOOL	Регистрозависимость поиска.
Выходы	BUFFER_SEARCH	BOOL	Начальная позиция найденной строки
Используемые модули	TO_UPPER		

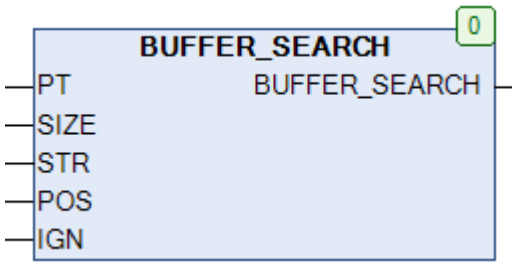


Рис. 25.13. Внешний вид функции **_BUFFER_SEARCH** на языке CFC

Функция **BUFFER_SEARCH** интерпретирует буфер как набор ASCII-кодов, и производит в нем поиск строки **STR** начиная с позиции **POS** (где **0** – позиция первого элемента буфера). Функция возвращает позицию первого вхождения строки в буфер. Если строка не найдена, то функция возвращает **-1**. Если вход **IGN** имеет значение **TRUE**, то поиск будет регистронезависимым, при этом искомая строка **STR** должна состоять только из заглавных букв (т.е. при STR='AB' в буфере также будет производиться поиск строк 'AB', 'ab', 'Ab' и 'aB'). Если вход **IGN** имеет значение **FALSE**, то поиск будет регистрозависимым.

Поясним вышесказанное на примере. Пусть объявлен буфер **abyBuffer**:

```
abyBuffer: ARRAY [0..9] OF BYTE:=[242, 229, 241, 242, 0, 210, 197, 209, 210, 0];
```

При этом 242/229/241/242 – это ASCII-коды (DEC) букв т/е/с/т, а 210/197/209/210 – это ASCII-коды (DEC) букв Т/Е/С/Т. На рис. 25.14 приведен пример работы функции в случае регистронезависимого (**IGN=TRUE**) и регистрозависимого (**IGN=FALSE**) поиска строки 'ТЕСТ' в данном буфере. Очевидно, что в первом случае функция вернет **0** (найдя строку 'тест', расположенную с 0-го элемента буфера), а во втором – **5** (найдя строку 'ТЕСТ', расположенную с 5-го элемента в буфера).

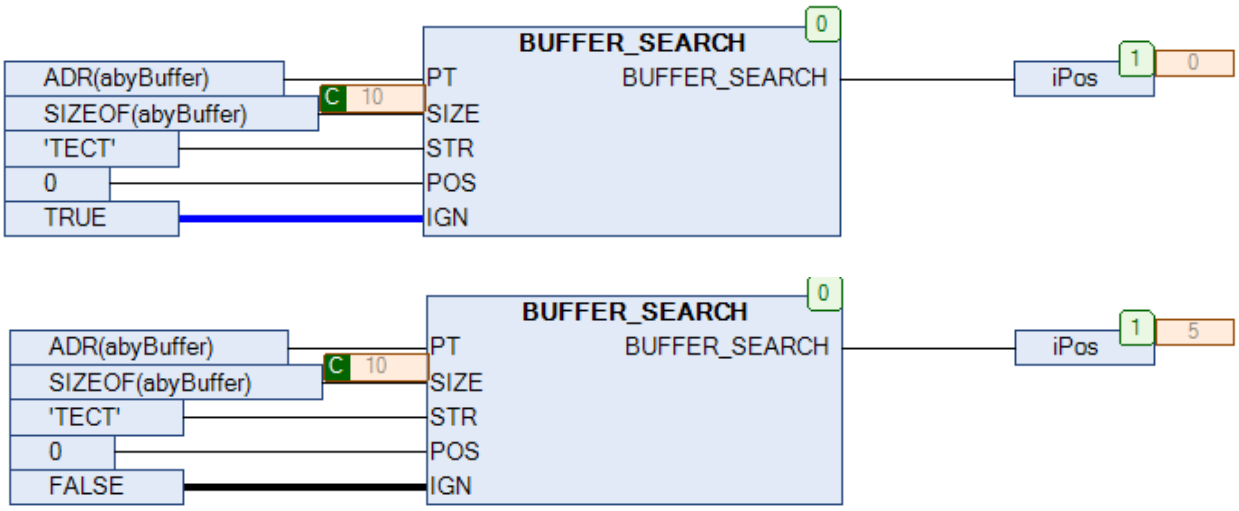


Рис. 25.14. Пример работы с функцией **BUFFER_SEARCH** на языке CFC

25.8. BUFFER_TO_STRING

Тип модуля: функция	Переменная	Тип	Описание
Входы	PT	POINTER TO ARRAY [0..32767] OF BYTE	Указатель на буфер.
	SIZE	UINT	Размер буфера.
	START	START	
	STOP	STOP	
Выходы	BUFFER_TO_STRING	STRING	Извлеченная строка.

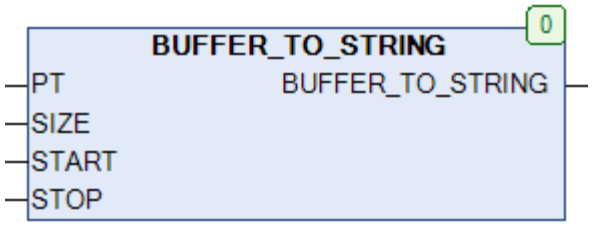


Рис. 25.15. Внешний вид функции **BUFFER_TO_STRING** на языке CFC

Функция **BUFFER_TO_STRING** интерпретирует буфер как набор ASCII-кодов, и возвращает текст, размещенный в нем с позиции **START** по позицию **STOP** в виде строки (где **0** – позиция первого элемента буфера).

Пример: пусть объявлен буфер [2, 12, 116, 101, 115, 116, 0, 0, 0, 0], где 116/101/115/116 – это ASCII-коды (DEC) букв t/e/s/t соответственно. После вызова функции (рис. 25.16) из буфера будет извлечена строка ‘test’.

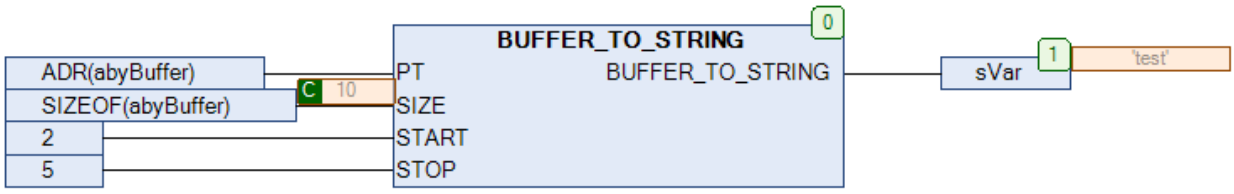


Рис. 25.16. Пример работы с функцией **BUFFER_TO_STRING** на языке CFC

См. также обратную функцию _STRING_TO_BUFFER, которая записывает строку в буфер.

26. Работа со списками

26.1. Вступление

Функции, описанные в данной главе, используются для операций над списками. Список представляет собой строковую переменную (типа **STRING**), содержащую записи, каждая из которых начинается со специального символа (этот символ определяется пользователем). Максимальный размер списка определяется глобальной константой **LIST_LENGTH**.

Функции обрабатывают списки через **VAR_IN_OUT** переменные (входы-выходы) – т.е. на вход функции подается исходный список, и на этот же вход возвращается результат его обработки. Такой подход позволяет обрабатывать списки путем прямых манипуляций с памятью ПЛК и является крайне эффективным, так как не требует копирования содержимого списков.

В рамках рассматриваемых примерах в качестве разделителя используется символ '**&**' (ASCII-код – 16#26).

Примеры списков:

- '&12&345' – список с двумя записями ('12' и '345');
- '' – список без записей;
- '&&&' – список с тремя пустыми записями;
- '&12&345&&ABC' – список с четырьмя записями ('12', '345', пустая запись, 'ABC').

Номера записей отсчитываются с единицы (т.е. первая запись списка имеет номер **1**).

26.2. LIST_ADD

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	SEP	BYTE	ASCII-код символа-разделителя.
	INS	STRING	Добавляемая запись.
Входы-выходы	LIST	STRING(LIST_LENGTH)	Список.
Выходы	LIST_ADD	BOOL	Флаг завершения операции.
Используемые модули	CHR_TO_STRING		

Рис. 26.1. Внешний вид функции **LIST_ADD** на языке CFC

Функция **LIST_ADD** добавляет в конец списка **LIST** с разделителем **SEP** запись **INS**. После выполнения операции выход функции принимает значение **TRUE**.

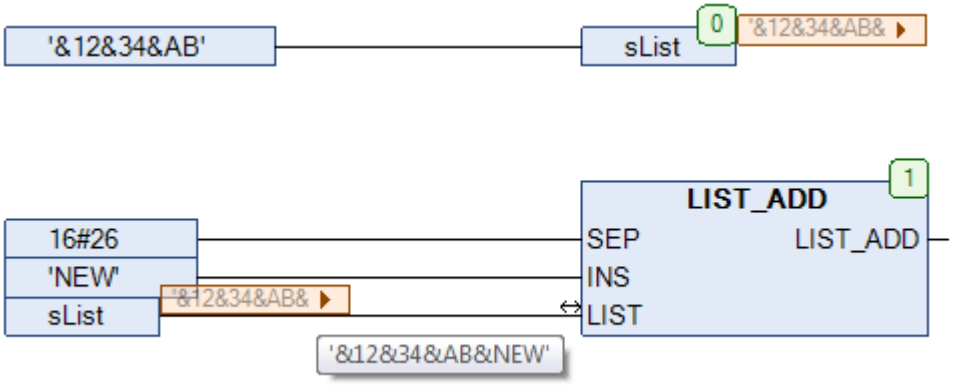


Рис. 26.2. Пример работы с функцией **LIST_ADD** на языке CFC

26.3. LIST_CLEAN

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	SEP	BYTE	ASCII-код символа-разделителя.
Входы-выходы	LIST	STRING(LIST_LENGTH)	Список.
Выходы	LIST_CLEAN	BOOL	Флаг завершения операции.

Рис. 26.3. Внешний вид функции **LIST_CLEAN** на языке CFC

Функция **LIST_CLEAN** очищает список от пустых записей. После выполнения операции выход функции принимает значение **TRUE**.

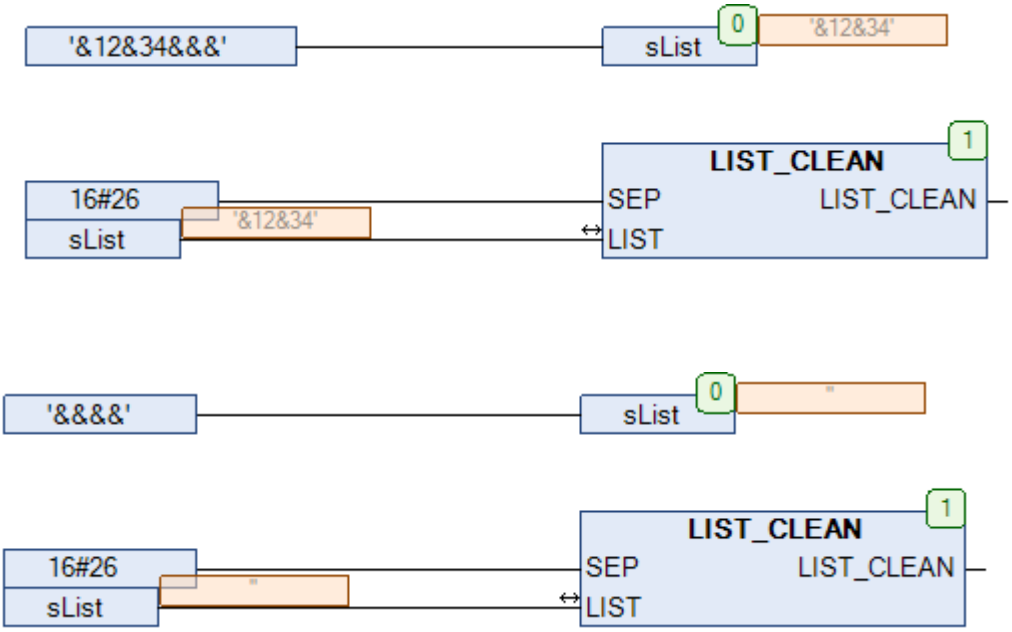


Рис. 26.4. Примеры работы с функцией **LIST_CLEAN** на языке CFC

26.4. LIST_GET

Тип модуля: функция	Переменная	Тип	Описание
Входы	SEP	BYTE	ASCII-код символа-разделителя.
	POS	INT	Номер читаемой записи.
Входы-выходы	LIST	STRING(LIST_LENGTH)	Список.
Выходы	LIST_GET	STRING(LIST_LENGTH)	Прочитанная запись.

Рис. 26.5. Внешний вид функции **LIST_GET** на языке CFC

Функция **LIST_GET** возвращает из списка **LIST** с разделителем **SEP** запись с номером **POS**. Нумерация записей начинается с **1**.

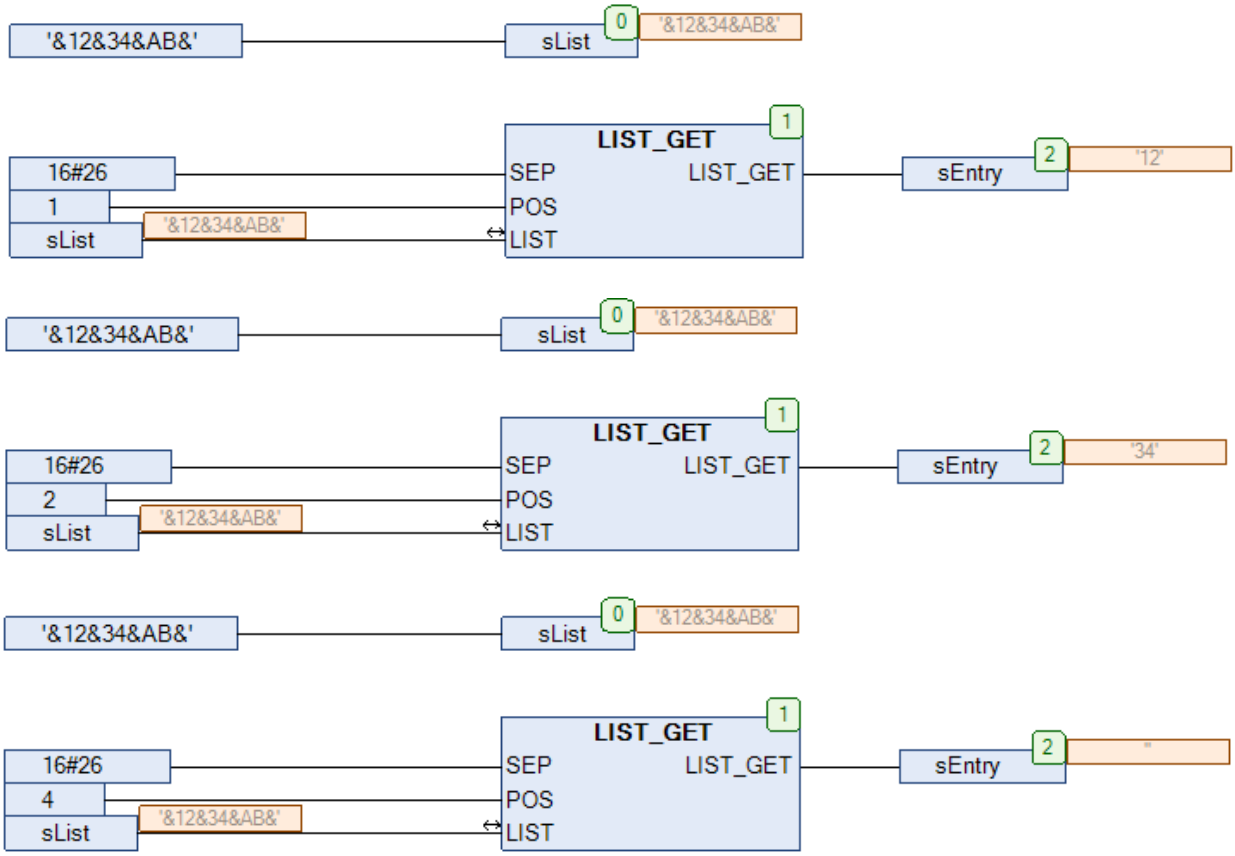


Рис. 26.6. Примеры работы с функцией **LIST_GET** на языке CFC

26.5. LIST_INSERT

Тип модуля: функция	Переменная	Тип	Описание
Входы	SEP	BYTE	ASCII-код символа-разделителя.
	POS	INT	Позиция для вставки записи.
	INS	STRING	Добавляемая запись.
Входы-выходы	LIST	STRING(LIST_LENGTH)	Список.
Выходы	LIST_INSERT	BOOL	Флаг завершения операции.
Используемые модули	CHR_TO_STRING		

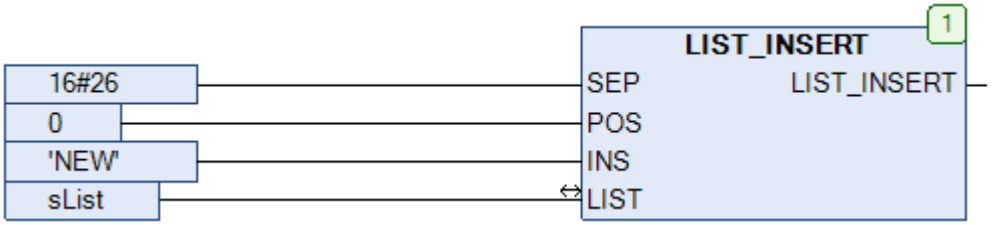
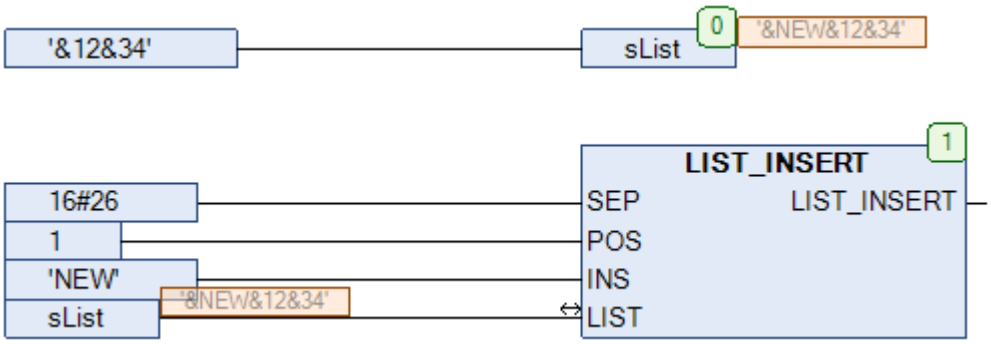


Рис. 26.7. Внешний вид функции **LIST_INSERT** на языке CFC

Функция **LIST_INSERT** добавляет в список **LIST** с разделителем **SEP** запись **INS**. Запись заменяет существующую запись номер **POS**, при этом существующая и все последующие записи смещаются. Если **POS=0**, то запись добавляется в начало списка (так же, как и при **POS=1**). Если значение **POS** превышает число существующих записей списка, то список дополняется пустыми записями.

После выполнения операции выход функции принимает значение **TRUE**.



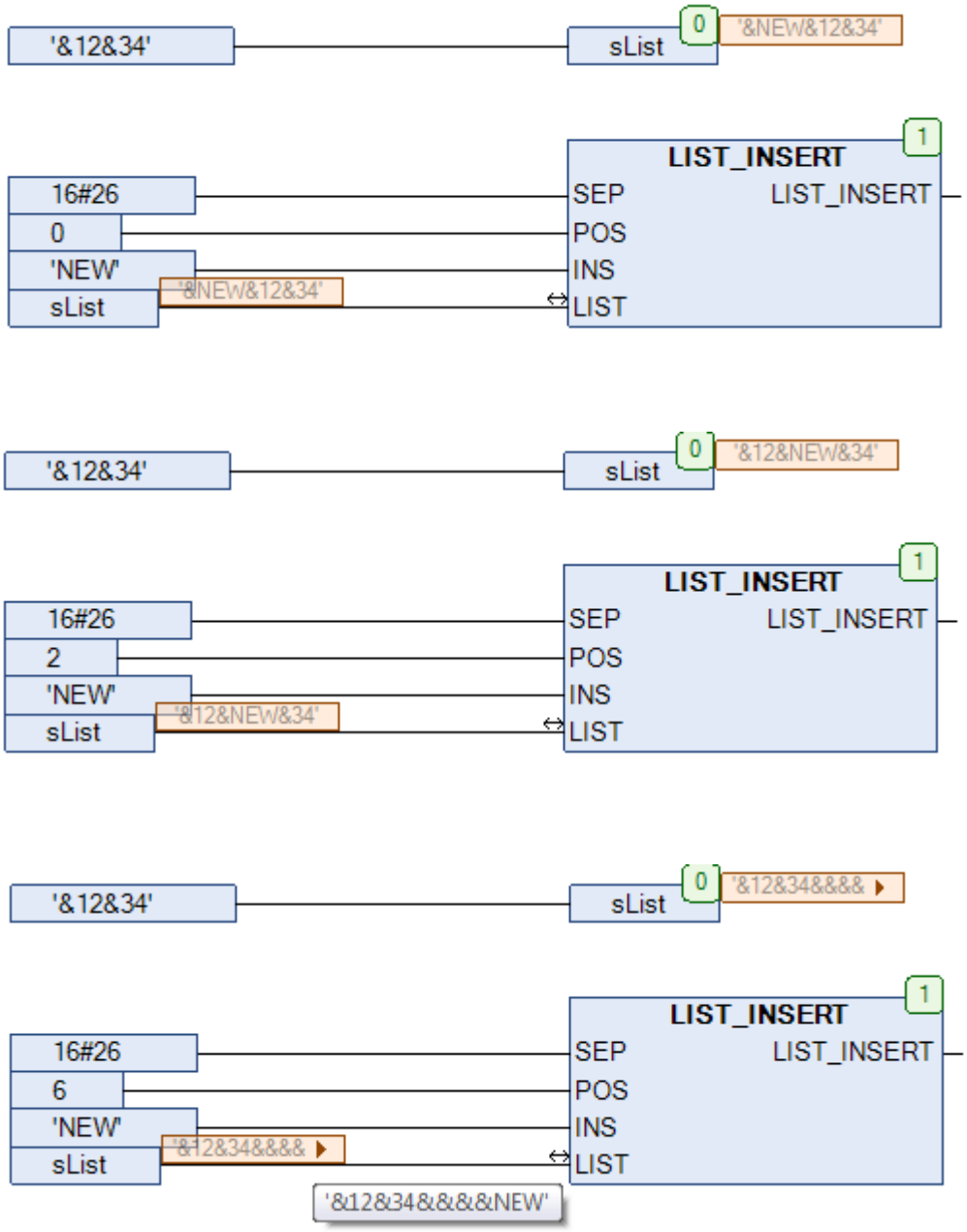


Рис. 26.8. Пример работы с функцией **LIST_INSERT** на языке CFC

26.6. LIST_LEN

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	SEP	BYTE	ASCII-код символа-разделителя.
Входы-выходы	LIST	STRING(LIST_LENGTH)	Список.
Выходы	LIST_LEN	INT	Число записей в списке.

Рис. 26.9. Внешний вид функции **LIST_LEN** на языке CFC

Функция **LIST_LEN** возвращает число записей в списке **LIST** с разделителем **SEP**.

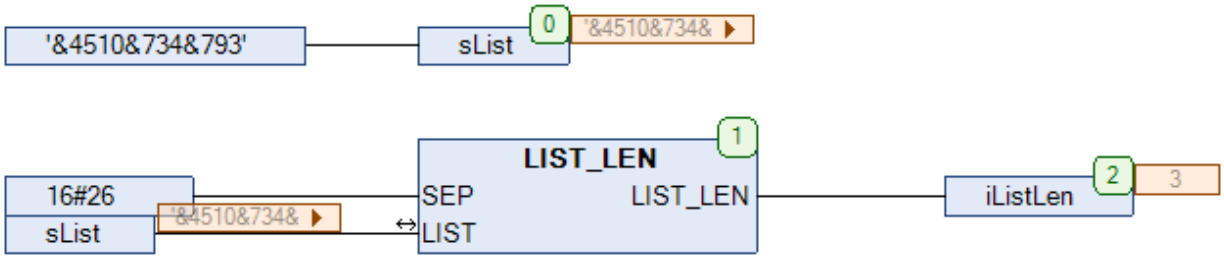


Рис. 26.10. Пример работы с функцией **LIST_LEN** на языке CFC

26.7. LIST_NEXT

Тип модуля: ФБ	Переменная	Тип	Описание
Входы	SEP	BYTE	ASCII-код символа-разделителя.
	RST	BOOL	Сигнал перезапуска блока.
Входы-выходы	LIST	STRING(LIST_LENGTH)	Список.
Выходы	LEL	STRING(LIST_LENGTH)	Запись списка.
	NUL	BOOL	Флаг «прочитаны все записи».

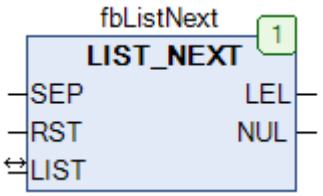


Рис. 26.11. Внешний вид ФБ **LIST_NEXT** на языке CFC

Функциональный блок **LIST_NEXT** используется для чтения записей из списка **LIST** с разделителем **SEP**. При каждом последующем вызове блока на выход **LEL** транслируется следующая запись списка. Когда прочитаны все записи (или в списке их нет), выход **NUL** принимает значение **TRUE**. По переднему фронту на входе **RST** происходит перезапуск блока.

```
VAR
    fbListNext:         LIST_NEXT;
    sList:              STRING(LIST_LENGTH) :=  '&34&&AB&1122&Test';
    asListElements:     ARRAY [1..10] OF STRING(LIST_LENGTH);
    iPos:               INT;
END_VAR
```

```
iPos := 0;
fbListNext(LIST := sList, SEP := 16#26);

WHILE NOT fbListNext.NUL AND iPos <= 10 DO
    asListElements[iPos] :=     fbListNext.LEL;
    fbListNext(list := sList);
    iPos                :=     iPos + 1;
END_WHILE;
```

Watch 1

Выражение	Приложение	Тип	Значение
PLC_PRG.asListElements	Device.Application	ARRAY [1..10] OF STRIN...	
asListElements[1]		STRING(LIST_LENGTH)	'34'
asListElements[2]		STRING(LIST_LENGTH)	''
asListElements[3]		STRING(LIST_LENGTH)	'AB'
asListElements[4]		STRING(LIST_LENGTH)	'1122'
asListElements[5]		STRING(LIST_LENGTH)	'Test'
asListElements[6]		STRING(LIST_LENGTH)	''
asListElements[7]		STRING(LIST_LENGTH)	''
asListElements[8]		STRING(LIST_LENGTH)	''
asListElements[9]		STRING(LIST_LENGTH)	''
asListElements[10]		STRING(LIST_LENGTH)	''

Рис. 26.12. Пример работы с ФБ **LIST_NEXT** на языке ST

26.8. LIST_RETRIEVE

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	SEP	BYTE	ASCII-код символа-разделителя.
	POS	INT	Номер вырезаемой записи.
Входы-выходы	LIST	STRING(LIST_LENGTH)	Список.
Выходы	LIST_RETRIEVE	STRING(LIST_LENGTH)	Вырезанная запись.

Рис. 26.13. Внешний вид функции **LIST_RETRIEVE** на языке CFC

Функция **LIST_RETRIEVE** вырезает из списка **LIST** с разделителем **SEP** запись с номером **POS**. Нумерация запись начинается с **1**. Если элемент с данным номером отсутствует, то функция возвращает пустую строку. См. также функцию LIST_RETRIEVE_LAST, которая используется для вырезания последней записи списка.

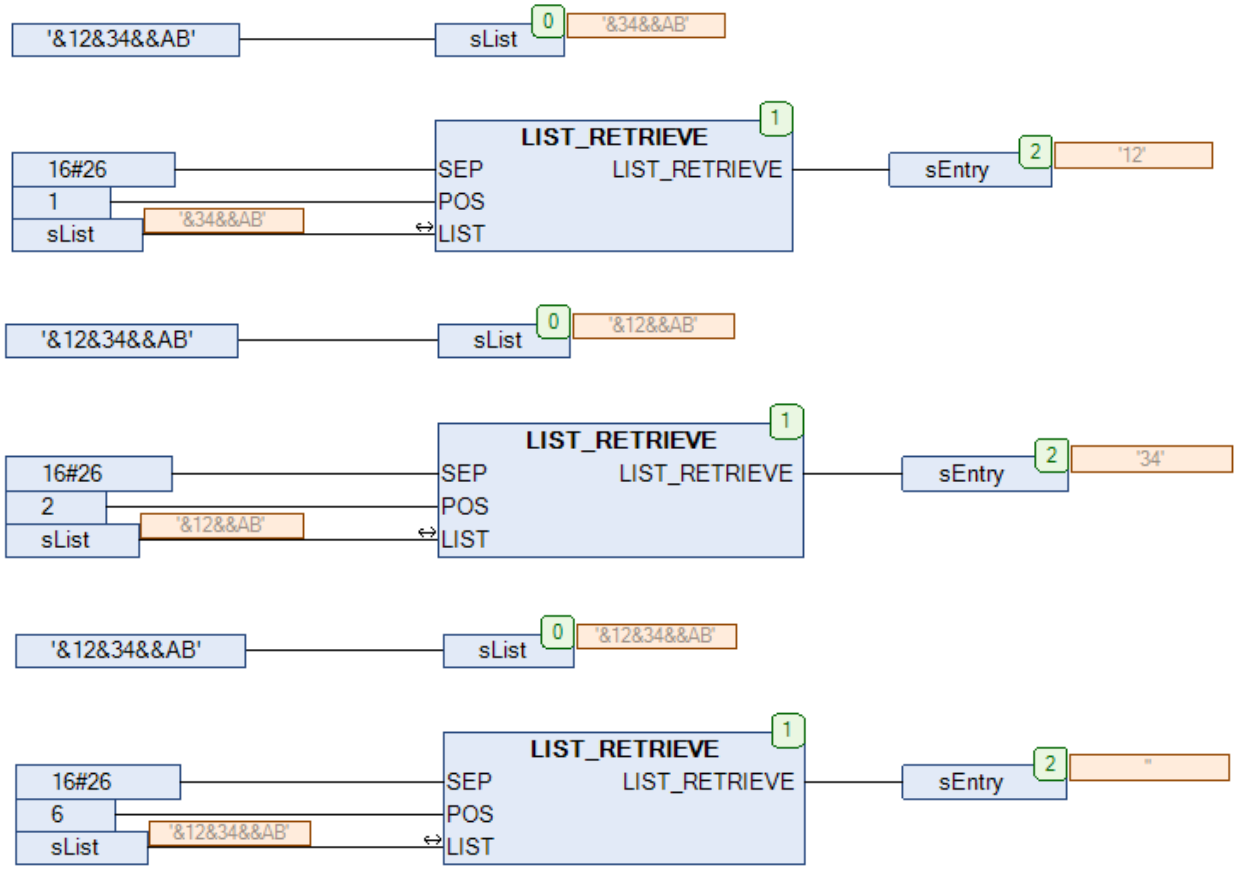


Рис. 26.14. Примеры работы с функцией **LIST_RETRIEVE** на языке CFC

26.9. LIST_RETRIEVE_LAST

Тип модуля: функция	**Переменная**	**Тип**	**Описание**
Входы	SEP	BYTE	ASCII-код символа-разделителя.
Входы-выходы	LIST	STRING(LIST_LENGTH)	Список.
Выходы	LIST_RETRIEVE	STRING(LIST_LENGTH)	Вырезанная запись.

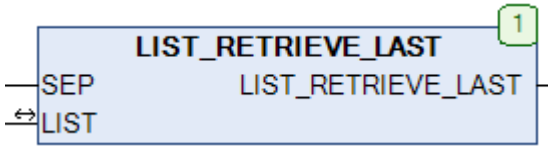


Рис. 26.15. Внешний вид функции **LIST_RETRIEVE_LAST** на языке CFC

Функция **LIST_RETRIEVE_LAST** вырезает из списка **LIST** с разделителем **SEP** последнюю запись. См. также функцию LIST_RETRIEVE, которая используется для вырезания заданной записи списка.

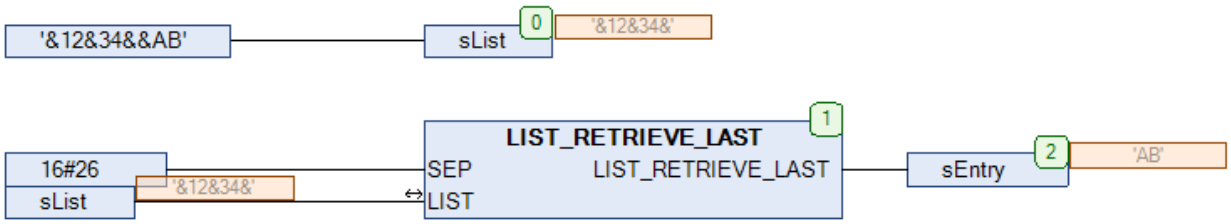


Рис. 26.16. Пример работы с функцией **LIST_RETRIEVE_LAST** на языке CFC

История версий

№ п/п	Автор замечания	Описание
v0.9 (17.10.2017)		
1	ЕК, ЕЧ, ВБ	Собрана предварительная версия документа.
v1.0 (01.12.2017)		
2	ЕК, ЕЧ	Собрана релизная версия после вычитки и оформления.