Unimod Pro

РУКОВОДСТВО ПО ПРОГРАММИРОВАНИЮ

ВВЕДЕНИЕ

АО „ТРЭИ” постоянно совершенствует и развивает свою продукцию. В связи с этим информация, содержащаяся в данном документе, может изменяться без дополнительного предупреждения пользователей.

Россия,
440028, Пенза, ул. Титова, 1Г
Телефон (fax): +7 (8412) 49-95-39
e-mail: tr-penza@trei.biz

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Пакет программ **Unimod Pro** предназначен для создания технологических приложений, исполняющихся на модулях фирмы **TREI**. Unimod Pro - это инструментальная CASE-система для программирования логических контроллеров (PLC), основанная на языках программирования стандарта **IEC 1131-3**. Основным языком для написания приложений в Unimod Pro является язык ST (структурированный текст). Это структурный язык высокого уровня, разработанный для процессов автоматизации. Этот язык, в основном, используется для создания сложных процедур, которые не могут быть легко выражены при помощи графических языков. В связи с особенностями работы модулей фирмы **TREI,** язык ST был модифицирован. Упростился синтаксис некоторых стандартных операторов, были добавлены новые операторы не входящие в спецификацию ST.

Документ предназначен для разработчиков программного обеспечения, а также для проектировщиков систем контроля и управления.

При работе используйте документацию следующих фирм:

TREI

- «Unimod Pro. Руководство пользователя»
- «Устройство программного управления TREI-5B-05. Руководство по эксплуатации»
- «Unimod Pro. Исполнительная система»

1. АРХИТЕКТУРА ПРОЕКТА

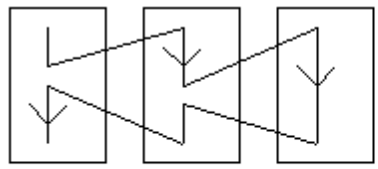
Проект в Unimod Pro состоит из следующих частей: словари переменных, взаимосвязи (привязки) переменных с каналами ввода/вывода и программных блоков. Программные блоки, входящие в проект, могут быть написаны на языке ST, FBD, LD или SFC. Все программные блоки делятся на четыре вида: программы, функции, дочерние программы и функциональные блоки. Проект может не содержать программ. Функциональные блоки в Unimod Pro, реализованные на целевой платформе и не могут быть модифицированы. Программные блоки в проекте должны взаимодействовать между собой по определенным правилам.

1.1 Программы

Программа - это логическая программируемая единица, которая описывает операции с **переменными** процесса. Все программы в Unimod Pro являются циклическими, т.е. выполняются на каждом цикле целевой системы. Программы проекта не являются связанными между собой. Одна программа проекта не может вызывать другую программу проекта. Все программы в каждом цикле целевой системы выполняются последовательно, одна за другой. Порядок выполнения программ определяется пользователем. Любая программа проекта может вызывать любую функцию или функциональный блок проекта или функции и функциональные блоки, реализованные на целевой системе. Любая программа проекта может вызывать только свои собственные дочерние программы.

1.2 Функции (подпрограммы)

Исполнение функций управляется вызвавшей их программой или другой функцией. Исполнение вызвавшей программы или функции приостанавливается до тех пор, пока не закончит свою работу вызванная ими подпрограмма:



Любая функция может вызывать одну или несколько подпрограмм. Подпрограмма может иметь только локальные переменные.

Для интеллектуальных модулей переменные функции хранятся во временной области памяти. После завершения подпрограммы, память под выделенные ей переменные освобождается и становится доступна для следующей подпрограммы. Каждая такая подпрограмма может вызывать сама себя (рекурсивный вызов).

<u>Предупреждение:</u> В случае большой вложенности подпрограмм или рекурсивных вызовов может возникнуть ошибка переполнения стека.

Для мастер-модулей (Мастер-ПК и Мастер-М911Е), память под переменные функции выделяется статически, вследствие чего, рекурсивный вызов функций запрещён.

<u>Предупреждение:</u> Подпрограммы не “запоминают” локальных значений своих локальных переменных и не могут вызывать функциональные блоки.

Интерфейс подпрограммы должен быть определен явно, с типами и уникальными именами (в пределах области функции) каждого вызываемого и возвращаемого параметра. Для того чтобы поддержать стандарт языка **ST,** возвращаемый параметр должен иметь то же имя, что и подпрограмма.

Следующий пример показывает, как установить значение возвращаемого параметра в теле подпрограммы:

```
ST:     (*присвоить значение возвращаемому параметру, используя его имя
        то же имя что и у подпрограммы*)
        subprog_name := <expression>;
```

Каждая функция имеет только одно возвращаемое значение равное имени функции:

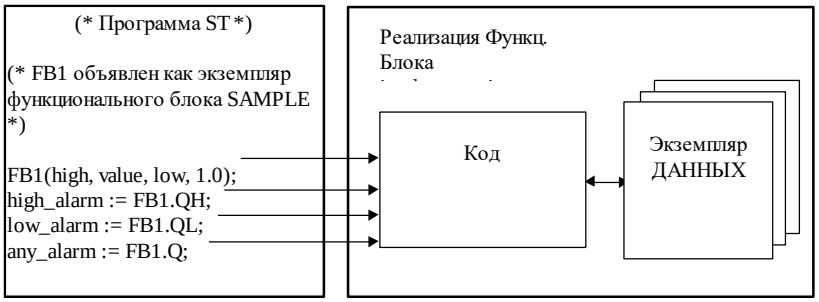
```
ST:     (*присвоить значение переменной mymax значение функции max*)
        Mymax :=  max(1,5);
```

1.3 Дочерние программы

Исполнение дочерних программ управляется вызвавшей их родительской программой или другой дочерней программой, являющейся родительской по отношению к ним. Исполнение вызвавшей программы или дочерней программы приостанавливается до тех пор, пока не закончит свою работу вызванная ими дочерняя программа.
Любая программа на мастер-модуле может иметь дочерние программы. В свою очередь, любая дочерняя программа может иметь свои собственные дочерние программы. Для описания дочерней программы может быть использован любой язык, кроме SFC. Дочерняя программа может иметь собственный локальный словарь, в то же время в ней могут использоваться локальные переменные родительской программы. Вызывать дочернюю программу может только её родительская программа.
Фактически, дочерние программы совмещают в себе свойства программ и функций. Как и функции, они должны возвращать значение и могут принимать список параметров. Однако, как и программы, они могут иметь собственные дочерние программы и вызывать их.
Максимальное суммарное число функций и дочерних программ для каждого модуля – 255.

1.4 Функциональные блоки

Исполнение функциональных блоков управляется вызвавшей их программой. Исполнение вызвавшей программы приостанавливается до тех пор, пока не закончит свою работу вызванный функциональный блок. Локальные переменные функциональных блоков копируются для каждого экземпляра. Когда программа вызывает блок, на самом деле, вызывается экземпляр блока: вызывается тот же код, но используются данные, захваченные специально для данного экземпляра блока. Значения переменных экземпляра передаются от одного цикла к другому.

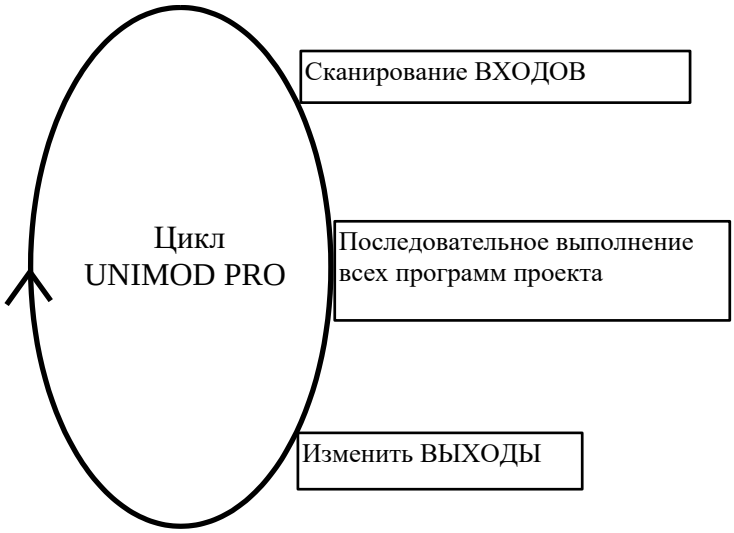


В отличие от функции, функциональный блок может возвращать несколько значений. Доступ к выходным значениям организуется через выходные переменные для данного типа блока.

1.5 Правила исполнения

В Unimod Pro все операции происходят по циклам. Время цикла может иметь различное значение в зависимости от сложности приложения и количества объявленных в проекте юнитов/мезонин. По окончании текущего цикла, сразу начинается следующий цикл.

Основные операции в течение временного цикла:



Это позволяет системе:
- гарантировать, что входные переменные сохраняют свое значение в течение временного цикла,
- гарантировать, что устройства вывода изменяются не более одного раза в течение временного цикла,
- правильно работать с одними и теми же глобальными переменными из различных программ.

2. ОСНОВНЫЕ ПОНЯТИЯ

2.1 Основные типы

Любая константа, выражение или переменная, используемая в программе должна характеризоваться своим типом. Вот основные типы программных объектов:

BOOLEAN: булевская величина
INTEGER: целая величина
REAL: вещественная величина
TIMER: таймерная величина
MESSAGE: сообщение/строка

Замечание: ***Переменные типа TIMER (таймерные) содержат значения не большие, чем сутки, и не могут быть использованы для записи данных. Переменные типа MESSAGE (сообщения) используются только для мастер-модуля.***

2.2 Константы

Константы относятся к одному типу. Одно и то же обозначение не может быть использовано для представления констант разных типов.

Логические константы
Существует только две логические константы:
TRUE эквивалентно целому значению 1
FALSE эквивалентно целому значению 0
Для написания ключевых слов **«true»** и **«false»** можно использовать, как заглавные буквы, так и прописные.

Целые аналоговые константы
Целые константы представляются знаковыми длинными целыми (32 бита) величинами:
от **-2147483647** до **+2147483647**
Целые аналоговые константы могут быть представлены с одной из следующих **баз**. Целые константы должны начинаться с **префикса**, который определяет используемую базу:

База	Префикс	Пример
DECIMAL		**-908**
HEXADECIMAL	**"16#"**	**16#1A2B3C4D**
OCTAL	**"8#"**	**8#1756402**
BINARY	**"2#"**	**2#1101_0001_0101_1101**

Символ подчеркивания (’_’) может быть использован для того, чтобы разделить группы цифр. Он не имеет особенного значения и используется для улучшения читаемости констант.

Вещественные аналоговые константы
Действительные аналоговые константы могут быть записаны в **десятичном** или **научном** представлении. **Десятичная точка** (‘**.**’) разделяет целую и десятичную части. Десятичную точку нужно использовать, для того чтобы отличить действительную константу от целой. Научное представление использует буквы «**E»** или «**F»** для того, чтобы отделить **мантиссу** от **экспоненты**. Экспоненциальная часть в научном представлении должна быть знаковой целой величиной от -**37** до **+37**.

Примеры вещественных констант:
0.23
1.5E-3
Выражение 546 не является действительной константой, правильно будет писать 546.0

Таймерные константы

Таймерные константы представляют временные значения от **0** секунд до **23h59m59s999ms**. Наименьшая допустимая единица - это миллисекунда. Стандартные временные единицы, использующиеся в константах это:

Час	буква **h** должна следовать за количеством часов
Минута	буква **m** должна следовать за количеством минут
Секунда	буква **s** должна следовать за количеством секунд
Миллисекунда	буквы **ms** должны следовать за количеством миллисекунд

Таймерные константы должны начинаться с «**T#**» или «**TIME#**». Для написания ключевых слов можно использовать как заглавные, так и прописные латинские буквы. Некоторые единицы могут быть опущены. Вот примеры таймерных констант:

T#1H450MS 1 час, 450 миллисекунд

time#1H3M 1 час, 3 минуты

Выражение «0» представляет целую константу, а не таймерное значение.

Строковые константы

Строковые константы заключаются в кавычки 'message'. Максимальная длина строки 128 символов.

Предупреждение: символ кавычка (’) не может быть использован внутри строки-константы. Строка-константа должна быть записана в одной строке исходного текста программы

Пустая строка-константа представляется двумя кавычками без пробела или табуляции внутри: ‘ ’

Для представления непечатных символов используется знак **$**:

$$ - знак доллара $

$' - одиночная кавычка

$l или **$L** – строка вверх

$n или **$N** - новая строка

$p или **$P** - пропуск страницы

$r или **$R** - возврат каретки

$t или **$T** - табуляция

$" - двойные кавычки

$XX - где XX шестнадцатеричные цифры, код ASCII символа

Примеры констант-сообщений:

'Hello'

'RL'

2.3 Переменные

Переменные программы могут быть **локальными** или **глобальными**. Локальные переменные могут использоваться только одной программой. Глобальные переменные могут использоваться любой программой проекта. Имена переменных должны удовлетворять следующим правилам:

имя не может быть длиннее **22** символов

первым символом должна быть **латинская буква**

последующими символами могут быть **буквы, цифры** или символ подчеркивания

Предупреждение: буквы должны использоваться латинские, цифры – арабские

2.4 Зарезервированные ключевые слова

Ниже представлен список зарезервированных ключевых слов. Такие идентификаторы не могут быть использованы в качестве имен программ, переменных функциональных блоков.

A	ANA, ABS, ACOS, ADD, ANA, AND, AND_MASK, ANDN, ARRAY, ASIN, AT, ATAN,
B	BCD_TO_BOOL, BCD_TO_INT, BCD_TO_REAL, BCD_TO_STRING, BCD_TO_TIME, BOO, BOOL, BOOL_TO_BCD, BOOL_TO_INT, BOOL_TO_REAL, BOOL_TO_STRING, BOOL_TO_TIME, BY, BYTE,
C	CAL, CALC, CALCN, CALN, CALNC, CASE, CONCAT, CONSTANT, COS,
D	DATE, DATE_AND_TIME, DELETE, DINT, DIV, DO, DT, DWORD,
E	ELSE, ELSIF, EN, END_CASE, END_FOR, END_FUNCTION, END_IF, END_PROGRAM, END_REPEAT, END_RESSOURCE, END_STRUCT, END_TYPE, END_VAR, END_WHILE, ENO, EQ, EXIT, EXP, EXPT,
F	FALSE, FEDGE, FIND, FOR, FUNCTION,
G	GE, GFREEZE, GKILL, GRST, GSTART, GSTATUS, GT,
I	IF, INSERT, INT, INT_TO_BCD, INT_TO_BOOL, INT_TO_REAL, INT_TO_STRING, INT_TO_TIME,

J	JMP, JMPC, JMPCN, JMPN, JMPNC,
L	LD, LDN, LE, LEFT, LEN, LIMIT, LINT, LN, LOG, LREAL, LT, LWORD, LTERMO1, LTERMO2, LTERMO3, LTERMO4, LTERMO5
M	MAX, MID, MIN, MOD, MOVE, MSG, MUL, MUX,
N	NE, NOT,
O	OF, ON, OPERATE, OR, OR_MASK, ORN,
P	PROGRAM
R	R, REDGE, READ_ONLY, READ_WRITE, REAL, REAL_TO_BCD, REAL_TO_BOOL, REAL_TO_INT, REAL_TO_STRING, REAL_TO_TIME, REDGE, REPEAT, REPLACE, RESSOURCE, RET, RETAIN, RETC, RETCN, RETN, RETNC, RETURN, RIGHT, ROL, ROR,
S	S, SEL, SHL, SHR, SIN, SINT, SQRT, ST, STN, STRING, STRING_TO_BCD, STRING_TO_BOOL, STRING_TO_INT, STRING_TO_REAL, STRING_TO_TIME, STRUCT, SUB, SYS_ERR_READ, SYS_ERR_TEST, SYS_INITALL, SYS_INITANA, SYS_INITBOO, SYS_INITTMR, SYS_RESTALL, SYS_RESTANA, SYS_RESTBOO, SYS_RESTTMR, SYS_SAVALL, SYS_SAVANA, SYS_SAVBOO, SYS_SAVTMR, SYS_TALLOWED, SYS_TCURRENT, SYS_TMAXIMUM, SYS_TOVERFLOW, SYS_TRESET, SYS_TWRITE, SYSTEM,
T	TAN, TASK, THEN, TIME, TIME_OF_DAY, TIME_TO_BCD, TIME_TO_BOOL, TIME_TO_INT, TIME_TO_REAL, TIME_TO_STRING, TMR, TO, TOD, TRUE, TSTART, TSTOP, TYPE,
U	UDINT, UINT, ULINT, UNTIL, USINT,
V	VAR, VAR_ACCESS, VAR_EXTERNAL, VAR_GLOBAL, VAR_IN_OUT, VAR_INPUT, VAR_OUTPUT,
W	WHILE, WITH, WORD,
X	XOR, XOR_MASK, XORN

2.5 Булевские переменные

Термин «булевские» означает **логические**. Такие переменные могут принимать одно из двух булевых значений: **TRUE (ИСТИНА)** или **FALSE (ЛОЖЬ)**. Обычно, булевские переменные используются в логических выражениях. Булевские переменные могут иметь один из следующих **атрибутов**:

Внутренняя: переменная, хранящаяся в памяти, изменяемая программой
Константа: неизменяемая переменная, хранящаяся в памяти, с начальным значением
Вход: переменная, связанная с устройством ввода (обновляется системой)
Выход: переменная, связанная с устройством вывода
Стековая: переменная, теряющая значение по окончанию программы, хранится в стеке системы

2.6 Аналоговые переменные

Аналоговые означает - **непрерывные**. Такие переменные могут принимать значения знаковых целых или знаковых вещественных чисел с плавающей запятой. Имеют следующий формат:

BYTE 8 битовое беззнаковое целое: от **0** до **+255**
INTEGER 32 битовое знаковое целое: от **-2147483647** до **+2147483647**
REAL 32 битовое знаковое число с плавающей запятой: **+/- 3.4E +/- 38**
DOUBLE 64 битовое знаковое число с плавающей запятой: **+/- 1.7E +/- 308**

Аналоговые переменные могут иметь один из следующих атрибутов:
Внутренняя: переменная, хранящаяся в памяти, изменяемая программой
Константа: неизменяемая переменная, хранящаяся в памяти, с начальным значением
Вход: переменная, связанная с устройством ввода (обновляется системой)
Выход: переменная, связанная с устройством вывода
Стековая: переменная, теряющая значение по окончанию программы, хранится в стеке системы

2.7 Таймерные переменные

Термин «таймерные» означает **часы** и **счетчики**. Такие переменные могут принимать значения, выраженные в единицах времени и, обычно, используются в таймерных выражениях. Значение таймерной переменной не может превышать **23h59m59s999ms** и не может быть отрицательным. Таймерная переменная хранится в 32 битовом слове. Его внутреннее представление - положительное число миллисекунд.

Таймерные переменные могут иметь один из следующих **атрибутов**:
Внутренняя: переменная, хранящаяся в памяти, изменяемая программой
Константа: неизменяемая переменная, хранящаяся в памяти, с начальным значением
Стековая: переменная, теряющая значение по окончанию программы, хранится в стеке системы

Примечание: Таймерная переменная не может иметь атрибутов **«ВХОД»** или **«ВЫХОД»**.

Таймерные переменные автоматически обновляются Unimod Pro. Когда таймерная переменная активна, ее значение автоматически увеличивается в соответствии с часами реального времени целевой задачи. Для управления таймерными переменными используются следующие операторы языка **ST:**

TSTART() - запустить автоматическое обновление переменной (таймера)
TSTOP() - остановить автоматическое обновление переменной (таймера)

2.8 Строковые переменные

Сообщения или строковые переменные содержат символьные строки. Длина строки может меняться в процессе работы. Длина строковой переменной не может превосходить значения определенного при объявлении переменной. Максимальная длина строки ограничена 128 символами.

Строковые переменные могут иметь один из следующих **атрибутов**:

Внутренняя: переменная, хранящаяся в памяти, изменяемая программой
Константа: неизменяемая переменная, хранящаяся в памяти, с начальным значением

Примечание: ***Строковая переменная не может иметь атрибутов «ВХОД» или «ВЫХОД».***

2.9 Структурные переменные

В Unimod Pro поддерживает два вида структурных переменных: массивы и структуры.

Массив – это расположенные последовательно друг за другом в памяти элементы одного и того же простого типа (boolean, integer, real, timer, message). Каждый массив имеет имя. Доступ к элементам массива осуществляется по имени массива и индексу (порядковому номеру) элемента.

Unimod Pro поддерживает многомерные массивы.

Предупреждение: ***Использование массивов возможно только на языке ST. Выход индекса за память массива приведёт к неправильной работе системы.***

Структуры – это объединение одной и более переменных, возможно разных типов, имеющее для простоты одно имя. Отдельные составляющие её переменные, называются полями.

Предупреждение: ***Структурные переменные поддерживаются только мастер-модулями. Массив не может быть входом или выходом. Структура может описывать только переменные обмена на интеллектуальных модулях.***

2.10 Комментарии

Комментарии могут быть введены в язык ST. Должны начинаться со специального символа «**(***» и заканчиваться символом «***)***». Комментарии могут быть введены в любом месте программы **ST** и могут занимать несколько строк.

Например:

```
counter := ivalue;   (* присвоить значение основному счетчику *)
     (* это комментарий
     на двух строках *)
     c := counter (* комментарий можно расположить где угодно *) + base_value + 1;
```

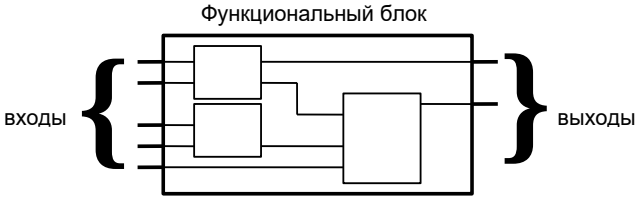
Комментарии могут частично перекрываться. Это означает, что символ «(*» может быть использован внутри комментария, а символ «*)» **- нет**.

3. ЯЗЫК ФУНКЦИОНАЛЬНЫХ БЛОЧНЫХ ДИАГРАММ (FBD)

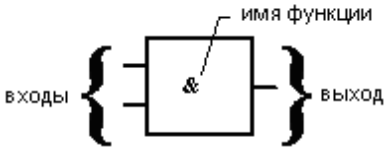
Язык **функциональных блочных диаграмм (FBD) -** графический язык. Он позволяет программисту строить сложные процедуры, используя существующие **функции** из библиотеки Unimod Pro и **связывая** их вместе при помощи графических диаграмм.

3.1 Основной формат диаграмм FBD

FBD диаграмма описывает функцию между **входными переменными** и **выходными переменными**. Функция описывается как множество **элементарных функциональных блоков**. Входные и выходные переменные связываются в блоки при помощи линий связи. Выход функционального блока может быть также связан с входом другого функционального блока.



Вся функция FBD программы построена из стандартных **элементарных** функциональных блоков из библиотеки Unimod Pro. Каждый функциональный блок имеет фиксированное **количество входных точек связи** и фиксированное количество **выходных точек связи**. Функциональный блок представляется одиночным **прямоугольником**. Входы соединяются с **левым** краем. Выходы соединяются с **правым** краем. Элементарный функциональный блок реализует одну функцию между входами и выходами. Имя функции, реализуемой блоком, пишется на символе прямоугольника. Каждый вход или выход блока имеют определенный тип.



Входные переменные FBD программы должны быть связаны с точками входа функционального блока. Тип каждой переменной должен быть тем же что и тип соответствующего входа. Входом FBD блока может быть **константа**, любая **внутренняя**, **входная** или **выходная** переменная.

Выходные переменные FBD программы должны быть связаны с точками выхода функционального блока. Тип каждой переменной должен быть тем же что и тип соответствующего выхода. Выходом FBD блока может быть внутренняя или выходная переменная или имя программы (только для подпрограмм). Когда выходом является имя редактируемой подпрограммы, оно представляет присвоение возвращаемого значения подпрограммы (возвращаемого в вызывающую программу). Входные и выходные переменные, входы и выходы функциональных блоков соединены линиями связи. Линии могут быть использованы для соединения двух логических точек диаграммы:

- Входной переменной и входа функционального блока
- Выхода функционального блока и входа другого блока
- Выхода функционального блока и выходной переменной

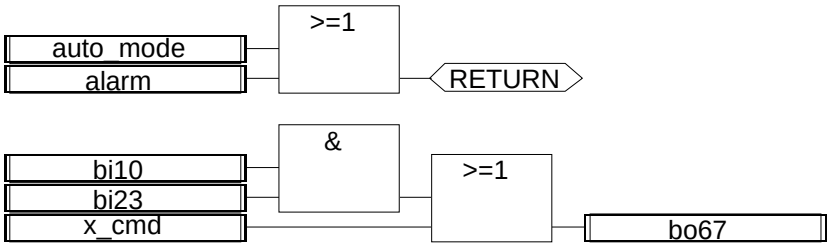
Связи ориентированы, это означает, что данные передаются с левого конца к правому. Левый и правый концы связи должны быть **одного типа**.

Множественные правые связи могут быть использованы, чтобы передать информацию от его левого конца к каждому правому концу. Все концы связи должны быть одного типа.

3.2 Оператор RETURN

Ключевое слово **<RETURN>** может быть выходом диаграммы. Оно должно быть связано с логическим выходом функционального блока. Оператор **<RETURN>** представляет собой условное завершение программы: если выход блока связанного с оператором имеет тип **TRUE,** остальная часть диаграммы не выполняется.

(* Пример FBD программы использующей оператор RETURN *)




```
(* ST эквивалент: *)
If auto_mode OR alarm Then
        Return;
End_if;
bo67 := (bi10 AND bi23) OR x_cmd;
```

3.3 Прыжки и метки

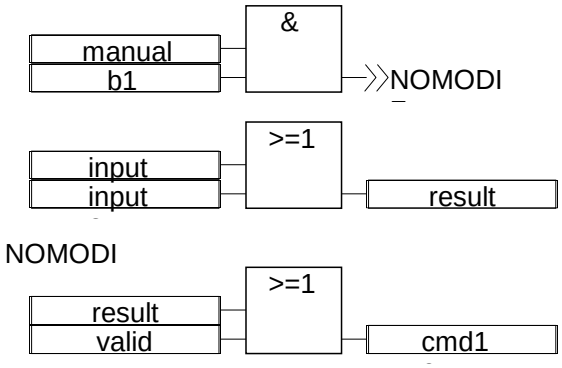
Прыжки и метки используются для управления выполнением диаграммы. К правому краю символа метки или прыжка не может быть присоединено никаких других объектов. Используются следующие обозначения:

>>LAB .....прыжок на метку (имя метки "LAB")
LAB: .....определение метки (имя метки "LAB")

Примечание. В имени метки должны использоваться только латинские буквы

Если линия связи слева от символа прыжка находится в состоянии **TRUE**, исполнение программы переходит на соответствующую метку.

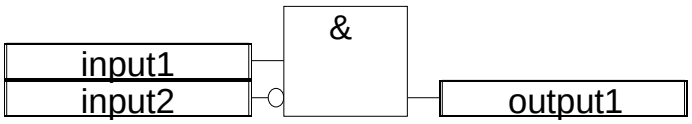
(* Пример FBD программы с использованием меток и прыжков *)



3.4 Логическое отрицание

Одиночная линия связи с правым концом, присоединенным к входу функционального блока, может заканчиваться **логическим отрицанием**. Отрицание представляется маленьким кружком. Когда используется логическое отрицание, левый и правый концы линии связи должны иметь тип **BOOLEAN.**

(* Пример FBD программы использующей логическое отрицание *)

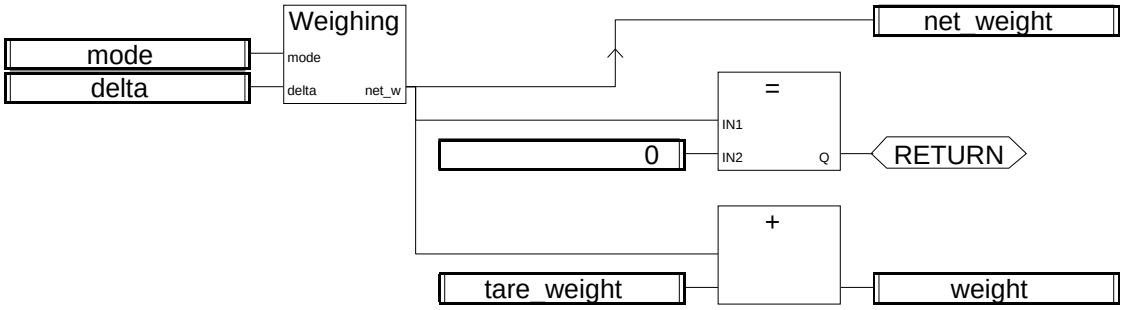



```
(* ST эквивалент: *)
output1 := input1 AND (NOT (input2));
```

3.5 Вызов функций и функциональных блоков из FBD

Язык FBD позволяет вызывать подпрограммы, функции и функциональные блоки. Подпрограммы, функции и функциональные блоки представляются прямоугольником функции. Имя, написанное в прямоугольнике, это имя подпрограммы, функции или функционального блока. В случае подпрограммы или функции единственным выходом прямоугольника является возвращаемая величина. Функциональные блоки могут иметь более одного выхода.

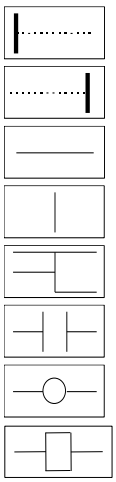
(* Пример FBD программы использующей блок SUB PROGRAM *)




```
(* ST Эквивалент *)
net_weight := Weighing (mode, delta); (*вызов подпрограммы*)
If (net_weight = 0) Then
                        Return;
End_if;
weight := net_weight + tare_weight;
```

4. ЯЗЫК РЕЛЕЙНЫХ ДИАГРАММ (LD)

Язык релейных диаграмм - это графическое представление логических уравнений, комбинирующее **контакты** (входы) и **витки** (выходы). Язык LD позволяет описывать работу с **булевскими** данными, помещая **графические символы** в схему программы. Графические символы LD организованы внутри схемы так же, как электрическая схема. Справа и слева LD диаграмма должна соединяться с вертикальными силовыми рельсами. Ниже представлены основные компоненты LD диаграммы.



Левая вертикальная силовая шина

Правая вертикальная силовая шина

Горизонтальная линия связи

Вертикальная линия связи

Множественные линии соединения (соединенные вместе)

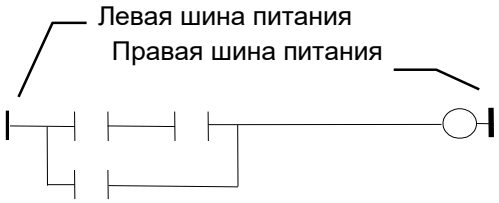
Контакт, связанный с переменной

Виток, связанный с переменной

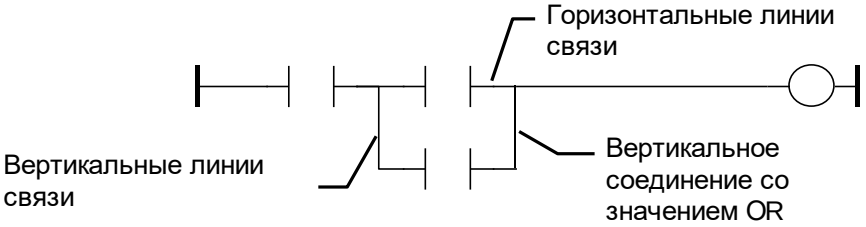
Блок

4.1 Силовые рельсы и соединительные линии

LD диаграмма ограничена справа и слева вертикальными линиями, которые называются **левым силовым рельсом** и **правым силовым рельсом** соответственно.



Символы LD диаграммы связаны с силовыми рельсами и другими символами при помощи соединительных линий. Соединительные линии могут быть горизонтальными или вертикальными.



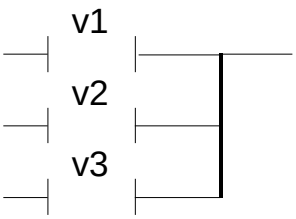
Каждый отрезок линии имеет состояние **TRUE** или **FALSE**. Отрезки, соединенные напрямую имеют одно и то же булевское состояние. Любая горизонтальна линия, соединенная с **левым вертикальным рельсом** имеет состояние **TRUE**.

4.2 Множественные соединения

Левый и правый концы соединительной линии имеют одно и тоже логическое состояние. Комбинируя вертикальные и горизонтальные линии можно строить **множественные соединения**. Состояния концов множественного соединения определяются правилами логики.

Левое множественное соединение объединяет **более чем одну** горизонтальную линию, соединенную с вертикальной линией и одну горизонтальную линию, подходящую с **правой** стороны. Булевское состояние правого конца - это **логическое ИЛИ**(OR) всех левых концов.

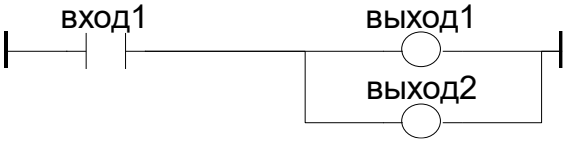
(* Пример множественного левого соединения*)



(* состояние правого конца (v1 OR v2 OR v3) *)

Правое множественное соединение объединяет **одну** горизонтальную линию, соединенную с вертикальной линией с **левой** стороны и **более чем одну** горизонтальную линию, подходящую с **правой** стороны. Булевское состояние левого конца распространяется на все правые концы.

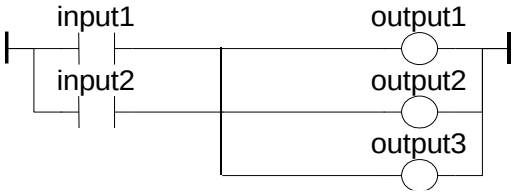
(* Пример множественного ПРАВОГО соединения*)



(* ST эквивалент: *)
output1 := input1;
output2 := input1;

Множественное соединение слева и справа объединяет **более чем одну** горизонтальную линию, соединенную с вертикальной линией с левой стороны и **более чем одну** горизонтальную линию, подходящую с правой стороны. Булевское состояние каждого правого конца - это логическое ИЛИ (OR) всех левых концов.

(*Пример множественного ЛЕВОГО и ПРАВОГО соединения *)



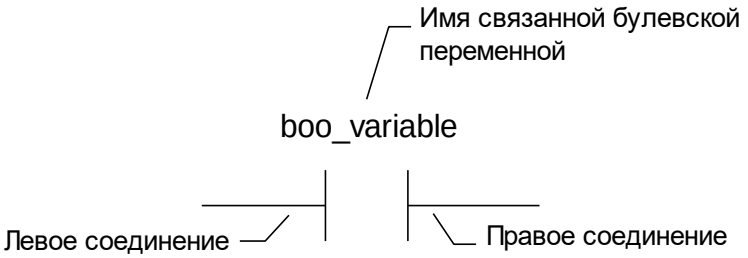
(* ST Эквивалент: *)
output1 := input1 OR input2;
output2 := input1 OR input2;
output3 := input1 OR input2;

4.3 Основные контакты и витки языка LD

Для представления контактов используются символы:
- Прямой контакт
- Инвертированный контакт
- Контакт с определением фронта

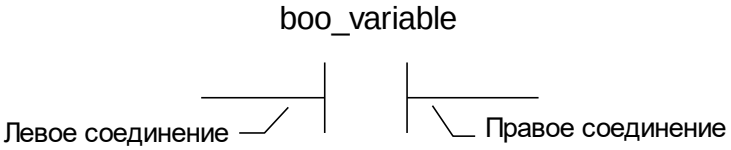
Для представления витков используются символы:
- Прямой виток
- Инвертированный виток
- SET виток
- RESET виток
- Виток с определением фронта

Имя переменной пишется над этими графическими символами:



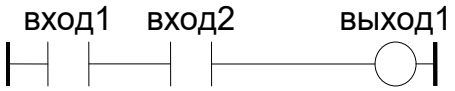
4.3.1 Прямой контакт

Прямой контакт позволяет производить логические операции между состояниями линий и логическими переменными.



Состояние линии соединения на правом конце - это логическое И(AND) состояния левого конца и значения переменной контакта.

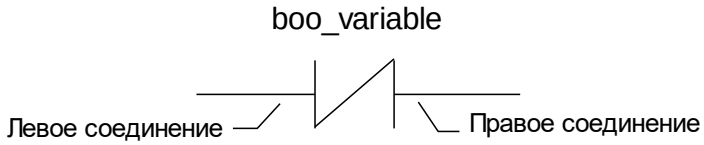
(* Пример использования прямых соединений*)




```
(* ST Эквивалент: *)
output1 := input1 AND input2;
```

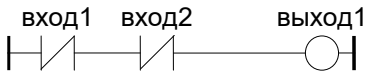
4.3.2 Инвертированный контакт

Инвертированный контакт позволяет производить логические операции между состоянием линии и отрицанием логической переменной.



Состояние линии соединения на правом конце - это логическое И (AND) состояния левого конца и отрицания значения переменной контакта.

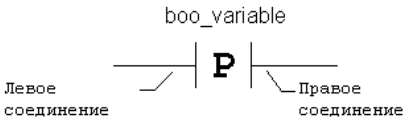
(* Пример использования ИНВЕРТИРОВАННЫХ контактов*)




```
(* ST Эквивалент: *)
output1 := NOT (input1) AND (NOT (input2));
```

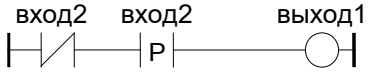
4.3.3 Контакт с определением переднего фронта

Этот контакт позволяет производить логические операции между состоянием линии и логической переменной.



Состояние линии соединения на правом конце принимает значение TRUE, когда значение на левом конце - TRUE и значения переменной контакта меняется с FALSE на TRUE. Во всех остальных случаях оно устанавливается равным FALSE.

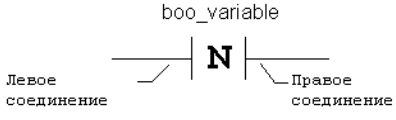
(* Пример использования контактов ПЕРЕДНЕГО ФРОНТА*)




```
(* ST Эквивалент: *)
output1 := input1 AND (input2 AND (NOT (input2prev)));
(* input2prev - значение input2 на предыдущем цикле *)
input2prev := input2;
```

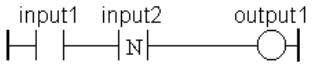
4.3.4 Контакт с определением заднего фронта

Этот контакт позволяет производить логические операции между состоянием линии и логической переменной.



Состояние линии соединения на правом конце принимает значение TRUE когда значение на левом конце - TRUE и значения переменной контакта меняется с TRUE на FALSE. Во всех остальных случаях оно устанавливается равным FALSE.

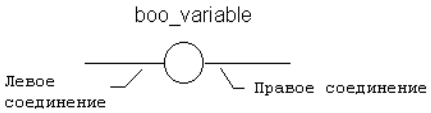
(*Пример использования конакта заднего фронта*)




```
(* ST Эквивалент: *)
output1 := input1 AND (NOT (input2) AND input2prev);
(* input2prev - значение input2 на предыдущем цикле *)
input2prec := input2;
```

4.3.5 Прямой виток

Прямой виток дает логический выход состояния линии соединения.

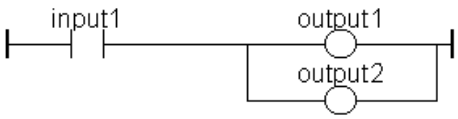


Переменной витка передается значение состояния левой лини соединения. Состояние левого соединения распространяется на правое соединение. Правое соединение может быть связано с правым вертикальным силовым рельсом.

Соответствующая логическая переменная должна быть выходом или внутренней.

Соответствующее имя может быть именем программы (только для подпрограмм). Это соответствует присвоению возвращаемой величины подпрограммы.

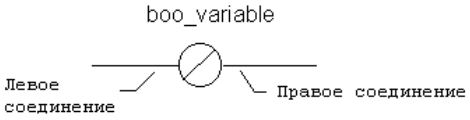
(*Пример использования прямого витка*)




```
(* ST Эквивалент: *)
output1 := input1;
output2 := input1;
```

4.3.6 Инвертированный виток

Инвертированный виток дает логический выход отрицания состояния линии соединения.

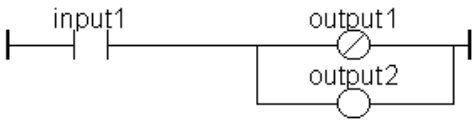


Переменной витка приписывается отрицание значение состояния левой линии соединения. Состояние левого соединения распространяется на правое соединение. Правое соединение может быть связано с правым вертикальным силовым рельсом.

Соответствующая логическая переменная должна быть выходом или внутренней.

Соответствующее имя может быть именем программы (только для подпрограмм). Это соответствует присвоению возвращаемой величины подпрограммы.

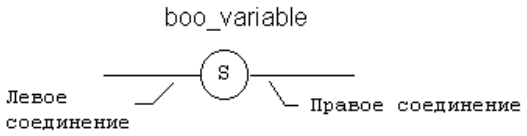
(*Пример использования инвертированного витка*)




```
(* ST Эквивалент: *)
output1 := NOT (input1);
output2 := input1;
```

4.3.7 SET виток

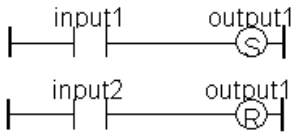
SET виток дает **логический выход** состояния линии соединения.



Соответствующая переменная принимает значение TRUE, когда **состояние левой линии связи** становится равным TRUE. Переменная удерживает это состояние до тех пор, пока она не будет инвертирована витком RESET. Состояние левого соединения распространяется на правое соединение. Правое соединение может быть связано с правым вертикальным силовым рельсом.

Соответствующая логическая переменная должна быть выходной или внутренней.

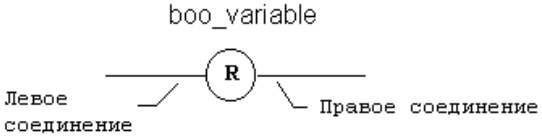
(*Пример использования витков SET и RESET*)




```
(* ST Эквивалент: *)
IF input1 THEN
  output1 := TRUE;
END_IF;
IF input2 THEN
  output1 := FALSE;
END_IF;
```

4.3.8 RESET виток

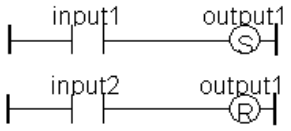
RESET виток дает логический выход состояния линии соединения.



Соответствующая переменная принимает значение FALSE, когда состояние левой линии связи становится равным TRUE. Переменная удерживает это состояние до тех пор, пока она не будет инвертирована витком SET. Состояние левого соединения распространяется на правое соединение. Правое соединение может быть связано с правым вертикальным силовым рельсом.

Соответствующая логическая переменная должна быть выходной или внутренней.

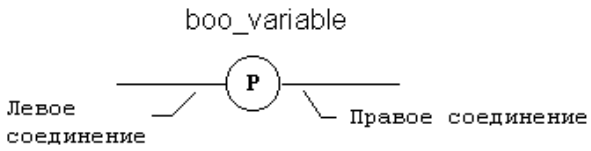
(*Пример использования витков SET и RESET*)




```
(* ST Эквивалент: *)
IF input1 THEN
  output1 := TRUE;
END_IF;
IF input2 THEN
  output1 := FALSE;
END_IF;
```

4.3.9 Виток с определением переднего фронта

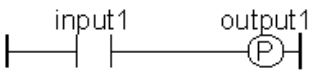
“Положительный” виток дает логический выход состояния линии соединения.



Соответствующая переменная принимает значение TRUE, когда состояние левой линии связи меняет значение с FALSE на TRUE. Во всех остальных случаях переменная принимает значение FALSE. Состояние левого соединения распространяется на правое соединение. Правое соединение может быть связано с правым вертикальным силовым рельсом.

Соответствующая логическая переменная должна быть выходной или внутренней.

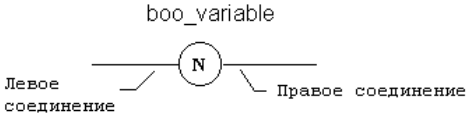
(*Пример использования "положительного" витка*)




```
(* ST Эквивалент: *)
IF (input1 and NOT(input1prev)) THEN
  output1 := TRUE;
ELSE
  output1 := FALSE;
END_IF;
(* input1prev - значение input1 на предыдущем цикле *)
input1prev := input1;
```

4.3.10 Виток с определением заднего фронта

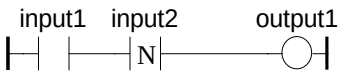
“Отрицательный” виток дает логический выход состояния линии соединения.



Соответствующая переменная принимает значение TRUE, когда состояние левой линии связи меняет значение с TRUE на FALSE. Во всех остальных случаях переменная принимает значение FALSE. Состояние левого соединения распространяется на правое соединение. Правое соединение может быть связано с правым вертикальным силовым рельсом.

Соответствующая логическая переменная должна быть выходной или внутренней.

(* Пример использования “Отрицательного” витка *)




```
(* ST Эквивалент: *)
IF (NOT(input1) and input1prev) THEN
  output1 := TRUE;
ELSE
  output1 := FALSE;
END_IF;
(* input2prev - значение input2 на предыдущем цикле *)
input2prev := input2;
```

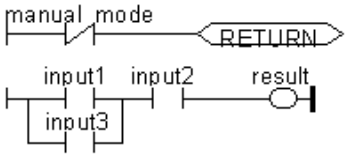
4.4 Оператор RETURN

Метка RETURN может быть использована как выход, чтобы представить условное завершение программы. Никаких символов к правому концу RETURN подключать нельзя.

Если левый конец линии соединения имеет состояние TRUE, то программы завершается, не выполняя операций, введенных в следующих строках программы.

Замечание: Если LD программа - это подпрограмма, то ее имя должно быть связано с выходным витком, чтобы установить возвращаемое значение.

(*Пример использования символа RETURN*)




```
(* ST Эквивалент: *)
If Not (manual_mode) Then RETURN; End_if;
result := (input1 OR input3) AND input2;
```

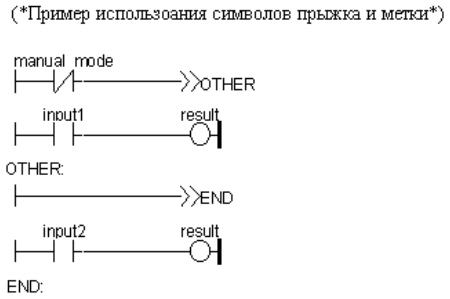
4.5 Прыжки и метки

Метки, условные и безусловные прыжки, могут быть использованы для управления выполнением диаграммы. Никаких символов к правому концу символа метки и прыжка подключать нельзя. Используются следующие обозначения:

>>LAB.....прыжок на метку "LAB"
LAB:.....определение метки "LAB"

Предупреждение: В имени метки должны использоваться только латинские буквы

Если линия связи слева от символа прыжка находится в состоянии **TRUE** , исполнение программы переходит на соответствующую метку.

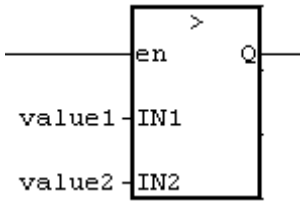


4.6 Блоки

Используя блоки в LD, вы подключаете блок с функцией к логическим линиям. Функция, в действительности, может быть оператором, функциональным блоком или функцией. Так как блоки не всегда имеют логические входы и/или логические выходы, введение блоков в LD диаграммы приводит к добавлению нескольких новых параметров (EN, ENO) в интерфейс блока. Параметры «EN», «ENO» не добавляются, если вы используете FBD/LD редактор, и вы можете подключить переменные требуемого типа.

4.6.1 Вход «EN»

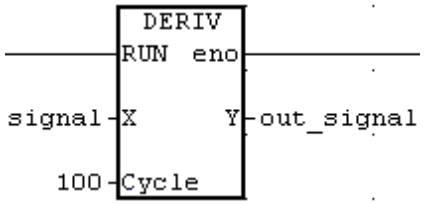
В некоторых операторах, функциях или функциональных блоках первый вход не булевский. Так как первый вход всегда должен быть подключен к шине, на первую позицию автоматически вводится другой вход, называемый «**EN»**. Блок выполняется только тогда, когда вход **EN=**TRUE. Ниже представлен пример оператора сравнения и эквивалентный код на ST:




```
IF rung_state THEN
   q := (value1 > value 2);
ELSE
   q := FALSE;
END_IF;
(* продолжить шину с состоянием q *)
```

4.6.2 Выход «ENO»

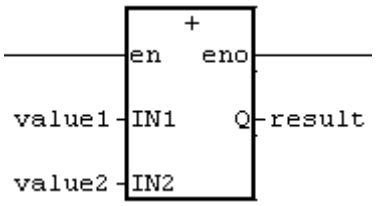
В некоторых операторах, функциях и функциональных блоках первый выход не булевский. Так как первый выход всегда должен быть подключен к шине, на первую позицию автоматически вводится другой выход, называемый «**ENO»**. Выход ENO всегда имеет то же значение, что и первый вход блока. Ниже представлен пример функционального блока AVERAGE и эквивалентный код на ST:




```
DERIV(rung_state, Signal, 100);
OutSignal := DERIV.Y;
eno := rung_state;
(* продолжить шину с состоянием eno*)
```

4.6.3 Параметры «EN» и «ENO»

В некоторых случаях требуются одновременное использование «**EN»** и «**ENO».** Ниже представлен пример с арифметическим оператором и эквивалентный код на ST:




```
IF rung_state THEN
   result := (value1 + value2);
END_IF;
eno := rung_state;
(* продолжить шину с состоянием eno *)
```

5. ЯЗЫК СТРУКТУРИРОВАННОГО ТЕКСТА (ST)

ST (Structured text) - это структурный язык высокого уровня, разработанный для процессов автоматизации. Этот язык, в основном, используется для создания сложных процедур, которые не могут быть легко выражены при помощи более наглядных графических языков.

5.1 Основной синтаксис ST

ST программа - это список ST **операторов**. Имена, используемые в исходном коде (идентификаторы переменных, константы, ключевые слова) разделены **неактивными разделителями** (пробелами, символами окончания строки и табуляции) и **активными разделителями**, которые имеют определенное значение (например, разделитель «>» означает сравнение «больше чем»). В текст могут быть введены комментарии. Комментарий должен начинаться «(*» и заканчиваться «*)». Каждый оператор должен заканчивается точкой с запятой (**;**). Основные операторы языка ST:

- оператор **присвоения** (variable := expression)
- вызов **функции**
- вызов **функционального блока**
- операторы **выбора** (IF, CASE)
- **итеративные** операторы (FOR, WHILE, REPEAT)
- **управляющие** операторы (RETURN, EXIT)
- специальные операторы, расширения Unimod Pro.

Неактивные разделители могут быть свободно введены между активными разделителями, константами и идентификаторами. Неактивные разделители - это **пробелы**, **символы окончания строки** и **табуляции**. Конец строки может быть введен в любом месте программы. Для улучшения читаемости программ нужно использовать неактивные разделители в соответствии со следующими правилами:

- Не пишите более одного оператора в строке
- Используйте табуляцию для сдвига сложных операторов
- Вводите комментарии для улучшения читаемости строк и параграфов

5.2 Выражения и скобки

Выражения ST объединяют **операторы** и **операнды** (переменные или константы). Для одного выражения (объединяющего операнды с одним ST оператором), **тип** операндов должен быть одним и тем же. Это одно выражение имеет тот же тип, что и его операторы и может быть использовано в более сложном выражении. Например:

(boo_var1 AND boo_var2)	тип BOOLEAN
not (boo_var1)	тип BOOLEAN
(2 + 4)*3	тип INTEGER
(t#1s23 + 1)	неправильное выражение

Скобки используются для того, чтобы отделить части выражения и явно определить приоритетность операций. Когда в сложном выражении нет скобок, приоритетность ST операторов задана неявно. Например:

2 + 3 * 6	равно 2+18=20	оператор * имеет высший приоритет
(2+3) * 6	равно 5*6=30	приоритет задается скобками

5.3 Массивы и структуры

Для выделения элемента массива используются квадратные скобки [ ].
Индексы многомерного массива разделяются между собой запятой «**,**». Пример:

Array[1] - выделение первого элемента одномерного массива

b_array[3,4] – выделение элемента третьей строки, четвёртого столбца массива **b_array.**

Для выделения поля структуры используется оператор точка «**.**». Пример:

St.idig1 - выбор переменной с именем idig1 из структуры с именем **St**.

5.4 Вызовы функций и функциональных блоков

Стандартные ST вызовы функций могут быть использованы для каждого из следующих объектов:

- Функции
- Библиотечные функции и функциональные блоки

5.4.1 Вызовы функций

Имя: имя вызываемой функции
Значение: вызывает функции и дает возвращаемое значение

Синтаксис: <variable> := <subprog> (<par1>, ... , <par N>);
Операнды: За типом возвращаемого значения и параметрами вызова должен следовать интерфейс, определенный для подпрограммы

Возвращаемое значение: значение, возвращаемое функцией

Вызовы функций могут быть использованы в выражении.

Пример 1: вызов функции

```
(* Основная ST программа *)
(* получает аналоговое значение *)
ana_timeprog := SPlimit ( tprog_cmd );
```

Пример 2: Вызов функции
(* функции, использующиеся в сложных выражениях: min, max *)

```
limited_value := min (16, max (0, input_value) );
```

5.4.2 Вызов функциональных блоков

Имя: имя экземпляра функционального блока
Значение: вызывает функциональный блок из библиотеки Unimod Pro и передает возвращаемые параметры

Синтаксис: (* вызвать функциональный блок *)
<**blockname**> **(** <**p1**>**,** <**p2**> ... **);**
(получить возвращаемые параметры *)
<**result**> **:=** <**blockname**>**.** <**ret_param1**>**;**
...
<**result**> **:=** <**blockname**>**.** <**ret_paramN**>**;**
Операнды: Параметры и выражения, которые имеют тот же тип, что и параметры этого функционального блока
Возвращаемое значение: См. Синтаксис получения возвращаемых параметров

Примечание. Смотрите библиотеку Unimod Pro для того, чтобы найти значение и тип каждого параметра функционального блока. Экземпляр функционального блока (имя копии) должно быть объявлено в словаре.

Пример:

```
(*вызов функционального блока ST программой *)
(* объявить экземпляр функционального блока в словаре: *)
(* trigb1 : block R_TRIG - определение переднего фронта*)

(* активизация функционального блока из языка ST *)
trigb1 (b1);
```

```
(* доступ к возвращаемым параметрам *)
If (trigb1.Q) Then nb_edge := nb_edge + 1;  End_if;
```

5.5 Булевские операторы языка ST

- NOT	логическое отрицание
- AND	логическое И (AND)
- OR	логическое ИЛИ (OR)
- XOR	логическое исключающее ИЛИ (OR)

Описание этих операторов представлено в главе «Операторы, функциональные блоки и функции».

5.6 Основные операторы ST

Основные операторы языка ST:

1. Присвоение
2. Оператор RETURN
3. Структура IF-THEN-ELSIF-ELSE
4. Оператор CASE
5. Итерационный оператор WHILE
6. Итерационный оператор REPEAT
7. Итерационный оператор FOR
8. Оператор EXIT
9. Оператор GOTO

5.6.1 Присвоение

Name: :=
Значение: присваивает переменной значение выражения
Синтаксис: <variable> := <any_expression>;
Операнды: переменная должна быть внутренняя или выход, переменная и выражение должны иметь один и тот же тип

Выражение может быть вызовом подпрограммы или функции из библиотеки Unimod Pro.
Пример:

```
(* ST программа с присвоением *)

(* variable <<= variable *)
bo23 := bo10;

(* variable <<= expression *)
bo56 := bx34 OR alrm100 & (level >= over_value);
result := (100 * input_value) / scale;

(* присвоение с вызовом функции *)
limited_value := min (16, max (0, input_value) );
```

5.6.2 Оператор RETURN

Имя: **RETURN**
Значение: заканчивает выполнение текущей программы
Синтаксис: RETURN;
Операнды: (нет)

```
(* ST программа *)
If not (CU) then
        Q := false;
```

```
        CV := 0;
        RETURN;   (* закончить программу*)
end_if;

if R then
        CV := 0;
else
        if (CV < PV) then
                CV := CV + 1;
        end_if;
end_if;
Q := (CV >= PV);
```

5.6.3 Структура IF-THEN-ELSIF-ELSE

Имя: **IF... THEN... ELSIF... THEN... ELSE... END_IF**
Значение: выполняет один или два списка ST операторов, выбор осуществляется в соответствии со значением булевского выражения

Синтаксис: IF <boolean_expression> THEN

```
        <statement> ;
        <statement> ;
        ...
      ELSIF <boolean_expression> THEN
        <statement> ;
        <statement> ;
        ...
      ELSE
        <statement> ;
        <statement> ;
        ...
      END_IF;
```

Операторы ELSE и ELSIF - дополнительные. Если ELSE отсутствует и условие равно FALSE, то никаких инструкций не выполняется.
Пример:
(*ST программа, использующая оператор IF*)

```
IF manual AND not (alarm) THEN
        level := manual_level;
        bx126 := bi12 OR bi45;
ELSIF over_mode THEN
        level := max_level;
ELSE
        level := (lv16 * 100) / scale;
END_IF;

(* IF структура  без ELSE *)
If overflow THEN
          alarm_level := true;
END_IF;
```

5.6.4 Оператор CASE

Имя: **CASE ... OF ... ELSE ... END_CASE**
Значение: выполняет один или несколько списков ST операторов, выбор осуществляется в соответствии с целым выражением

Синтаксис: CASE <integer_expression> OF

```
        <value> : <statements> ;
```

```
    <value> , <value> : <statements> ;
    ...
  ELSE
    <statements> ;
  END_CASE;
```

Значением CASE должна быть целая константа. Несколько значений разделенных запятыми могут указывать на один и тот же список операторов. Оператор ELSE – дополнительный, выполняется, когда не одна из констант не совпала с условием.

Пример:

```
(*ST программа , использующая оператор CASE*)

CASE error_code OF
    255:  err := 1;
          fatal_error := TRUE;
    1:    err := 2;
    2, 3: err := 7;
ELSE
    err := 0;
END_CASE;
```

5.6.5 Итерационный оператор WHILE

Имя: **WHILE... DO... END_WHILE**
Значение: итерационная структура для группы ST операторов, условие вычисляется **прежде** выполнения итерации

Синтаксис: WHILE <boolean_expression> DO

```
    <statement> ;
    <statement> ;
    ...
  END_WHILE;
```

Предупреждение: ***Так как Unimod Pro синхронная система, входные переменные не обновляются во время итераций WHILE. Изменение состояния входных переменных не может быть использовано для описания условия оператора WHILE.***

Пример:

```
(*ST программа, использующая оператор WHILE, подсчитывает факториал 5*)

fac := 1;
i:=1;
WHILE (i<6) DO
    fac := fac * i;
    i := i + 1;
END_WHILE;
```

5.6.6 Итерационный оператор REPEAT

Имя: **REPEAT... UNTIL... END_REPEAT**
Значение: итерационная структура для группы ST операторов, условие вычисляется **<u>после</u>** выполнения итерации. Условие является условием продолжения выполнения цикла.

Синтаксис: REPEAT
<statement> ;
<statement> ;
...
UNTIL <boolean_condition>
END_REPEAT;

<u>Предупреждение:</u> Так как Unimod Pro синхронная система входные переменные не обновляются во время итераций REPEAT. Изменение состояния входных переменных не может быть использовано для описания условия оператора REPEAT.

Пример:
(*ST программа, использующая оператор REPEAT, подсчитывает факториал 5*)

```
fac := 1;
i:=1;
REPEAT
        fac := fac * i;
        i := i + 1;
UNTIL ( i<=5 )
END_REPEAT;
```

5.6.7 Оператор FOR

Имя: **FOR... TO... BY... DO... END_FOR**
Значение: выполняет ограниченное число итераций, используя целую аналоговую индексную переменную

Синтаксис: FOR <index>:= <mini> TO <maxi> BY <step> DO
<statement> ;
<statement> ;
END_FOR;

Операнды:
index: внутренняя аналоговая переменная, увеличивающаяся на каждом витке
mini: начальное значение для индекса (перед первой итерацией)

maxi: максимально-допустимое значение индекса
step: приращение индекса на каждом шаге

Оператор [BY step] - дополнительный. Если он не используется, то приращение равно 1.
“While” эквивалент оператора FOR:

```
index := mini;
while (index <= maxi) do
  <statement> ;
  <statement> ;
  index := index + step;
end_while;
```

Пример:
(*ST программа, использующая оператор FOR, подсчитывает факториал 5*)

```
fac := 1;
```

```
FOR i := 1 TO 5 BY 1 DO
        fac := fac * i;
END_FOR;
```

5.6.8 Оператор EXIT

Имя: **EXIT**
Значение: выход из итерационных операторов FOR, WHILE, REPEAT
Синтаксис: **EXIT;**
Операнды: **(нет)**
EXIT, обычно, используется в операторе IF внутри блоков FOR, WHILE, REPEAT.

Пример:

```
(*ST программа, использующая оператор EXIT*)
num:=10;
fac := 1;
i:=1;

WHILE true DO
        IF fac>num THEN
            Num:=fac;
            EXIT;
        END_IF;
        fac := fac * i;
        i := i + 1;
END_WHILE;
```

5.6.9 Оператор GOTO

Имя: **GOTO**
Значение: переход на метку
Синтаксис: **GOTO метка;**
Операнды: метка – уникальных идентификатор в программе, не описывается в словарях, имя может совпадать с именем переменной, распознается наличием за именем символа двоеточие «:» Одна программа не может содержать более одной метки с одним именем. Количество меток в программе может быть не ограничено.

Примечание. В имени метки должны использоваться только латинские буквы

Пример:

```
(*ST программа, использующая оператор GOTO*)

IF I>5 THEN
          GOTO label;
END_IF;
f: (* метка *) num:=10;
label: (* метка *) fac := 1;
```

5.7 Расширения ST

Следующие функции являются расширениями языка ST:
TSTART-TSTOP: управление таймером

Примечание: эти функции не относятся к стандарту IEC 1131-3.

5.7.1 TSTART оператор

Имя: **TSTART**
Значение: запускает переменную таймер, таймер не модифицируется командой TSTART, счет начинается с текущего значения таймера

Синтаксис: TSTART (<time_variable>);
Операнды: Любая неактивная таймерная переменная
Возвращаемое значение: (нет)

Пример:

```
(*ST программа, использующая оператор TSTART и TSTOP*)
TSTART (T);
IF T>t#1s THEN
        TSTOP (T);
END_IF
```

Таймер сохраняет свое значение в течение одного цикла.

5.7.2 TSTOP оператор

Имя: **TSTOP**
Значение: останавливает переменную таймер, таймер не модифицируется командой TSTOP.

Синтаксис: TSTOP (<time_variable>);
Операнды: Любая активная таймерная переменная
Возвращаемое значение: (нет)

Пример: см. оператор TSTART.

5.8 Комментарии

Комментарии в языке ST должны начинаться со специального символа “(*” и заканчиваться символом “*)” . Комментарии могут быть введены в любом месте программы ST и могут занимать несколько строк.

```
counter := ivalue;   (* присвоить значение основному счетчику *)
(* это комментарий
на двух строках *)
c := counter (* комментарий можно расположить где угодно *) + base_value + 1;
```

Комментарии могут частично перекрываться. Это означает, что символ “(*” может быть использован внутри комментария, а символ “*)” нет.

6. ЯЗЫК ПОСЛЕДОВАТЕЛЬНЫХ ФУНКЦИОНАЛЬНЫХ СХЕМ (SFC)

Язык последовательных функциональных схем (SFC) – это графический язык, который используется для описания последовательных операций. Процесс представляется в виде набора определённых шагов, связанных переходами. К каждому переходу прикреплено логическое условие. Действия внутри шагов описаны более детально при помощи других языков (ST, LD, FBD).

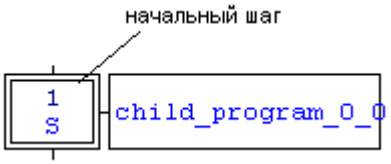
6.1 Основной формат схемы SFC

SFC программа - это графический набор шагов и переходов, соединенных вместе связями. Для обозначения схождений и расхождений используются множественные связи. Вот основные графические правила для SFC:
- шаги не могут следовать подряд
- переходы не могут следовать подряд.

6.2 Шаги и начальные шаги

Шаг представляется одиночным квадратом. Каждому шагу присваивается номер, написанный внутри квадрата. Соответствующее шагу действие определяется внутри прямоугольника , присоединенного к символу шага. Реальная работа шага (действия) описывается в отдельном окне среды программирования и не отражается на диаграмме.
Шаги на схеме могут быть пустыми (не связанными ни с каким действием), что не вызывает ошибки при компиляции проекта. Определить действия, соответствующие шагу, можно в любое время.

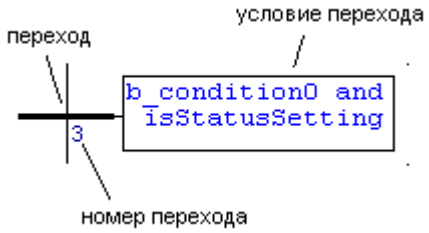
Если шаг связан с действием, также должен быть определён классификатор, определяющий способ влияния активного шага на действие, и, для некоторых классификаторов – временная константа.
Начальная ситуация программы SFC описывается начальными шагами. Начальный шаг обозначается графическим символом с двойной рамкой. После запуска программы маркер автоматически устанавливается на каждый начальный шаг.



SFC-программа (не пустая) должна содержать, по крайней мере, один начальный шаг.

6.3 Переходы

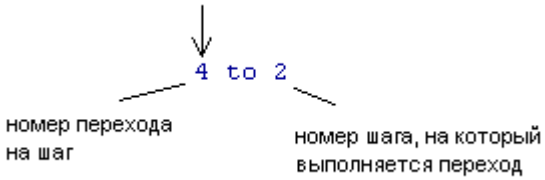
Переходы представлены горизонтальными полосками, которые пересекают линии связи. Каждому переходу присвоен номер, следующий за символом перехода. Описание перехода располагается справа от символа перехода.
Условием перехода может служить логическая переменная, логическое выражение, константа – любое выражение булевского типа. Условия отображаются на диаграмме справа от черты перехода.



Переход выполняется при соблюдении двух условий:
1) переход разрешен (соответствующий ему шаг активен);
2) условие перехода имеет значение TRUE.

6.4 Переход на шаг

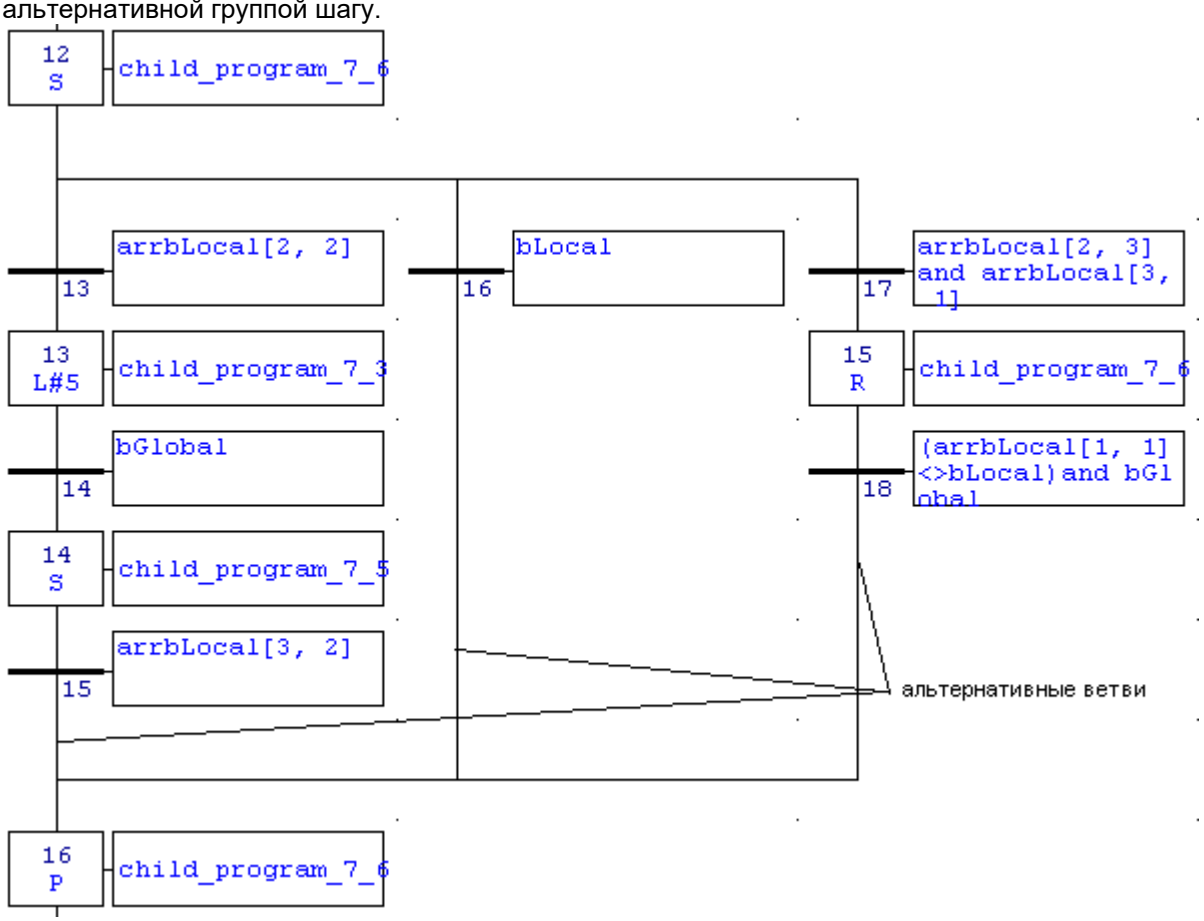
В общем случае SFC-схема выполняется сверху вниз. Стандартом допускается создание переходов на произвольный шаг. Символ перехода на шаг может быть использован, чтобы определить связь от перехода к шагу. То есть, переход выполняется на шаг, имя которого указано под стрелкой.



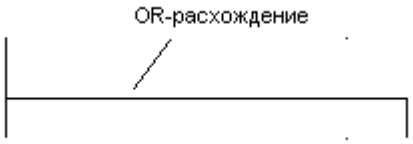
Прыжок из одной ветви параллельного блока наружу вызывает эффект размножения маркера. Прыжок внутрь параллельного блока нарушает параллельность ветвей. Применение этих приёмов запрещено.

6.5 Альтернативные ветви

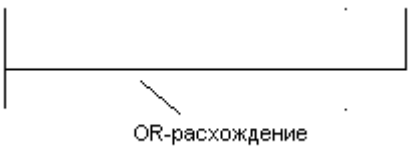
Несколько ветвей SFC могут быть альтернативными ветвями. Признаком альтернативных ветвей на схеме является одинарная горизонтальная линия. Каждая альтернативная ветвь начинается и заканчивается собственным условием перехода. Проверка альтернативных условий выполняется слева направо. Если верное условие найдено, то прочие альтернативы не рассматриваются. В альтернативных ветвях всегда работает только одна из ветвей, поэтому ее окончание и будет означать переход к следующему за альтернативной группой шагу.



В SFC-схеме начало альтернативного ветвления создаётся с помощью OR-расхождения. OR-расхождение – это множественная связь от одного шага к нескольким переходам.



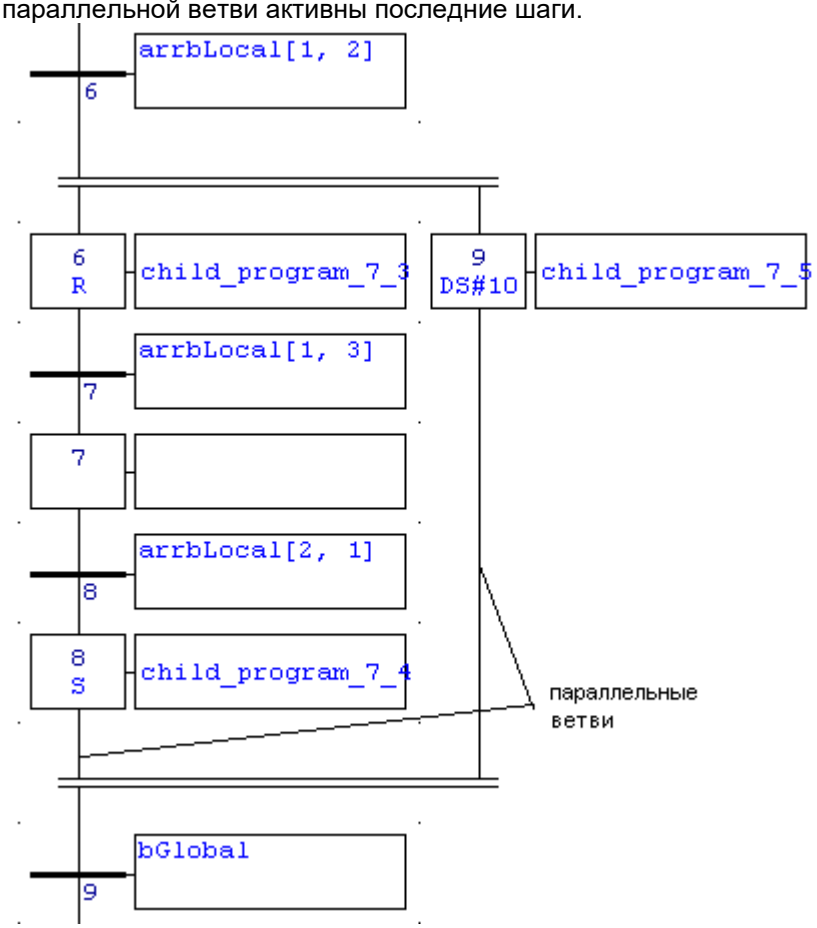
Конец альтернативного ветвления создаётся с помощью OR-схождения. OR-схождение – это множественная связь от нескольких переходов к одному шагу.



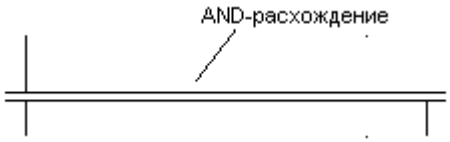
При создании альтернативных ветвей желательно задавать взаимоисключающие условия перехода на них. В этом случае вероятность допустить ошибку при анализе или в процессе доработки диаграммы значительно ниже.

6.6 Параллельные ветви

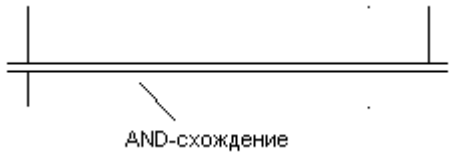
Несколько ветвей SFC могут быть параллельными. Признаком параллельных ветвей на схеме является двойная горизонтальная линия. Каждая параллельная ветвь начинается и заканчивается шагом. То есть, условие входа в параллельность всегда одно, условие выхода тоже одно на всех.
Параллельные ветви выполняются теоретически одновременно. В жизни это означает — в одном рабочем цикле, слева направо.
Условие перехода, завершающее параллельность, проверяется только в случае, если в каждой параллельной ветви активны последние шаги.



В SFC-схеме начало параллельного ветвления создаётся с помощью AND-расхождения. AND-расхождение – это множественная связь от одного перехода к нескольким шагам.



Конец параллельного ветвления создаётся с помощью AND-схождения. AND-схождение – это множественная связь от нескольких шагов к одному и тому же переходу.



6.7 Действия внутри шагов

В роли действий SFC выступают дочерние программы без входных параметров, возвращающие булевское значение. Каждое действие может сопоставляться одному или нескольким шагам. Соответствующее шагу действие определяется внутри прямоугольника , присоединенного к символу шага.
Если шаг связан с действием, квадрат, отображающий шаг, содержит специальное поле — классификатор. Классификатор (qualifier) определяет способ влияния активного шага на данное действие. Следующая таблица даёт описание возможных классификаторов.

Классификатор	**Краткое обозначение**	**Описание**
отсутствие классификатора	-	То же самое, что и классификатор N
N	Несохраняемое (Non-stored)	Данное действие будет выполняться в каждом рабочем цикле, пока активен шаг.
P	Импульс (Pulse)	Действие выполняется один раз при активации шага.
S	Сохраняемое (Stored)	Действие активируется и остается активным до сброса. Действие продолжит выполняться в каждом цикле даже тогда, когда шаг уже не активен.
R	Сброс (Reset)	Действие деактивируется.
L	Ограниченное по времени (time Limited)	Действие активируется вместе с шагом и остается активным на заданное время, но не дольше, чем шаг. Возможны два варианта деактивации действия: действие деактивируется по истечении времени, либо по причине деактивации шага.
SL	Сохраняемое и ограниченное по времени (Stored and time Limited)	Действие активируется вместе с шагом и остается активным заданное время, вне зависимости от активности шага. Действие можно деактивировать досрочно из другого шага с классификатором R.
D	Отложенное (Delayed)	Действие активируется через заданное время после активации шага и остается активным, пока активен шаг. Если шаг окажется активным меньше заданного времени, то действие не будет активировано.
DS	Отложенное сохраняемое (Delayed and Stored)	Действие активируется через заданное время после активации шага и остается активным до сброса. Если шаг активен меньше заданного времени, то действие не будет активировано. При параллельном выполнении сброса в процессе отсчета времени (в другом шаге с классификатором R) действие не будет активироваться.
SD	Сохраняемое отложенное (Stored and time Delayed)	Действие активируется через заданное время после активации шага, даже если шаг уже не активен. Но если в процессе отсчета задержки активации выполнить сброс (в другом шаге с классификатором R), то активация не произойдет. Активированное действие остается активным до сброса.

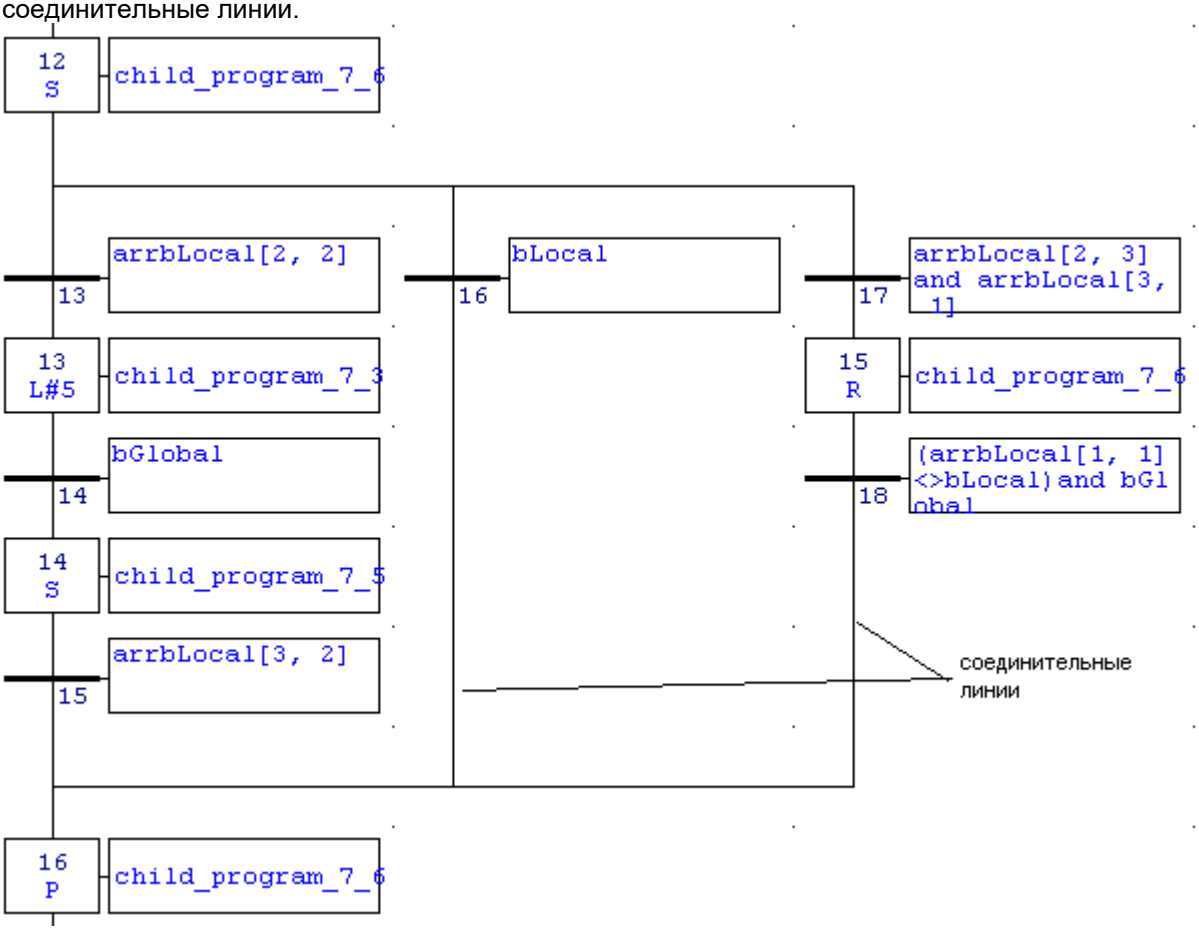
Классификаторы L, SL, D, DS, SD требуют указания константы времени, например: t#100ms.

6.8 Условия, присоединённые к переходам

К каждому переходу присоединяется логическое выражение, которое является условием прохождения этого перехода. Взамен логического выражения к переходу можно привязать вызов булевской функции или дочерней программы без параметров. Условия отображаются на диаграмме справа от черты перехода.
Если к переходу не присоединено выражение, то по умолчанию условие - TRUE.

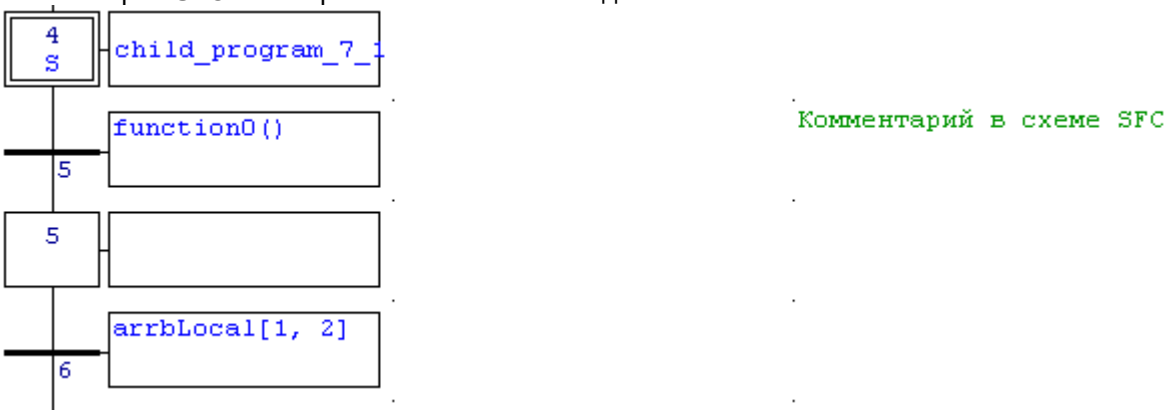
6.9 Соединительные линии

Во многих случаях параллельные или альтернативные ветви не обладают одинаковым количеством шагов и переходов в каждой ветви. Тогда, во избежание появления разрывов в схеме, следует использовать соединительные линии.



6.10 Комментарий

Комментарий SFC может располагаться в свободном месте на схеме.



7. ОПЕРАТОРЫ, ФУНКЦИОНАЛЬНЫЕ БЛОКИ И ФУНКЦИИ

7.1 Операторы

В Unimod Pro реализованы следующие операторы:

Работа с данными
Присвоение
Аналоговое отрицание

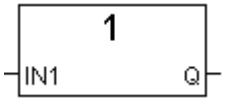
Булевские операции
Логическое отрицание (NOT)
Логическое И (AND)
Логическое ИЛИ (OR)
Логическое исключающее ИЛИ (XOR)

Арифметические операции
Сложение
Вычитание
Умножение
Деление

Логические операции
Аналоговое побитовое И (AND)
Аналоговое побитовое ИЛИ (OR)
Аналоговое побитовое исключающее ИЛИ (XOR)
Аналоговое побитовое отрицание

Сравнения
Меньше
Меньше или равно
Больше
Больше или равно
Равно
Неравно

7.1.1 Присваивание (:=1)



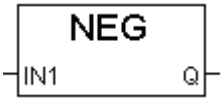
Вход:	IN1	любой тип
Выход:	Q	любой тип

Описание:
Присваивает значение одной переменной другой переменной.

Этот блок очень удобен для прямой связки диаграммы входа с диаграммой выхода. Он может быть также использован (с линией булевского отрицания) для инвертирования состояния линии соединенной с диаграммой выхода.

```
(* ST эквивалент: *)
ao23 := ai10;
```

7.1.2 Аналоговое отрицание NEG

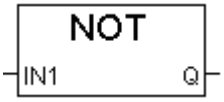


Вход: IN1 INTEGER-REAL
Выход: Q INTEGER-REAL

Замечание: вход и выход должны иметь один и тот же формат
Описание:
Выполняет отрицание переменной.

(* ST эквивалент: *)
ao23 := - (ai10);

7.1.3 Логическое отрицание NOT

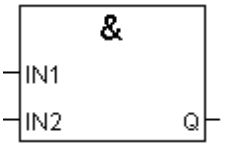


Вход:
IN1 BOOLEAN

Выход:
Q BOOLEAN логическое отрицание

Описание:
Логическое отрицание.
(* ST эквивалент: *)
bo10 := not bi101;

7.1.4 Логическое И (AND) &



Входы:
N1 BOOLEAN
N2 BOOLEAN

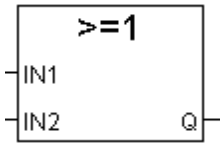
Выход:
Q BOOLEAN логическое И двух (или более) входов

Замечание: Для этого оператора количество входов может быть больше чем два.
Описание:
Логическое И двух (или более) значений.

(* ST эквивалент: *)

```
bo10 := bi101 AND bi102;
```

7.1.5 Логическое ИЛИ (OR) >=1



Входы:

IN1	BOOLEAN
IN2	BOOLEAN

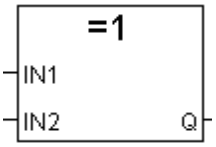
Выход:

Q	BOOLEAN	логическое ИЛИ двух (или более) входов

Замечание: Для этого оператора количество входов может быть больше чем два.
Описание:
Логическое ИЛИ двух (или более) значений.

```
(* ST эквивалент: *)
bo10 := bi101 OR bi102;
```

7.1.6 Логическое исключающее ИЛИ (XOR) =1



Входы:

IN1	BOOLEAN
IN2	BOOLEAN

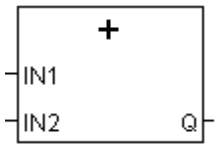
Выход:

Q	BOOLEAN	логическое исключающее ИЛИ двух входов

Описание:
Логическое исключающее ИЛИ двух значений.

```
(* ST эквивалент: *)
bo10 := bi101 XOR bi102;
```

7.1.7 Сложение (+)



Входы:

IN1	INTEGER-REAL-TIMER-MESSAGE
IN2	INTEGER-REAL-TIMER-MESSAGE

Выход:

Q	INTEGER-REAL-TIMER-MESSAGE	знаковое сложение входов

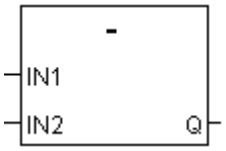
Замечания: Количество входов может быть больше чем два. Все входы и выходы должны быть одного формата

Описание:

Складывает две (или более) переменных.

```
(* ST эквивалент: *)
ao10 := ai101 + ai102+ ai103+ ai104;
```

7.1.8 Вычитание (-)



Входы:

IN1	INTEGER-REAL-TIMER
IN2	INTEGER-REAL-TIMER

Выход:

Q	INTEGER-REAL-TIMER	вычитание (IN1 - IN2)

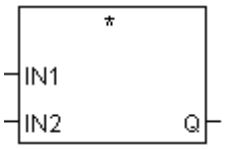
Замечания: Все входы и выходы должны быть одного формата

Описание:

Вычитает одну переменную из другой (первый - второй).

```
 (* ST эквивалент: *)
ao10 := ai101 - ai102;
```

7.1.9 Умножение (*)



Входы:

IN1	INTEGER-REAL
IN2	INTEGER-REAL

Выход:

Q	INTEGER-REAL	знаковое умножение IN1 на IN2

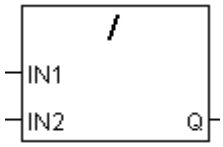
Замечания: Количество входов может быть больше чем два. Все входы и выходы должны быть одного формата

Описание:

Умножает две или более переменных.

```
(* ST эквивалент *)
ao10 := ai101 * ai102;
```

7.1.10 Деление (/)



Входы:

IN1	INTEGER-REAL
IN2	INTEGER-REAL

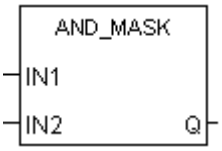
Выход:

Q	INTEGER-REAL	знаковое деление IN1 на IN2

Замечания: Все входы и выходы должны быть одного формата
Описание:
Делит две переменные (первый / второй).

```
(* ST эквивалент *)
ao10 := ai101 / ai102;
```

7.1.11 Аналоговое побитовое И (AND) AND_MASK



Входы:

IN1	INTEGER	
IN2	INTEGER	

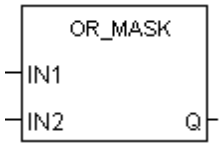
Выход:

Q	INTEGER	побитовый логический AND между IN1 и IN2

Описание:
Целый побитовый И.

```
(* ST эквивалент: *)
parity := AND_MASK (xvalue, 1);
```

7.1.12 Аналоговое побитовое ИЛИ (OR) OR_MASK



Входы:

IN1	INTEGER	
IN2	INTEGER	

Выход:

Q	INTEGER	побитовый логический OR между IN1 и IN2

Описание:

Целый побитовый ИЛИ.
(* ST эквивалент: *)
is_odd := OR_MASK (xvalue, 1);

7.1.13 Аналоговое побитовое исключающее ИЛИ (XOR) XOR_MASK

Входы:

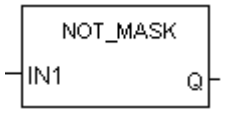
IN1	INTEGER	
IN2	INTEGER	

Выход:

Q	INTEGER	побитовый логический XOR между IN1 и IN2

Описание:
Целый побитовый исключающий ИЛИ.
(* ST эквивалент: *)
is_r := XOR_MASK (xvalue, 1);

7.1.14 Аналоговое побитовое отрицание NOT_MASK



Вход:

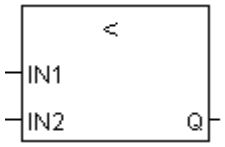
IN1	INTEGER	

Выход:

Q	INTEGER	побитовое логическое отрицание 32-х разрядного IN1

Описание:
Целое побитовое логическое отрицание.
(*ST эквивалент: *)
result := NOT_MASK (xvalue);

7.1.15 Меньше <



Входы:

IN1	INTEGER-REAL-TIMER-MESSAGE	
IN2	INTEGER-REAL-TIMER-MESSAGE	

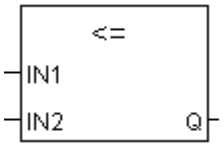
Выход:

Q	INTEGER-REAL-TIMER-MESSAGE	TRUE, если IN1 < IN2

Замечания: Все входы должны быть одного формата
Описание:
Проверить, что одна величина меньше другой.
(* ST Эквивалент: *)

aresult := (10 < 25); (* aresult is TRUE *)

7.1.16 Меньше или равно <=



Входы:

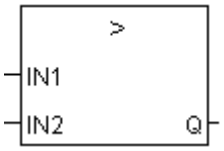
IN1	INTEGER-REAL-TIMER-MESSAGE	
IN2	INTEGER-REAL-TIMER-MESSAGE	

Выход:

Q	INTEGER-REAL-TIMER-MESSAGE	TRUE, если IN1 <= IN2

Замечания: Все входы должны быть одного формата
Описание:
Проверить, что одна величина меньше или равна другой.
(* ST Эквивалент: *)
aresult := (10 <= 25); (* aresult is TRUE *)

7.1.17 Больше >



Входы:

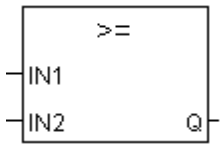
IN1	INTEGER- REAL-TIMER- MESSAGE	
IN2	INTEGER- REAL-TIMER- MESSAGE	

Выход:

Q	INTEGER- REAL-TIMER- MESSAGE	TRUE, если IN1 > IN2

Замечания: Все входы должны быть одного формата
Описание:
Проверить, что одна величина больше другой.
(* ST Эквивалент: *)
aresult := (10 > 25); (* aresult is TRUE *)

7.1.18 Больше или равно >=



Входы:

IN1	INTEGER-REAL-TIMER-MESSAGE	
IN2	INTEGER-REAL-TIMER-MESSAGE	

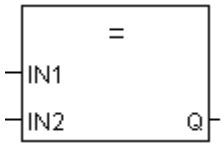
Выход:

Q	INTEGER-REAL-TIMER-MESSAGE	TRUE, если IN1 >= IN2

Замечания: Все входы должны быть одного формата
Описание:
Проверить, что одна величина больше или равна другой.
(* ST Эквивалент: *)

aresult := (10 >= 25); (* aresult is TRUE *)

7.1.19 Равно =



Входы:

IN1	INTEGER-REAL-TIMER-MESSAGE	
IN2	INTEGER-REAL-TIMER-MESSAGE	

Выход:

Q	INTEGER-REAL-TIMER-MESSAGE	TRUE, если IN1 < IN2

Замечания: Все входы должны быть одного формата. Не рекомендуется использовать при сравнении таймерных и вещественных переменных.

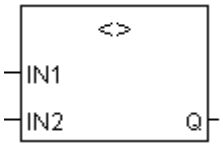
Описание:

Проверить, что одна величина равна другой.

(* ST Эквивалент: *)

aresult := (10 = 25); (* aresult is TRUE *)

7.1.20 Неравно <>



Входы:

IN1	INTEGER-REAL-TIMER-MESSAGE	
IN2	INTEGER-REAL-TIMER-MESSAGE	

Выход:

Q	INTEGER-REAL-TIMER-MESSAGE	TRUE, если IN1 <> IN2

Замечания: Все входы должны быть одного формата. Не рекомендуется использовать при сравнении таймерных и вещественных переменных.

Описание:

Проверить, что одна величина не равна другой.

(* ST Эквивалент: *)

aresult := (10 <> 25); (* aresult is TRUE *)

7.2 Функции преобразования типов

В Unimod Pro реализованы следующие функции преобразования:

Преобразования булевских величин

BOOL_TO_BYTE	преобразовывает булевское значение в байтовое беззнаковое
BOOL_TO_DOUBLE	преобразовывает булевское значение в вещественное двойной точности
BOOL_TO_INT	преобразовывает булевское значение в целое
BOOL_TO_REAL	преобразовывает булевское значение в вещественное
BOOL_TO_TIME	преобразовывает булевское значение в таймер
BOOL_TO_STRING	преобразовывает булевское значение в строку

Преобразования целых величин

INT_TO_BOOL	преобразовывает целое значение в булевское
INT_TO_BYTE	преобразовывает целое значение в байтовое беззнаковое
INT_TO_DOUBLE	преобразовывает целое значение в вещественное двойной точности
INT_TO_REAL	преобразовывает целое значение в вещественное
INT_TO_TIME	преобразовывает целое значение в таймер
INT_TO_STRING	преобразовывает целое значение в строку

Преобразования действительных величин

REAL_TO_BOOL	преобразовывает действительное значение в булевское
REAL_TO_BYTE	преобразовывает действительное значение в байтовое беззнаковое
REAL_TO_DOUBLE	преобразовывает действительное значение в вещественное двойной точности
REAL_TO_INT	преобразовывает действительное значение в целое
REAL_TO_TIME	преобразовывает действительное значение в таймер
REAL_TO_STRING	преобразовывает действительное значение в строку

Преобразования таймеров

TIME_TO_BOOL	преобразовывает таймер в булевское значение
TIME_TO_BYTE	преобразовывает таймер в байтовое беззнаковое значение
TIME_TO_DOUBLE	преобразовывает таймер в вещественное двойной точности
TIME_TO_INT	преобразовывает таймер в целое значение
TIME_TO_REAL	преобразовывает таймер в вещественное значение
TIME_TO_STRING	преобразовывает таймер в строку

Преобразования строк

STRING_TO_BOOL	преобразовывает строку в булевское значение
STRING_TO_BYTE	преобразовывает строку в байтовое беззнаковое значение
STRING_TO_DOUBLE	преобразовывает строку в вещественное двойной точности
STRING_TO_INT	преобразовывает строку в целое значение
STRING_TO_REAL	преобразовывает строку в действительное значение
STRING_TO_TIME	преобразовывает строку в таймерное значение

7.2.1 BOOL_TO_BYTE - Преобразование булевского значения в байтовое беззнаковое

Вход: IN1 BOOLEAN
Выход: Q BYTE

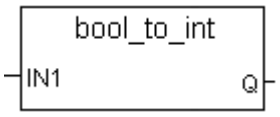
Описание:
Преобразовывает булевское значение в байтовое беззнаковое.
(* ST эквивалент: *)
integer := BOOL_TO_BYTE(TRUE);

7.2.2 BOOL_TO_DOUBLE - Преобразование булевского значения в вещественное двойной точности

Вход: IN1 BOOLEAN
Выход: Q DOUBLE

Описание:
Преобразовывает булевское значение в вещественное двойной точности.
(* ST эквивалент: *)
integer := BOOL_TO_DOUBLE(TRUE);

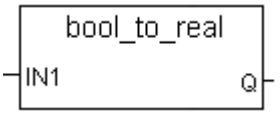
7.2.3 BOOL_TO_INT - Преобразование булевского значения в целое



Вход: IN1 BOOLEAN
Выход: Q INTEGER

Описание:
Преобразовывает булевское значение в целое.
(* ST эквивалент: *)
integer := BOOL_TO_INT(TRUE);

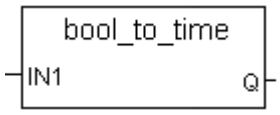
7.2.4 BOOL_TO_REAL - Преобразование булевского значения в вещественное



Вход: IN1 BOOLEAN
Выход: Q REAL

Описание:
Преобразовывает булевское значение в вещественное.
(* ST эквивалент: *)
real := BOOL_TO_REAL(TRUE);

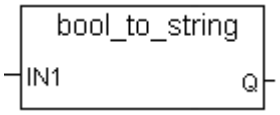
7.2.5 BOOL_TO_TIME - Преобразование булевского значения в таймерное



Вход: IN1 BOOLEAN
Выход: Q TIMER

Описание:
Преобразовывает булевское значение в таймер.
(* ST эквивалент: *)
timer := BOOL_TO_TIME(TRUE);

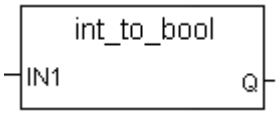
7.2.6 BOOL_TO_STRING - Преобразование булевского значения в строку



Вход: IN1 BOOLEAN
Выход: Q MESSAGE

Описание:
Преобразовывает булевское значение в строку.
(* ST эквивалент: *)
string := BOOL_TO_STRING(FALSE);

7.2.7 INT_TO_BOOL - Преобразование целого значения в булевское



Вход: IN1 INTEGER
Выход: Q BOOLEAN

Описание:
Преобразовывает целое значение в булевское.
(* ST эквивалент: *)
odig1 := INT_TO_BOOL(100);

7.2.8 INT_TO_BYTE - Преобразование целого значения в байтовое беззнаковое

Вход: IN1 INTEGER
Выход: Q BYTE

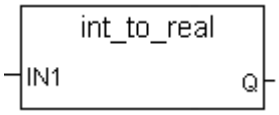
Описание:
Преобразовывает целое значение в байтовое беззнаковое.
(* ST эквивалент: *)
odig1 := INT_TO_BYTE(100);

7.2.9 INT_TO_DOUBLE - Преобразование целого значения в вещественное двойной точности

Вход: IN1 INTEGER
Выход: Q DOUBLE

Описание:
Преобразовывает целое значение в вещественное двойной точности.
(* ST эквивалент: *)
odig1 := INT_TO_DOUBLE(100);

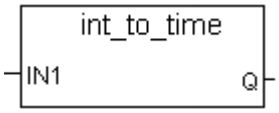
7.2.10 INT_TO_REAL - Преобразование целого значения в вещественное



Вход: IN1 INTEGER
Выход: Q REAL

Описание:
Преобразовывает целое значение в вещественное.
(* ST эквивалент: *)
real1 := INT_TO_REAL(100);

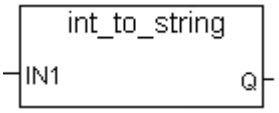
7.2.11 INT_TO_TIME - Преобразование целого значения в таймерное



Вход: IN1 INTEGER
Выход: Q TIMER

Описание:
Преобразовывает целое значение в таймер.
(* ST эквивалент: *)
timer := INT_TO_TIME(152);

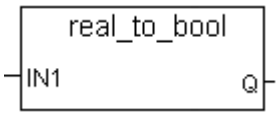
7.2.12 INT_TO_STRING - Преобразование целого значения в строку



Вход: IN1 INTEGER
Выход: Q MESSAGE

Описание:
Преобразовывает целое значение в строку.
(* ST эквивалент: *)
string := INT_TO_STRING(100);

7.2.13 REAL_TO_BOOL - Преобразование вещественного значения в булевское



Вход: IN1 REAL
Выход: Q BOOLEAN

Описание:
Преобразовывает вещественное значение в булевское.
(* ST эквивалент: *)
odig1 := REAL_TO_BOOL(100.0);

7.2.14 REAL_TO_BYTE - Преобразование вещественного значения в байтовое беззнаковое

Вход: IN1 REAL
Выход: Q BYTE

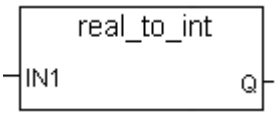
Описание:
Преобразовывает вещественное значение в байтовое беззнаковое.
(* ST эквивалент: *)
odig1 := REAL_TO_BYTE(100.0);

7.2.15 REAL_TO_DOUBLE - Преобразование вещественного значения в вещественное двойной точности

Вход: IN1 REAL
Выход: Q DOUBLE

Описание:
Преобразовывает вещественное значение в вещественное двойной точности.
(* ST эквивалент: *)
odig1 := REAL_TO_DOUBLE(100.0);

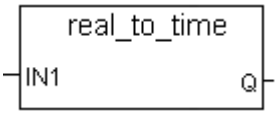
7.2.16 REAL_TO_INT - Преобразование вещественного значения в целое



Вход: IN1 REAL
Выход: Q INTEGER

Описание:
Преобразовывает вещественное значение в целое.
(* ST эквивалент: *)
int1 := REAL_TO_INT(100.0);

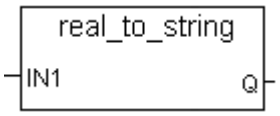
7.2.17 REAL_TO_TIME - Преобразование вещественного значения в таймерное



Вход: IN1 REAL
Выход: Q TIMER

Описание:
Преобразовывает вещественное значение в таймер.
(* ST эквивалент: *)
timer1 := REAL_TO_TIME(152.0);

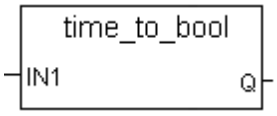
7.2.18 REAL_TO_STRING - Преобразование вещественного значения в строку



Вход: IN1 REAL
Выход: Q MESSAGE

Описание:
Преобразовывает вещественное значение в строку.
(* ST эквивалент: *)
string1 := REAL_TO_STRING(100.0);

7.2.19 TIME_TO_BOOL - Преобразование таймерного значения в булевское



Вход: IN1 TIMER
Выход: Q BOOLEAN

Описание:
Преобразовывает таймер в булевское значение.
(* ST эквивалент: *)
odig1 := TIME_TO_BOOL(T#100ms);

7.2.20 TIME_TO_BYTE - Преобразование таймерного значения в байтовое беззнаковое

Вход: IN1 TIMER
Выход: Q BYTE

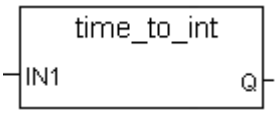
Описание:
Преобразовывает таймер в байтовое беззнаковое.
(* ST эквивалент: *)
odig1 := TIME_TO_BYTE(T#100ms);

7.2.21 TIME_TO_DOUBLE - Преобразование таймерного значения в вещественное двойной точности

Вход: IN1 TIMER
Выход: Q DOUBLE

Описание:
Преобразовывает таймер в вещественное двойной точности.
(* ST эквивалент: *)
odig1 := TIME_TO_DOUBLE(T#100ms);

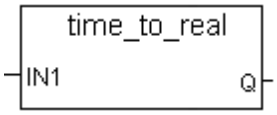
7.2.22 TIME_TO_INT - Преобразование таймерного значения в целое



Вход: IN1 TIMER
Выход: Q INTEGER

Описание:
Преобразовывает таймер в целое значение.
(* ST эквивалент: *)
count := TIME_TO_INT(T#100ms);

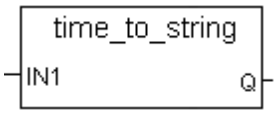
7.2.23 TIME_TO_REAL - Преобразование таймерного значения в вещественное



Вход: IN1 TIMER
Выход: Q REAL

Описание:
Преобразовывает таймер в вещественное значение.
(* ST эквивалент: *)
real1 := TIME_TO_REAL(T#100ms);

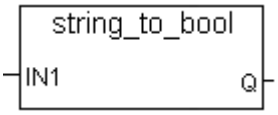
7.2.24 TIME_TO_STRING - Преобразование таймерного значения в строку



Вход: IN1 TIMER
Выход: Q MESSAGE

Описание:
Преобразовывает таймер в строку.
(* ST эквивалент: *)
string1 := TIME_TO_STRING(TIME#100ms);

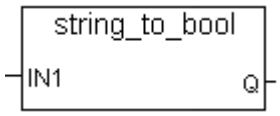
7.2.25 STRING_TO_BOOL - Преобразование строки в булевское значение



Вход: IN1 MESSAGE
Выход: Q BOOLEAN

Описание:
Преобразовывает строку в булевское значение.
(* ST эквивалент: *)
odig1 := STRING_TO_BOOL('true');

7.2.26 STRING_TO_BYTE - Преобразование строки в байтовое беззнаковое значение



Вход: IN1 MESSAGE
Выход: Q BYTE

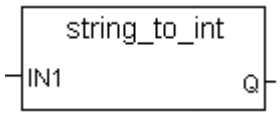
Описание:
Преобразовывает строку в булевское значение.
(* ST эквивалент: *)
odig1 := STRING_TO_BYTE('true');

7.2.27 STRING_TO_DOUBLE - Преобразование строки в булевское значение

Вход: IN1 MESSAGE
Выход: Q DOUBLE

Описание:
Преобразовывает строку в булевское значение.
(* ST эквивалент: *)
odig1 := STRING_TO_DOUBLE('true');

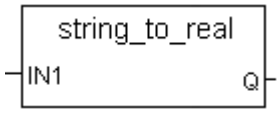
7.2.28 STRING_TO_INT - Преобразование строки в целое значение



Вход: IN1 MESSAGE
Выход: Q INTEGER

Описание:
Преобразовывает строку в целое значение.
(* ST эквивалент: *)
count := STRING_TO_INT('10');

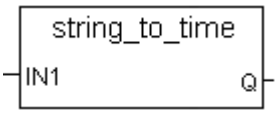
7.2.29 STRING_TO_REAL - Преобразование строки в вещественное значение



Вход: IN1 MESSAGE
Выход: Q REAL

Описание:
Преобразовывает строку в вещественное значение.
(* ST эквивалент: *)
count := STRING_TO_REAL('10.0');

7.2.30 STRING_TO_ TIME - Преобразование строки в таймерное значение



Вход: IN1 MESSAGE
Выход: Q TIMER

Описание:
Преобразовывает строку в таймерное значение.
(* ST Эквивалент: *)
time1 := STRING_TO_TIME('T#1S');

7.3 Функциональные блоки

В таблице приведены описанные в библиотеке Unimod Pro функциональные блоки и их реализация на различных типах модулей.

Код	Название	Описание	Мастер-ПК	M841E M921E M902E M903E M915E	M911	M900/M800
	Генерация сигналов					
4	BLINK	Мультивибратор	●	●	●	●
48	PULSE	Одновибратор	●	●	●	●
50	SIN_GEN	Генератор синусоидального сигнала	●	●	●	●
49	PSET	Программный задатчик	●	●		
	Динамические преобразования					
40	AVRGM	Среднее скользящее	●	●	●	●
41	AVRGD	Среднее дискретное	●	●	●	●
37	DFILTER	Дискретный фильтр	●	●	●	●
39	FILTER	Экспоненциальное сглаживание	●	●	●	●
38	PFILTER	Фильтр высокочастотной помехи	●	●	●	●
36	APERT	Апертура	●	●	●	●
42	INTEG	Интегрирование	●	●	●	●
43	DERIV	Дифференцирование	●	●	●	●
44	DDERIV	Вторая производная	●	●	●	●
46	DCNV	Динамическое преобразование	●	●	●	●
45	LRATE	Ограничение скорости	●	●	●	●
47	DELAY	Запаздывание	●	●	●	●
147	ONDTR	Таймер задержки по включению со сбросом		●		
	Статические преобразования					
23	CMP	Сравнение	●	●	●	●
32	LIMITR	Ограничение	●	●	●	●
33	INS0	Вставка нуля	●	●	●	●
28	MAXR4	Максимум из 4 входов	●	●	●	●
29	MAXR8	Максимум из 8 входов	●	●	●	●
30	MINR4	Минимум из 4 входов	●	●	●	●
31	MINR8	Минимум из 8 входов	●	●	●	●
26	MAX_MIN4	Поиск максимума и минимума из 4 входов	●	●	●	●
27	MAX_MIN8	Поиск максимума и минимума из 8 входов	●	●	●	●
34	EXTRM	Экстремум	●	●	●	●
35	ADDM	Суммирование с выделением модуля и знака	●	●	●	●
149	CMP_REAL	Сравнение вещественных значений		●		
	Аналого-дискретные преобразования					
21	HYSTER	Гистерезис по верхнему пределу	●	●	●	●
22	LIM_ALR	Гистерезис по верхнему и нижнему пределам	●	●	●	●
24	MUXD4	Дискретный переключатель на 4 входа	●	●	●	●
25	MUXD8	Дискретный переключатель на 8 входов	●	●	●	●
17	DCHNG	Запрет изменений	●	●	●	●

18	DSIGN	Запрет знака	●	●	●	●
19	FLW_HOLD	Слежение, запоминание	●	●	●	●
20	LATCH	Запоминание	●	●	●	●
	Дискретное управление					
13	SR	Установить доминанту	●	●	●	●
12	RS	Сбросить доминанту	●	●	●	●
14	RF_TRIG	Определение переднего/заднего фронта	●	●	●	●
102	F_TRIG	Определение заднего фронта		●		
16	SEMA	Семафор	●	●	●	●
8	CTU	Счетчик вверх	●	●	●	●
9	CTD	Счетчик вниз	●	●	●	●
10	CTUD	Счетчик вверх-вниз	●	●	●	●
5	TON	Время включения	●	●	●	●
6	TOF	Время выключения	●	●	●	●
7	TP	Время пульсирования	●	●	●	●
11	MAJOR	Мажорирование	●	●	●	●
15	DTRIG	D – триггер	●	●	●	●
	Автоматическое регулирование					
55	RAN	Регулирование аналоговое	●	●	●	●
56	RIM	Регулирование импульсное	●	●	●	●
53	SET	Управление заданием	●	●	●	●
52	LCTRL	Управление локальное	●	●	●	●
51	CCTRL	Управление каскадное	●	●	●	●
54	PWM	Широтно-импульсная модуляция	●	●	●	●
83	ZADV	Управления запорной арматурой				●
148	RAN_	Регулирование аналоговое (для LD)		●		
	Работа со временем					
57	TIMEDATE	Преобразование время-дата	●	●	●	●
58	DATETIME	Преобразование дата-время	●	●	●	●
59	GETTIME	Получение текущего времени	●	●	●	●
60	GETTIME_	Получение текущего времени (спец.)	●	●	○	○
61	SETTIME	Установка текущего времени	●	●	●	●
62	SETTIME_	Установка текущего времени (спец.)	●	●	○	○
104	GPS_GETTIME	Получение текущего времени		●		
109	DRV_GETTIME	Получение текущего времени с удаленного устройства		●		
165	DRV_SYNCTIME	Синхронизация текущего времени с временем удаленного устройства				
151	LOCALTIME	Получение текущего времени и даты (с днем недели)		●		
	Modbus функции					
63	MB_PARAM	Установка параметров Modbus		●	●	●
64	MB_R_C	Чтение состояния 8-ми бинарных ячеек памяти		●	●	●
65	MB_R_B	Чтение состояния 8-ми дискретных входов		●	●	●
66	MB_R_H	Чтение целочисленного регистра		●	●	●
67	MB_R_I	Чтение целочисленного входного регистра		●	●	●
68	MB_R_F	Чтение пары регистров в формате с плавающей точкой		●	●	●
69	MB_R_R16	Чтение 16 целочисленных регистров		●	●	○

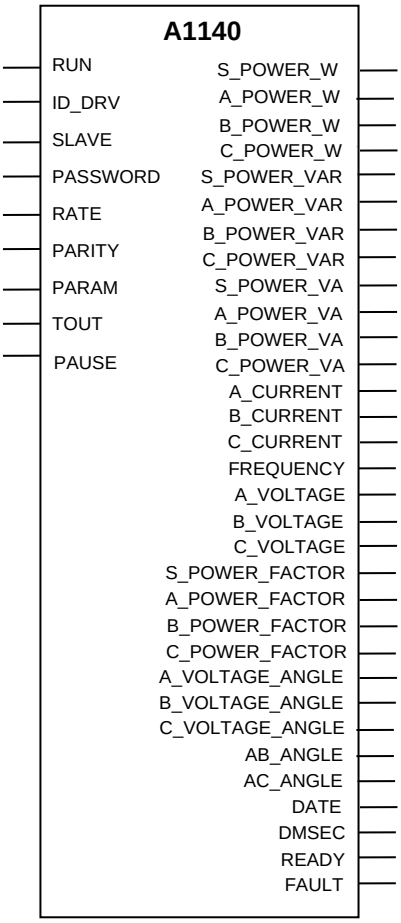
70	MB_R_F16	Чтение 16 пар регистров в формате с плавающей точкой		●	●	○
71	MB_W_C	Запись в бинарную ячейку памяти		●	●	●
72	MB_W_H	Запись целочисленного регистра		●	●	●
73	MB_W_F	Запись пары регистров в формате с плавающей точкой		●	●	●
74	MB_DIAG	Диагностика связи с подчиненным устройством		●	●	●
75	MB_STATE	Получение ID подчиненного устройства		●	●	●
88	MB_R	Чтение группы данных		●	●	○
89	MB_W	Запись группы данных		●	●	○
101	MB_RW	Чтение/запись регистров		●		
	OWEN функции					
106	OW_PARAM	Установка параметров ОВЕН		●		
107	OW_R	Чтение группы данных		●		
108	OW_W	Запись группы данных		●		
	Profibus-DP функции					
117	PB_PARAM	Установка параметров Profibus		●		
118	PB_CONTROL	Управление интерфейсом Profibus		●		
119	PB_READ	Чтение группы данных		●		
120	PB_WRITE	Запись группы данных		●		
	Библиотека регулирования и учета					
76	FLOW_R	Расчет расхода природного газа		●		○
156	FLOW_R_2005	Расчет расхода природного газа		●		○
77	AGA8_92	Расчет свойств природного газа (метод УС AGA8-92DC)		●		○
78	DS_CALC	Расчет плотности природного газа		●		○
160	DS_CALC_2015	Расчет плотности природного газа		●		○
79	GERG_91	Расчет свойств природного газа (метод GERG-91 мод.)		●		
80	NX_19	Расчет свойств природного газа (метод NX_19)		●		
81	VNIC_SMV	Расчет свойств природного газа (метод ВНИЦ СМВ)		●		○
105	MR_113	Расчет свойств природного газа (метод ГСССД МР 113-03)		●		○
161	MR_113_V2	Расчет свойств природного газа (метод ГСССД МР 113-03)		●		○
122	DENS_CALC	Расчет свойств нефти		●		○
123	MBRUTTO	Расчет свойств нефти		●		○
126	IF_97	Расчет свойств воды и пара		●		○
159	IF_97_2012	Расчет свойств воды и пара		●		○
131	AGA8_92_	Расчет свойств природного газа (метод УС AGA8-92DC) (модиф.)		●		○
132	DS_CALC_	Расчет плотности природного газа (модиф.)		●		○
133	GERG_91_	Расчет свойств природного газа (метод GERG-91 мод.)		●		
134	NX_19_	Расчет свойств природного газа (метод NX_19) (модиф.)		●		

135	VNIC_SMV_	Расчет свойств природного газа (метод ВНИЦ СМВ) (модиф.)		●		○
157	GOST_30319_2_2015	Расчет свойств природного газа		●		○
158	GOST_30319_3_2015	Расчет свойств природного газа		●		○
	HART функции					
90	HART_ADDR	Получение адреса устройства HART из его идентификационных данных	○	●	○	●
91	HART_0	Получения расширенного кода устройства HART, версии и идентификационного номера	○	●	○	●
92	HART_1	Считывание первичной переменной устройства HART	○	●	○	●
93	HART_2	Считывание тока и процента диапазона первичной переменной устройства HART	○	●	○	●
94	HART_3	Считывание четырех динамических переменных и тока первичной переменной устройства HART	○	●	○	●
95	HART_15	Считывание информации о выходе первичной переменной устройства HART	○	●	○	●
96	HART_35	Запись границ диапазона первичной переменной устройства HART	○	●	○	●
97	HART_41	Запуск самотестирования устройства HART.	○	●	○	●
98	HART_48	Считывание дополнительного статуса устройства HART	○	●	○	●
99	HART	Формирование пользовательского запроса к устройству HART	○	●	○	●
	Работа с сетевым принтером					
110	LPR	Печать файла		●		
111	LPQ	Чтение состояния принтера		●		
112	LPRM	Чистка очереди принтера		●		
	Разное					
1	OPERATE	Тестирование и переинициализация мезонина (юнита)	●	●	●	●
100	OPERATE_F	Тестирование и переинициализация мезонина (юнита)	●	●	●	●
2	OPERATEP	Управление пожарным извещателем				
3	SERV	Параметры физического входа		●		●
82	STATELPT	Статус LPT		●		○
103	LOAD_APPL	Загрузка/чтение приложения с модуля ввода/вывода		●		
124	PORT_OPEN	Открытие порта		●		○
125	PORT_CTRL	Управление портом		●		○
143	TRANSIT	Транзитный запрос		●		
145	INT_TO_BIT	Распаковка целого значения в булевские		●		
146	PORT_GATE	Открытие портов для транзита TCP\UDP запросов в COM		●		
152	SOCK_OPEN	Открытие сокета IP		●		

Использованные условные обозначения

Маркировка	Комментарий
●	Функция или функциональный блок реализован на данной платформе
○	Реализация невозможна (ограничение TIC-кода, отсутствие интерфейса или аппаратуры, для управления которыми предназначена данная функция или функциональный блок).
-	Прочерк обозначает, что данная функция или функциональный блок на данной платформе недоступна.
	Пустое поле обозначает, что функция или функциональный блок не реализованы, но реализация возможна.

7.3.1 A1140



Входы:

RUN	BOOLEAN	Признак выполнения функционального блока
ID_DRV	INTEGER	Идентификатор коммуникационного адаптера (COM1..COMn)
SLAVE	INTEGER	Адрес устройства A1140
PASSWORD	MESSAGE	Пароль (символьная строка – 8 символов)
RATE	INTEGER	Скорость передачи по последовательной линии (300..9600)
PARITY	INTEGER	Количество стоповых бит и режим контроля четности
PARAM	INTEGER	Режим работы
TOUT	INTEGER	Время ожидания ответа от Slave-устройства (мс)
PAUSE	INTEGER	Длительность паузы между передачами (мс)

Выходы:

S_Power_W	REAL	Активная мощность сети (кВт)
A_Power_W	REAL	Активная мощность фазы A (кВт)
B_Power_W	REAL	Активная мощность фазы B (кВт)
C_Power_W	REAL	Активная мощность фазы C (кВт)
S_Power_VAR	REAL	Реактивная мощность сети (кВАР)
A_Power_VAR	REAL	Реактивная мощность фазы A (кВАР)
B_Power_VAR	REAL	Реактивная мощность фазы B (кВАР)
C_Power_VAR	REAL	Реактивная мощность фазы C (кВАР)
S_Power_VA	REAL	Полная мощность сети (В-А)
A_Power_VA	REAL	Полная мощность фазы A (В-А)
B_Power_VA	REAL	Полная мощность фазы B (В-А)
C_Power_VA	REAL	Полная мощность фазы C (В-А)
A_Current	REAL	Ток фазы A (А)
B_Current	REAL	Ток фазы B (А)

C_Current	REAL	Ток фазы C (A)
Frequency	REAL	Частота сети (Гц)
A_Voltage	REAL	Напряжение фазы A (B)
B_Voltage	REAL	Напряжение фазы B (B)
C_Voltage	REAL	Напряжение фазы C (B)
S_PowerFactor	REAL	Коэффициент мощности сети
A_PowerFactor	REAL	Коэффициент мощности фазы A
B_PowerFactor	REAL	Коэффициент мощности фазы B
C_PowerFactor	REAL	Коэффициент мощности фазы C
A_VoltageAngle	REAL	Фазовый угол фазы A (рад.)
B_VoltageAngle	REAL	Фазовый угол фазы B (рад.)
C_VoltageAngle	REAL	Фазовый угол фазы C (рад.)
AB_Angle	INTEGER	Угол между фазами A и B (град.)
AC_Angle	INTEGER	Угол между фазами A и C (град.)
DATE	INTEGER	Дата (день,месяц,год)
DMSEC	INTEGER	Миллисек. с начала суток
READY	BOOLEAN	Признак завершения операции
FAULT	INTEGER	Код ошибки

Назначение:

Чтение показаний счетчика электрической энергии Альфа 1140 (Elster Метроника).
При этом на мастер-модуле должна быть запущена задача связи **elster**.

Описание:

Изменение установок происходит при единичном значении входа RUN. Новые установки вступают в силу незамедлительно.

Вход ID_DRV задаёт идентификатор физической линии COM1..COMn.

Аргументы RATE и PARITY используются для конфигурирования последовательной линии RS-485\RS-232.

Через вход RATE задается скорость обмена. RATE может принимать одно из следующих значений: 300, 600, 1200, 2400, 4800, 9600 бод.

Количество стоповых битов и режим контроля четности задается входом PARITY.

Возможные варианты режима контроля четности представлены в таблице 1.

Таблица 1 – Режим контроля четности

Значение	Количество стоповых бит	Контроль четности
0	1	Отключен
1	2	Отключен
2	1	EVEN
3	1	ODD
4	1	SPACE
5	1	MARK

Формат аргумента PARAM имеет следующий вид:

MSB X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X HDUP X **LSB**

Где: **HDUP** (разряд 1, half-duplex). Флаг контролирует режим обмена по линии RS-485. Сброшенный флаг HDUP устанавливает полнодуплексный режим обмена по двум дифференциальным линиям, отдельно для приема и передачи (RS-422). Установка флага HDUP разрешает при отсутствии посылки перевод

передатчика RS-485 в третье состояние; таким образом, становится возможна двунаправленный обмен пакетами по одной дифференциальной линии – режим half-duplex.

Параметры TOUT и PAUSE задают время ожидания ответа от A1140 -устройства и задержку после получения ответа и перед следующей передачей соответственно. Все значения задаются в миллисекундах.

Если входы PASSWORD, RATE, PARITY, TOUT, PAUSE имеют нулевые значения, то применяются значения “по – умолчанию”.

После завершения операции, выход функционального блока Ready устанавливается в состояние TRUE, а на выходе FAULT - код ошибки или нуль, если вызов отработан без ошибок.

Значения выхода FAULT (Код ошибки):

0 - Выполнено успешно

1 - Ошибка инициализации (некорректные параметры)

2 - Удаленное устройство не отвечает (таймаут истек)

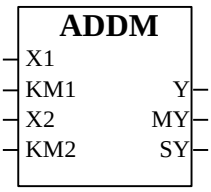
3 - Системная ошибка при работе с портом

4 - Системная ошибка при обращении к задаче связи (некорректный идентификатор устройства)

5 - Выполняется запрос

6 - Ошибка принятых данных

7.3.2 ADDM



Входы:

X1	REAL	1-й масштабируемый вход
KM1	REAL	1-й масштабный коэффициент
X2	REAL	2-й масштабируемый вход
KM2	REAL	2-й масштабный коэффициент

Выходы:

Y	REAL	Основной выход Y=X1*KM1+X2*KM2
MY	REAL	Модуль суммы
SY	BOOLEAN	Знак суммы: FALSE - положительная, TRUE – отрицательная

Назначение

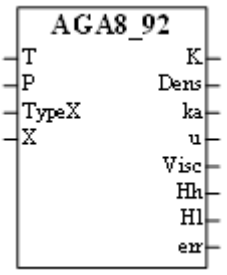
Функция ADDM применяется для суммирования двух чисел с выделением знака и модуля числа на отдельных выходах.

Описание алгоритма

Функциональный блок осуществляет масштабирование и суммирование двух входных сигналов X1 и X2. На выходах блока формируется три сигнала:

1. Сигнал Y=KM1*X1+KM2*X2.
2. Сигнал MY, равный модулю сигнала Y, MY=|Y|.
3. Сигнал SY, равный знаку сигнала Y (при Y≥0 SY=FALSE, иначе SY=TRUE).

7.3.2 AGA8_92



Входы:

t	REAL	Температура газа (град.С)
P	REAL	Давление газа (МПа)
TypeX	BOOLEAN	Тип компонентного состава газа: FALSE – в молярных %; TRUE – в объемных %.
X	#REAL	Имя массива с компонентным составом газа (элементы типа Real)

Выходы:

K	REAL	Коэффициент сжимаемости
Dens	REAL	Плотность газа при рабочих условиях (кг / куб.м)
ka	REAL	Показатель адиабаты
u	REAL	Скорость звука в газе (м / с)
Visc	REAL	Динамическая вязкость (мкПа * с)
Hh	REAL	Высшая удельная теплота сгорания (МДж / куб.м)
Hl	REAL	Низшая удельная теплота сгорания (МДж / куб.м)
err	INTEGER	Код ошибки: 0 – нет ошибки; 3 – недопустимый размер массива; 4 – недопустимая ссылка на массив; -1 – недопустимое значение температуры; -2 – недопустимое значение давления; -8 – недопустимое значение доли азота; -16 – недопустимое значение доли углекислого газа; -32 – недопустимое значение доли метана; -64 – недопустимое значение доли этана; -128 – недопустимое значение доли пропана; -256 – недопустимое значение доли бутана; -512 – недопустимое значение доли сероводорода; -1024 – недопустимое значение доли остальных компонентов.

Назначение

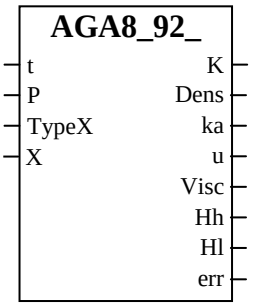
Функциональный блок используется для расчета свойств природного газа по методу УС AGA8_92DC (ГОСТ 30319.2-96).

Вход X содержит концентрации компонентов газа в следующем порядке:

X[1]	Метан
X[2]	Этан
X[3]	Пропан
X[4]	н-Бутан
X[5]	и-Бутан
X[6]	Азот
X[7]	Диоксид углерода
X[8]	Сероводород
X[9]	Ацетилен
X[10]	Этилен
X[11]	Пропилен
X[12]	н-Пентан
X[13]	и-Пентан

X[14]	н-Гексан
X[15]	Бензол
X[16]	н-Гептан
X[17]	Толуол
X[18]	н-Октан
X[19]	Гелий
X[20]	Водород
X[21]	Монооксид углерода
X[22]	Кислород

7.3.3 AGA8_92_



Входы:

t	REAL	Температура газа (град.С)
P	REAL	Давление газа (МПа)
TypeX	BOOLEAN	Тип компонентного состава газа: FALSE – в молярных %; TRUE – в объемных %.
X	#REAL	Имя массива с компонентным составом газа (элементы типа Real)

Выходы:

K	REAL	Коэффициент сжимаемости
Dens	REAL	Плотность газа при рабочих условиях (кг / куб.м)
ka	REAL	Показатель адиабаты
u	REAL	Скорость звука в газе (м / с)
Visc	REAL	Динамическая вязкость (мкПа * с)
Hh	REAL	Высшая удельная теплота сгорания (МДж / куб.м)
Hl	REAL	Низшая удельная теплота сгорания (МДж / куб.м)
err	INTEGER	Код ошибки: 0 – нет ошибки; 3 – недопустимый размер массива; 4 – недопустимая ссылка на массив; -1 – недопустимое значение температуры; -2 – недопустимое значение давления; -8 – недопустимое значение доли азота; -16 – недопустимое значение доли углекислого газа; -32 – недопустимое значение доли метана; -64 – недопустимое значение доли этана; -128 – недопустимое значение доли пропана; -256 – недопустимое значение доли бутана; -512 – недопустимое значение доли сероводорода; -1024 – недопустимое значение доли остальных компонентов.

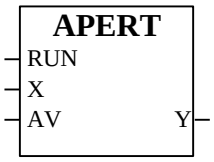
Назначение

Функциональный блок используется для расчета свойств природного газа по методу УС AGA8_92DC (ГОСТ 30319.2-96, Изменение №1 к ГОСТ 30319.1-96 от 01.06.2004).

Вход X содержит концентрации компонентов газа в следующем порядке:

X[1]	Метан
X[2]	Этан
X[3]	Пропан
X[4]	н-Бутан
X[5]	и-Бутан
X[6]	н-Пентан
X[7]	и-Пентан
X[8]	н-Гексан
X[9]	н-Гептан
X[10]	н-Октан
X[11]	Ацетилен
X[12]	Этилен
X[13]	Пропилен
X[14]	Бензол
X[15]	Толуол
X[16]	Водород
X[17]	Водяной пар
X[18]	Аммиак
X[19]	Метанол
X[20]	Сероводород
X[21]	Метилмеркаптан
X[22]	Диоксид серы
X[23]	Гелий
X[24]	Неон
X[25]	Аргон
X[26]	Монооксид углерода
X[27]	Азот
X[28]	Воздух
X[29]	Кислород
X[30]	Диоксид углерода

7.3.4 *APERT*



Входы:

RUN	BOOLEAN	Режим работы: TRUE - нормальный, FALSE - отключенный
X	REAL	Основной вход
AV	REAL	Величина апертуры

Выход:

Y	REAL	Основной выход

Назначение

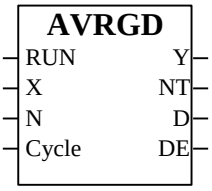
Функциональный блок APERT используется для фильтрации небольших изменений значения параметра (например, дребезг последнего бита в АЦП).

Описание алгоритма

При состоянии входа RUN=TRUE блок не пропускает на выход малые изменения сигнала. Если выполняется соотношение: $|X-Y| \geq AV$, то значение выхода изменится и будет равно входу Y=X. Если $|X-Y| < AV$ значение выхода не изменится и останется прежним.

При состоянии входа RUN=FALSE выход всегда равен входу (Y=X) независимо от значений X и AV.

7.3.5 *AVRGD*



Входы:

RUN	BOOLEAN	Режим работы: TRUE -нормальный, FALSE -отключенный
X	REAL	Основной вход
N	INTEGER	Заданное число отсчетов
Cycle	INTEGER	Период отсчета (мсек)

Выходы:

Y	REAL	Основной выход
NT	INTEGER	Номер текущего отсчета
D	BOOLEAN	Признак отсчета
DE	BOOLEAN	Признак окончания отсчетов

Назначение

Функциональный блок AVRGD применяется для усреднения аналогового сигнала на фиксированном отрезке времени за заданное число отсчетов.

Описание алгоритма

В алгоритме суммируются значения входного сигнала, полученные за N отсчетов, после чего полученная сумма делится на N. Число отсчетов, за которое усредняется выходной сигнал, задается входным параметром N.

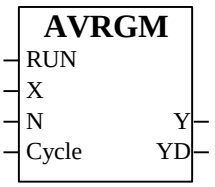
Счетчик внутри блока считает текущее число отсчетов NT и в момент, когда NT=N среднее значение, вычисленное сумматором, запоминается и обновляется сигнал на выходе Y. Тем самым заканчивается очередной интервал усреднения, после чего вплоть до окончания нового интервала сигнал Y не изменяется.

На выходе NT формируется текущий (внутри интервала) номер отсчета. Отсчеты производятся через период времени Cycle (входной параметр блока). На выходе D в начале каждого отсчета в течение одного цикла работы контроллера устанавливается D=TRUE, в остальное время D=FALSE. По окончанию интервала усреднения устанавливается сигнал в течение времени Cycle на выходе DE=TRUE (до окончания этого интервала DE=FALSE).

При состоянии входа RUN=FALSE или при значении N≤0 выходной сигнал Y=X, т.е. выходной сигнал равен текущему значению входного сигнала. Сумматор и счетчик циклов обнуляются. Выходы D, DE, NT равны нулю.

Примечание: При значении Cycle ≤0 период отсчета равен циклу контроллера.

7.3.6 *AVRGM*



Входы:

RUN	BOOLEAN	Режим работы: TRUE - нормальный, FALSE - отключенный
X	REAL	Основной вход
N	INTEGER	Количество циклов запаздывания от 1 до 128
Cycle	INTEGER	Период работы блока (мсек)

Выходы:

Y	REAL	Основной выход
YD	REAL	Запаздывающее значение входа

Назначение

Функциональный блок AVRGM применяется для вычисления среднего из нескольких (до 128) последних значений аналогового сигнала.

Описание алгоритма

Структура и работа алгоритма AVRGM соответствует алгоритму запаздывания DELAY, т.е. значение входного сигнала X заносится в массив и будет прочитано из этого массива через N циклов. Отличие алгоритма AVRGM заключается в наличии дополнительного сумматора, который усредняет значение сигнала на выходе. При состоянии входа RUN=TRUE выходной сигнал Y равен:

$$Y = \frac{\sum_{i=1}^{N} X_i}{N},$$

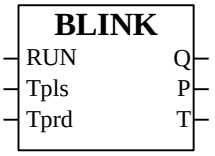
где $1 \le N \le 128$ – количество циклов запаздывания; X_i - значение сигнала X в i-ом цикле запаздывания.

Выход YD равен запаздывающему значению входа. Время запаздывания равно N·Cycle, где Cycle – период работы блока (длительность одного цикла запаздывания).

Если значение входа N≤0 или RUN=FALSE выходы алгоритма равны текущему значению входа Y=X и YD=X. В момент перехода входного сигнала RUN из состояния FALSE в состояние TRUE все N элементов массива инициализируются текущим значением входа X.

Примечание: Изменение значения N при состоянии входа RUN=TRUE алгоритмом не воспринимается, поэтому значение N можно изменять только при состоянии RUN=FALSE. Если значение параметра Cycle меньше времени цикла контроллера (или Cycle ≤0), то период работы блока равен циклу контроллера. Значение N>128 алгоритм воспринимает как N=128.

7.3.7 *BLINK*



Входы:

RUN	BOOLEAN	Режим работы: TRUE - пуск, FALSE - стоп
Tpls	INTEGER	Длительность импульсов (мсек)
Tprd	INTEGER	Период импульсов (мсек)

Выходы:

Q	BOOLEAN	Основной выход

P	INTEGER	Текущее время импульса (мсек)
T	INTEGER	Текущее время периода (мсек)

Назначение:

Функциональный блок BLINK применяется для периодического включения оборудования (двигателя, нагревателя, обеспечения мигающей сигнализации и т.п.).

Описание алгоритма

Алгоритм формирует последовательность импульсов с заданной длительностью и периодом следования. При состоянии входа RUN=TRUE с выхода блока Q формируются импульсы с длительностью Tpls и периодом следования Tprd. Фактическая длительность импульсов, также как и период следования **кратны циклу контроллера**.

Состояние основного выхода Q в режиме пуска (RUN=TRUE) при различных значениях Tpls и Tprd определяется таблицей:

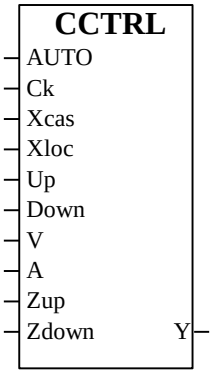
Tpls	Tprd	Выход Q
Tpls>0	Tprd>Tc	__П_П__П_П_
Tpls≤0	Безразлично	0
Tpls>0	Tprd≤Tc	1

Tc – время цикла контроллера.

На выходе P формируется время, отсчитываемое от переднего фронта импульса. В следующем такте после отработки импульса P=0. На выходе T формируется текущее время, отсчитываемое от начала нового периода. Если заданная длительность периода меньше времени цикла котроллера Tprd<Tc, то выход T равен нулю.

При переходе входного сигнала RUN в состояние FALSE последовательность импульсов прерывается, выход Q равен логическому нулю. Выходы P и T обнуляются.

7.3.8 *CCTRL*



Входы:

AUTO	BOOLEAN	Режим работы: TRUE - автоматический, FALSE -ручной
Ck	BOOLEAN	Ручной режим работы: TRUE -каскадный, FALSE -локальный
Xcas	REAL	Значение выхода блока в автоматическом режиме
Xloc	REAL	Значение выхода блока в ручном локальном режиме
Up	BOOLEAN	TRUE - увеличение значения выхода в ручном каскадном режиме
Down	BOOLEAN	TRUE - уменьшение значения выхода в ручном каскадном режиме
V	REAL	Скорость выхода в ручном каскадном режиме (ед./сек)
A	REAL	Ускорение выхода в ручном каскадном режиме (ед./сек^2)
Zup	BOOLEAN	Запрет в направлении "Больше"
Zdown	BOOLEAN	Запрет в направлении "Меньше"

Выход:

Y	REAL	Основной выход

Назначение

Функциональный блок CCTRL используется в составе каскадного регулятора, если необходимо ручное управление аналоговым регулятором.

Описание алгоритма.

Блок работает в одном из трех режимов: автоматическом, ручном каскадном и ручном локальном. Автоматический режим определяется состоянием входа AUTO=TRUE, при этом состояние входа Ck игнорируется. Ручной каскадный режим определяется состоянием входов AUTO=FALSE и Ck=TRUE. Ручной локальный режим определяется состоянием входов AUTO=FALSE и Ck=FALSE.

Выход блока Y в автоматическом режиме (AUTO=TRUE) равен входному сигналу Xcas. В ручном локальном режиме (AUTO=FALSE и Ck=FALSE) выход блока Y равен входному сигналу Xloc.

В ручном каскадном режиме выход блока изменяется в соответствии с состоянием дискретных входов Up и Down. Если Up=Down (оба входа в состоянии логического нуля или логической единицы), то выход Y не изменяется. Если Up=TRUE выходной сигнал Y увеличивается:

$$Y_i = Y_{i-1} + h \cdot Tc,$$

где Y_{i-1} – значение выхода в предыдущем цикле, Y_i – обновленное значение выхода в текущем цикле, h – скорость изменения значения выходного сигнала, Tc – длительность цикла.

Если Down=TRUE выходной сигнал Y уменьшается:

$$Y_i = Y_{i-1} - h \cdot Tc,$$

Скорость изменения значения выходного сигнала зависит от значения входных параметров V и A. Если $A \leq 0$, скорость h=V. Если A>0, то при наличии сигнала на входах Up или Down скорость h изменяется в каждом цикле контроллера в соответствии со следующей зависимостью:

$$h_i = h_{i-1} + A \cdot Tc,$$

где h_{i-1} – значение скорости в предыдущем цикле, h_i – значение скорости в текущем цикле.

В момент изменения состояния входов Up или Down скорость h=V.

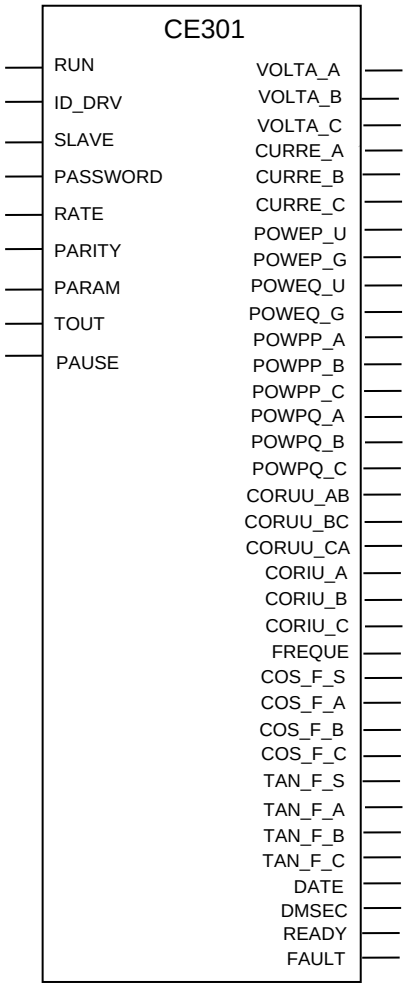
В блоке предусмотрены дискретные входы Zup и Zdown. При Zup=TRUE выходной сигнал Y не будет увеличиваться независимо от состояния входа Up. При Zdown=TRUE выходной сигнал не будет уменьшаться независимо от состояния входа Down. Состояние сигналов запрета анализируется алгоритмом только в ручном каскадном режиме.

На вход Xcas подается сигнал с основного выхода ведущего аналогового регулятора RAN, на вход Xloc подается значение регулируемого параметра (вход X2) ведомого регулятора. Сигнал с выхода блока CCTRL подается на вход X0 ведущего регулятора RAN. Таким образом, в ручном каскадном режиме с помощью дискретных сигналов Up и Down блока CCTRL можно изменять значение выходного сигнала Y блока RAN. В ручном локальном режиме выход ведущего регулятора RAN будет равен регулируемому параметру ведомого регулятора, т.е. входные сигналы X1 и X2 ведомого регулятора будут равны. Следовательно, рассогласование для ведомого регулятора будет равно нулю.

В момент переключения на ручной каскадный режим выход блока CCTRL равен выходу RAN. Таким образом, в момент переключения выход ведущего регулятора RAN не изменится и удара не произойдет.

Примечание: значение V<0 воспринимается алгоритмом как V=0.

7.3.9 CE301



Входы:

RUN	BOOLEAN	Признак выполнения функционального блока
ID_DRV	INTEGER	Идентификатор коммуникационного адаптера (COM1..COMn)
SLAVE	MESSAGE	Адрес устройства CE301
PASSWORD	MESSAGE	Пароль
RATE	INTEGER	Скорость передачи по последовательной линии (300..9600)
PARITY	INTEGER	Количество стоповых бит и режим контроля четности
PARAMINTEGER		Режим работы
TOUT	INTEGER	Время ожидания ответа от Slave-устройства (мс)
PAUSE	INTEGER	Длительность паузы между передачами (мс)

Выходы:

VOLTA_A	REAL	Действующее значение напряжения по фазе A (Вольт)
VOLTA_B	REAL	Действующее значение напряжения по фазе B (Вольт)
VOLTA_C	REAL	Действующее значение напряжения по фазе C (Вольт)
CURRE_A	REAL	Действующее значение тока по фазе A (Ампер)
CURRE_B	REAL	Действующее значение тока по фазе B (Ампер)
CURRE_C	REAL	Действующее значение тока по фазе C (Ампер)
POWEP_U	REAL	Мгновенное значение потребленной мощности (активная) (кВт)
POWEP_G	REAL	Мгновенное значение поставленной мощности (активная) (кВт)
POWEQ_U	REAL	Мгновенное значение потребленной мощности (реактивная) (квар)
POWEQ_G	REAL	Мгновенное значение поставленной мощности (реактивная) (квар)

POWPP_A	REAL	Мгновенное значение активной мощности по фазе A (кВт)
POWPP_B	REAL	Мгновенное значение активной мощности по фазе B (кВт)
POWPP_C	REAL	Мгновенное значение активной мощности по фазе C (кВт)
POWPQ_A	REAL	Мгновенное значение реактивной мощности по фазе A (квар)
POWPQ_B	REAL	Мгновенное значение реактивной мощности по фазе B (квар)
POWPQ_C	REAL	Мгновенное значение реактивной мощности по фазе C (квар)
CORUU_AB	REAL	Угол между векторами напряжений фаз A и B (град.)
CORUU_BC	REAL	Угол между векторами напряжений фаз B и C (град.)
CORUU_CA	REAL	Угол между векторами напряжений фаз C и A (град.)
CORIU_A	REAL	Угол между векторами тока и напряжения фазы A
CORIU_B	REAL	Угол между векторами тока и напряжения фазы B
CORIU_C	REAL	Угол между векторами тока и напряжения фазы C
FREQU	REAL	Частота сети (Гц)
COS_F_S	REAL	Коэффициент мощности суммарный
COS_F_A	REAL	Коэффициент мощности фазы A
COS_F_B	REAL	Коэффициент мощности фазы B
COS_F_C	REAL	Коэффициент мощности фазы C
TAN_F_S	REAL	Коэффициент реактивной мощности суммарный
TAN_F_A	REAL	Коэффициент реактивной мощности фазы A
TAN_F_B	REAL	Коэффициент реактивной мощности фазы B
TAN_F_C	REAL	Коэффициент реактивной мощности фазы C
DATE	INTEGER	Дата (день,месяц,год)
DMSEC	INTEGER	Миллисек. с начала суток
Ready	BOOLEAN	Признак завершения операции
Fault	INTEGER	Код ошибки

Назначение:
Чтение показаний счетчика электрической энергии CE301 (ЭНЕРГОМЕРА)
При этом на мастер-модуле должна быть запущена задача связи **ce301**.

Описание:

Изменение установок происходит при единичном значении входа RUN. Новые установки вступают в силу незамедлительно.

Вход ID_DRV задаёт идентификатор физической линии COM1..COMn.

Аргументы RATE и PARITY используются для конфигурирования последовательной линии RS-485\RS-232.

Через вход RATE задается скорость обмена. RATE может принимать одно из следующих значений: 300, 600, 1200, 2400, 4800, 9600 бод.

Количество стоповых битов и режим контроля четности задается входом PARITY.

Возможные варианты режима контроля четности представлены в таблице 1.

Таблица 1 – Режим контроля четности

Значение	Количество стоповых бит	Контроль четности
0	1	Отключен
1	2	Отключен
2	1	EVEN
3	1	ODD
4	1	SPACE
5	1	MARK

Формат аргумента PARAM имеет следующий вид:

MSB X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X HDUP X **LSB**

Где: **HDUP** (разряд 1, half-duplex). Флаг контролирует режим обмена по линии RS-485. Сброшенный флаг HDUP устанавливает полнодуплексный режим обмена по двум дифференциальным линиям, отдельно для приема и передачи (RS-422). Установка флага HDUP разрешает при отсутствии посылки перевод передатчика RS-485 в третье состояние; таким образом, становится возможна двунаправленный обмен пакетами по одной дифференциальной линии – режим half-duplex.

Параметры TOUT и PAUSE задают время ожидания ответа от CE301 -устройства и задержку после получения ответа и перед следующей передачей соответственно. Все значения задаются в миллисекундах.

Если входы PASSWORD, RATE, PARITY, TOUT, PAUSE имеют нулевые значения, то применяются значения “по – умолчанию”.

После завершения операции, выход функционального блока Ready устанавливается в состояние TRUE, а на выходе FAULT - код ошибки или нуль, если вызов отработан без ошибок.

Значения выхода FAULT (Код ошибки):

0 - Выполнено успешно

1 - Ошибка инициализации (некорректные параметры)

2 - Удаленное устройство не отвечает (таймаут истек)

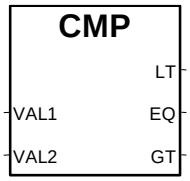
3 - Системная ошибка при работе с портом

4 - Системная ошибка при обращении к задаче связи (некорректный идентификатор устройства)

5 - Выполняется запрос

6 - Ошибка принятых данных

7.3.10 CMP



Входы:

VAL1	INTEGER	Любое знаковое целое аналоговое значение
VAL2	INTEGER	Любое знаковое целое аналоговое значение

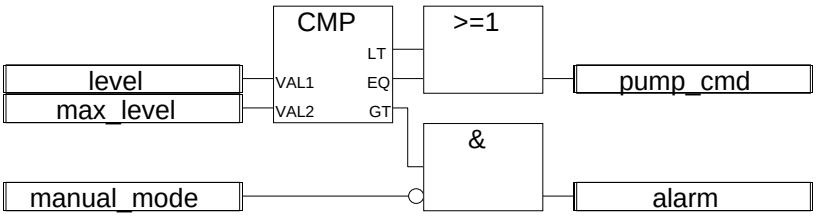
Выходы:

LT	BOOLEAN	TRUE если val1 меньше чем val2
EQ	BOOLEAN	TRUE если val1 равно val2
GT	BOOLEAN	TRUE если val1 больше чем val2

Описание:

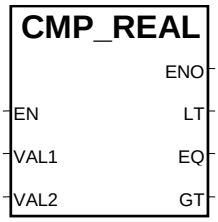
Сравнивает две величины: сообщает они равны или первая больше или меньше второй.

(*FBD пример блоков CMP*)




```
(* ST Эквивалент: Мы предполагаем, что CMP1 - это экземпляр блока CMP *)
CMP1(level, max_level);
pump_cmd := CMP1.LT OR CMP1.EQ;
alarm := CMP1.GT AND (NOT(manual_mode));
```

7.3.11 CMP_REAL



Входы:

EN	BOOLEAN	Вход разрешения
VAL1	REAL	Любое вещественное аналоговое значение
VAL2	IREAL	Любое вещественное аналоговое значение

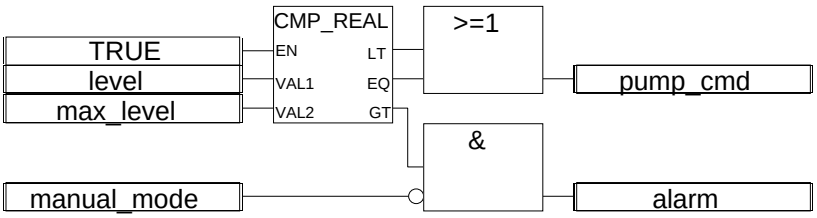
Выходы:

ENO	BOOLEAN	=EN
LT	BOOLEAN	TRUE если val1 меньше чем val2
EQ	BOOLEAN	TRUE если val1 равно val2
GT	BOOLEAN	TRUE если val1 больше чем val2

Описание:

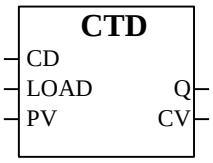
Сравнивает две величины: сообщает они равны или первая больше или меньше второй.

(*FBD пример блоков CMP_REAL*)




```
(* ST Эквивалент: Мы предполагаем, что CMP1 - это экземпляр блока CMP_REAL *)
CMP1(TRUE, level, max_level);
pump_cmd := CMP1.LT OR CMP1.EQ;
alarm := CMP1.GT AND (NOT(manual_mode));
```

7.3.12 *CTD*



Входы:

CD	BOOLEAN	Команда "считать"
LOAD	BOOLEAN	Команда "загрузить"
PV	INTEGER	Начальное значение

Выходы:

Q	BOOLEAN	Переполнение счетчика
CV	INTEGER	Результат счета

Назначение

Функциональный блок CTD представляет собой счетчик, содержимое которого можно уменьшать на 1 в каждом цикле работы контроллера. Совместно с блоком RF_TRIG блок CTD используется для подсчета числа дискретных событий (переход из состояния логического 0 в состояние логической 1 или обратных переходов).

Описание алгоритма

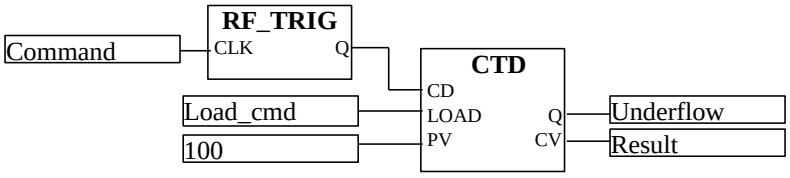
При состоянии входа CD=TRUE содержимое счетчика уменьшается до 0. Содержимое счетчика уменьшается на 1 в каждом цикле работы контроллера. Выход CV равен результату счета в текущем цикле. Если CD=FALSE содержимое счетчика не изменяется.

По команде “загрузить” (LOAD=TRUE) в счетчик записывается начальное значение PV. До тех пор, пока LOAD=TRUE, выход CV=PV.

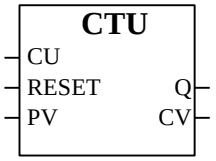
Выход Q принимает значение TRUE в тот момент, когда произошло переполнение счетчика, т.е. результат счета равен CV≤0.

Примечание: блок CTD не определяет передний и задний фронты входа (CD). Для того чтобы создать импульсный счетчик, на вход CD необходимо подавать сигнал с выхода блока RF_TRIG, который выделяет передний и задний фронты дискретного сигнала.

На рисунке приведен пример соединения блоков. Содержимое счетчика будет уменьшаться на 1 по переднему фронту сигнала **command**. Если установить инверсию на входе CLK блока RF_TRIG, содержимое счетчика будет уменьшаться по заднему фронту сигнала **command**.



7.3.13 *CTU*



Входы:

CU	BOOLEAN	Команда "считать"
RESET	BOOLEAN	Команда "сброс"
PV	INTEGER	Максимальное значение

Выходы:

Q	BOOLEAN	Переполнение счетчика
CV	INTEGER	Результат счета

Назначение

Функциональный блок CTU представляет собой счетчик, содержимое которого можно увеличивать на 1 в каждом цикле работы контроллера. Совместно с блоком RF_TRIG блок CTD используется для подсчета числа дискретных событий (переход из состояния логического 0 в состояние логической 1 или обратных переходов).

Описание алгоритма

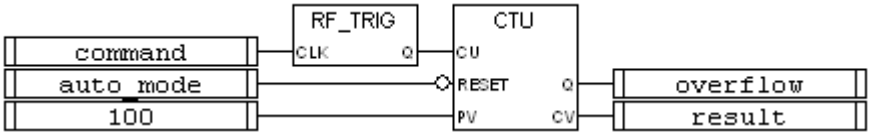
При состоянии входа CU=TRUE содержимое счетчика увеличивается до максимального значения PV. Содержимое счетчика увеличивается на 1 в каждом цикле работы контроллера. Выход CV равен результату счета в текущем цикле. Если CU=TRUE содержимое счетчика не изменяется.

По команде “сброс” (RESET=TRUE) содержимое счетчика обнуляется. До тех пор, пока RESET=TRUE, выход CV=0.

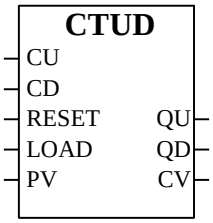
Выход Q принимает значение TRUE в тот момент, когда произошло переполнение счетчика, т.е. результат счета равен максимальному значению PV или PV≤0.

Примечание: блок CTU не определяет передний и задний фронты входа (CU). Для того чтобы создать импульсный счетчик, на вход CU необходимо подавать сигнал с выхода блока RF_TRIG, который выделяет передний и задний фронты дискретного сигнала.

Ниже приведен пример соединения блоков. Содержимое счетчика будет увеличиваться на 1 по переднему фронту сигнала **command**. Если установить инверсию на входе CLK блока RF_TRIG, содержимое счетчика будет увеличиваться по заднему фронту сигнала **command**.



7.3.14 *CTUD*



Входы:

CU	BOOLEAN	Команда на увеличение
CD	BOOLEAN	Команда на уменьшение
RESET	BOOLEAN	Команда "сброс"
LOAD	BOOLEAN	Команда "загрузить"
PV	INTEGER	Максимальное значение

Выходы:

QU	BOOLEAN	Переполнение счетчика: CV=PV
QD	BOOLEAN	Переполнение счетчика: CV=0
CV	INTEGER	Результат счета

Назначение

Функциональный блок CTUD представляет собой счетчик, содержимое которого можно уменьшать или увеличивать на 1 в каждом цикле работы контроллера. Совместно с блоком RF_TRIG блок CTUD используется для подсчета числа дискретных событий (переход из состояния логического 0 в состояние логической 1 или обратных переходов).

Описание алгоритма

При состоянии входа CU=TRUE содержимое счетчика увеличивается до максимального значения PV, при состоянии входа CD=TRUE уменьшается до 0. Содержимое счетчика увеличивается или уменьшается на 1 в каждом цикле работы контроллера. Выход CV равен результату счета в текущем цикле. Если CU=CD содержимое счетчика не изменяется.

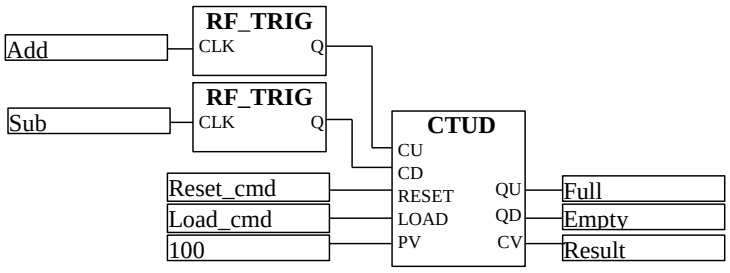
Работа счетчика разрешается, если отсутствуют сигналы "загрузить" и "сброс". По команде “загрузить” (LOAD=TRUE) в счетчик записывается значение PV. По команде “сброс” (RESET=TRUE) содержимое счетчика обнуляется, если нет команды “загрузить”.

Выход QU принимает значение TRUE в тот момент, когда произошло переполнение счетчика и результат счета равен максимальному значению PV или PV≤0.

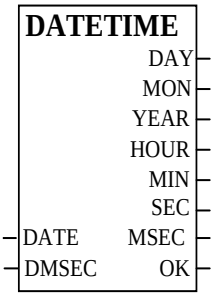
Выход QD принимает значение TRUE в тот момент, когда произошло переполнение счетчика и результат счета равен CV≤0.

Примечание: блок CTUD не определяет передний и задний фронты входов CU и CD. Для того чтобы создать импульсный счетчик, на входы CU и CD необходимо подавать сигнал с выходов двух блоков RF_TRIG, которые будут выделять передний и задний фронты дискретного сигнала.

На рисунке приведен пример соединения блоков. Содержимое счетчика будет увеличиваться или уменьшаться на 1 по переднему фронту сигналов **Add** или **Sub** соответственно. Если установить инверсию на входы CLK блоков RF_TRIG, содержимое счетчика будет изменяться по заднему фронту сигналов **Add** или **Sub**.



7.3.15 DATETIME



Входы:

DATE	INTEGER	Дата (день, месяц, год)
DMSEC	INTEGER	Миллисекунда с начала суток

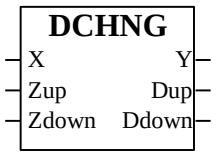
Выходы:

DAY	INTEGER	День с начала месяца
MON	INTEGER	Месяц с начала года
YEAR	INTEGER	Год
HOUR	INTEGER	Час с начала суток
MIN	INTEGER	Минута с начала часа
SEC	INTEGER	Секунда с начала минуты
MSEC	INTEGER	Миллисекунда с начала секунды
OK	INTEGER	Результат операции (0 - выполнено успешно)

Назначение

Преобразование времени из упакованного в развернутый формат.

7.3.16 *DCHNG*



Входы:

X	REAL	Основной вход
Zup	BOOLEAN	Команда запрета вверх
Zdown	BOOLEAN	Команда запрета вниз

Выходы:

Y	REAL	Основной выход
Dup	BOOLEAN	Признак запрета вверх
Ddown	BOOLEAN	Признак запрета вниз

Назначение

Функция DCHNG применяется для запрета увеличения или уменьшения сигнала, например, при действии технологических защит. В частности, запрет может блокировать изменение сигнала задания или выходного сигнала аналогового регулятора.

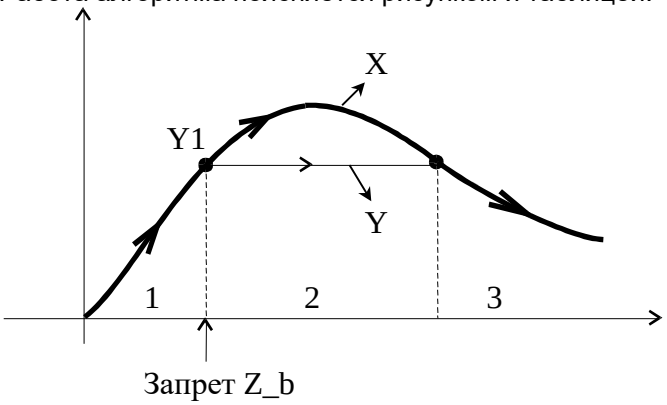
Описание алгоритма

Алгоритм содержит узел запрета, который реагирует на дискретные сигналы запрета в направлении увеличения (Zup - вверх) или уменьшения (Zdown - вниз). Помимо основного выхода Y алгоритм содержит два дискретных выхода Dup и Ddown, сигнализирующих о том, что алгоритм работает в режиме запрета.

Если команды запрета отсутствуют, то выходной сигнал равен входному Y =X. Если поступила команда Zup и X>Y, то алгоритм переходит в режим "запрета вверх", при этом Y =Y1 =const и Dup =1, где Y1 - значение Y в момент поступления команды запрета. Этот режим сохраняется до тех пор, пока входной сигнал не уменьшится до значения X<Y1, после чего режим запрета снимается и вновь Y=X.

Аналогично, но с реакцией на противоположную тенденцию изменения X, работает алгоритм в режиме "запрет вниз".

Работа алгоритма поясняется рисунком и таблицей.



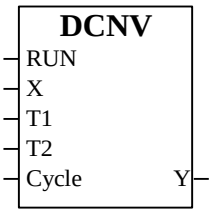
1. Zup=0; Y=X; Dup=0;
2. Zup=1; Y=Y1=const; Dup=1
3. Zup=1; Y=X; Dup=0

Zup	Zdown	X	Y	Dup	Ddown	Режим запрета
0	0	*	Y= X	0	0	-
1	0	X ≥Y	Y=Y1=const	1	0	+
1	0	X <Y	Y= X	0	0	-
0	1	X ≤Y	Y=Y1=const	0	1	+
0	1	X >Y	Y= X	0	0	-
1	1	X =Y	Y=Y1=const	1	1	+
1	1	X >Y	Y=Y1=const	1	0	+
1	1	X <Y1	Y=Y1=const	0	1	+

* - значение сигнала безразлично;

Y1 - значение Y в момент прихода команды запрета.

7.3.17 DCNV



Входы:

RUN	BOOLEAN	Режим работы
X	REAL	Основной вход
T1	REAL	Интегральная постоянная (сек)
T2	REAL	Дифференциальная постоянная (сек)
Cycle	INTEGER	Период работы блока (мсек)

Выход:

Y	REAL	Основной выход

Назначение

Функциональный блок DCNV применяется для динамической коррекции систем управления в тех случаях, когда требуется интегрально - дифференцирующее преобразование сигнала.

Описание алгоритма

При состоянии входа RUN=TRUE входной сигнал X преобразуется в выходной сигнал Y в соответствии с передаточной функцией интегрально-дифференцирующего звена:

$$W(p) = \frac{T2 \cdot p + 1}{T1 \cdot p + 1},$$

где T1 и T2 - постоянные времени.

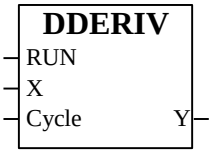
При T1=T2 выход алгоритма равен текущему значению входа. Если T1>T2 преобладает интегральная составляющая звена, если T2>T1 преобладает дифференциальная составляющая звена. Если T1≤0 или T2≤0 выход алгоритма равен нулю.

Параметр Cycle является периодом работы блока и шагом дискретизации при преобразовании входного сигнала.

При состоянии входа RUN=FALSE выход блока Y равен входному сигналу X. При этом период работы блока равен циклу контроллера.

Примечание: при значении Cycle≤0 период работы блока равен циклу контроллера.

7.3.18 *DDERIV*



Входы:

RUN	BOOLEAN	Режим работы
X	REAL	Основной вход
Cycle	INTEGER	Период работы блока (мсек)

Выход:

Y	REAL	Основной выход

Назначение

Функциональный блок DDERIV применяется в схемах динамической коррекции для получения сигналов, связанных со скоростью изменения параметра.

Описание алгоритма

Алгоритм представляет собой идеальное дифференцирующее звено второго порядка.

Передаточная функция алгоритма имеет вид:

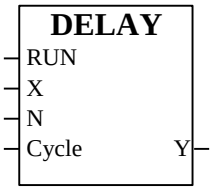
$$W(p) = \frac{Y(p)}{X(p)} = p^2,$$

В ручном режиме (RUN=FALSE) и в момент переключения на автоматический режим (RUN=TRUE) выход блока Y равен нулю.

Параметр Cycle является периодом работы блока и периодом дифференцирования (шагом дискретизации).

Примечание: при значении Cycle≤0 период работы блока равен циклу контроллера.

7.3.19 *DELAY*



Входы:

RUN	BOOLEAN	Режим работы: TRUE -нормальный, FALSE -отключенный
X	REAL	Основной вход
N	INTEGER	Количество циклов запаздывания от 1 до 128
Cycle	INTEGER	Длительность цикла (мсек)

Выход:

Y	REAL	Основной выход

Назначение

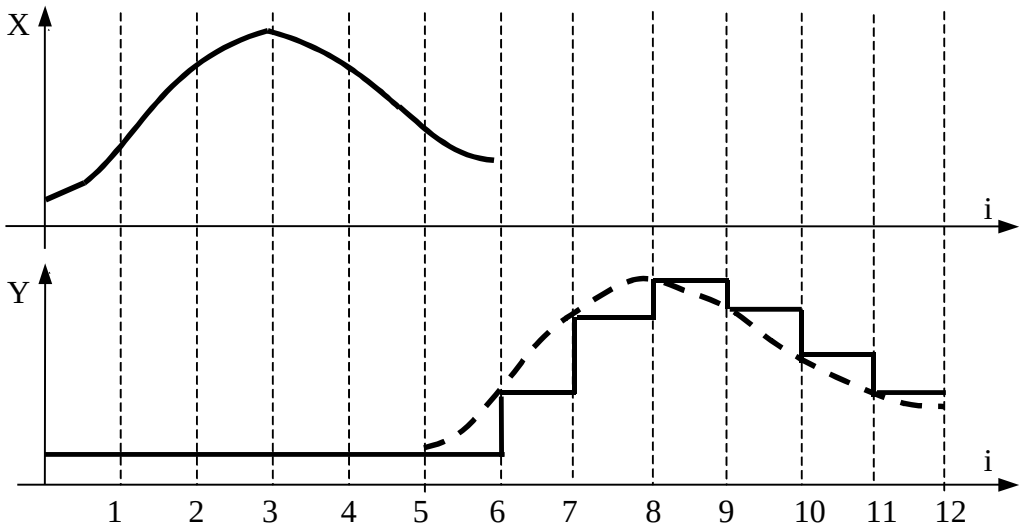
Функциональный блок DELAY предназначен для моделирования звена чистого запаздывания и используется для введения динамической коррекции или для моделирования динамических свойств объекта управления.

Описание алгоритма

При состоянии входа RUN=TRUE текущее значение входного сигнала X заносится в массив и будет прочитано из этого массива через N циклов. Таким образом, текущее значение X появится на выходе Y через N циклов. При этом выходной сигнал Y будет запаздывать относительно сигнала X на время N·Cycle, где Cycle – период работы блока (длительность одного цикла запаздывания). Рисунок поясняет работу алгоритма при N=5.

При состоянии входа RUN=FALSE или при N≤0 выход Y равен текущему значению входа X. При переходе входного сигнала RUN из состояния FALSE в состояние TRUE все N элементов массива инициализируются текущим значением X.

Примечание: Изменение значения N при состоянии входа RUN=TRUE алгоритмом не воспринимается, поэтому значение N можно изменять только при состоянии RUN=FALSE. Если значение параметра Cycle меньше времени цикла контроллера (или Cycle≤0), то период работы блока равен циклу контроллера. Значение N>128 алгоритм воспринимает как N=128.



i – количество циклов, прошедших с момента запуска алгоритма

7.3.20 DENS_CALC

Входы:

DENS1_CONFIG	Integer	Конфигурация 1 плотномера
DENS2_CONFIG	Integer	Конфигурация 2 плотномера
DENS1_MANUAL	Real	Ручное значение 1 плотномера
DENS2_MANUAL	Real	Ручное значение 2 плотномера
dens1_err	Integer	Ошибки 1 плотномера
dens2_err	Integer	Ошибки 2 плотномера
p120_MI_GOST	Boolean	Способ расчета плотн. при 20гр.С (FALSE – по МИ, TRUE – по ГОСТу)
T_01	Real	Период плотномера
DENS1_K0_D0	Real	Коэффициент K0
DENS1_K1_K	Real	Коэффициент K1
DENS1_K2_T0	Real	Коэффициент K2
DENS1_K18_TEMPC0	Real	Коэффициент K18
DENS1_K19_PRESC0	Real	Коэффициент K19
DENS1_K20A_tcal	Real	Коэффициент K20A
DENS1_K20B_pcal	Real	Коэффициент K20B
DENS1_K21A	Real	Коэффициент K21A
DENS1_K21B	Real	Коэффициент K21B
TEMP	Real	Температура в БИКе
PRESS	Real	Давление в БИКе
T_02	Real	Период 2 плотномера
DENS2_K0_D0	Real	Коэффициент K0
DENS2_K1_K	Real	Коэффициент K1
DENS2_K2_T0	Real	Коэффициент K2
DENS2_K18_TEMPC0	Real	Коэффициент K18
DENS2_K19_PRESC0	Real	Коэффициент K19
DENS2_K20A_tcal	Real	Коэффициент K20A
DENS2_K20B_pcal	Real	Коэффициент K20B
DENS2_K21A	Real	Коэффициент K21A
DENS2_K21B	Real	Коэффициент K21B

Выходы:

DENS	Real	Плотность
DENS1_P0	Real	Плотность без учета T и P для 1 плотномера
DENS1_Pt	Real	Плотность с учетом T для 1 плотномера
DENS1_Ptp	Real	Плотность с учетом T и P для 1 плотномера
DENS2_P0	Real	Плотность без T и P для 2 плотномера

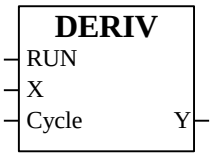
DENS2_Pt	Real	Плотность с учетом T для 2 плотномера
DENS2_Ptp	Real	Плотность с учетом T и P для 2 плотномера
DENS15	Real	Плотность при 15 град.C
DENS20	Real	Плотность при 20 град.C
CTL20	Real	Коэфф. CTL при 20 град.C
CPL20	Real	Коэфф. CPL при 20 град.C
KV15	Real	Коэфф. объемного расширения
KY15	Real	Коэфф. сжимаемости
err	Integer	Код ошибки: 0 – нет ошибки 1 – ошибка обмена с задачей связи

Назначение

Функциональный блок используется для расчета свойств нефти. Вычисляются текущие значения плотности нефти и коэффициентов, учитывающих влияние температуры и давления нефти.

При использовании данного функционального блока на контроллере должна быть запущена задача dens_calc. (правила запуска приводятся в документе “Unimod PRO. Исполнительная система”).

7.3.21 *DERIV*



Входы:

RUN	BOOLEAN	Режим работы
X	REAL	Основной вход
Cycle	INTEGER	Период работы блока (мсек)

Выход:

Y	REAL	Основной выход

Назначение

Функциональный блок DERIV применяется в схемах динамической коррекции для получения сигналов, связанных со скоростью изменения параметра.

Описание алгоритма

Алгоритм представляет собой идеальное дифференцирующее звено первого порядка.

Передаточная функция алгоритма имеет вид:

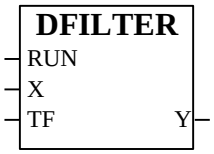
$$W(p) = \frac{Y(p)}{X(p)} = p\,,$$

В ручном режиме (RUN=TRUE) и в момент переключения на автоматический режим выход блока Y равен нулю.

Параметр Cycle является периодом работы блока и периодом дифференцирования (шагом дискретизации).

Примечание: при значении Cycle≤0 период работы блока равен циклу контроллера.

7.3.22 DFILTER



Входы:

RUN	BOOLEAN	Режим работы
X	BOOLEAN	Основной вход
TF	INTEGER	Минимальная длительность импульса (мсек)

Выход:

Y	BOOLEAN	Отфильтрованный выход

Назначение:

Используется для фильтрации входных дискретных сигналов.

Описание алгоритма:

Алгоритм анализирует длительность дискретных значений входного сигнала X. При состоянии входа RUN=FALSE входные импульсы с длительностью дискретных значений (1 или 0) меньше заданной не пропускаются на выход Y. Т.е. выполняются условия:

если t_x<TF, то t_y=TF;

если $t_x \geq$TF, то $t_y = t_x$;

где t_x – длительность импульсов на входе X;

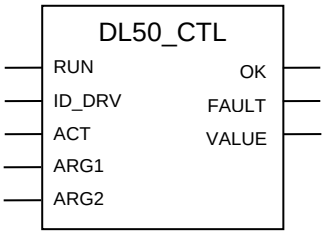
t_y – длительность импульсов на выходе Y;

TF – заданная длительность импульсов.

Длительность импульсов с выхода Y кратна времени цикла контроллера.

При состоянии входа RUN=FALSE входной сигнал X не фильтруется, выход Y равен входу X.

7.3.23 DL50_CTL



Входы:

RUN	BOOLEAN	Запуск функционального блока
ID_DRV	INTEGER	Идентификатор устройства (COM1..COMn)
ACT	INTEGER	Выполняемое действие
ACT1	INTEGER	Аргумент 1 (зависит от аргумента ACT)
ACT2	INTEGER	Аргумент 2 (зависит от аргумента ACT)

Выходы:

READY	BOOLEAN	Признак завершения операции
FAULT	INTEGER	Код завершения операции
VALUE	INTEGER	Возвращаемое значение

Назначение:

Служебные запросы дальномеру DL50Hi.

Описание:

Функциональный блок выполняет чтение/запись уставок дальномера DL50Hi, а также выполнение управляющих функций.

Вход ACT определяет выполняемую блоком функцию и может принимать следующие значения:

1 - Чтение параметров дальномера (параметр задается входом ARG1).
2 - Запись параметров дальномера (значение параметра задается входом ARG2).
3 - Request signal level.
4 - Reset to default settings (требует последующего вызова команды 5 "Activate and save parameter").
5 - Activate and save parameter.

Вход ARG1 используется только при значении ACT, равном 1 или 2, и задает пункт меню.

Выход Value содержит возвращаемое значение.

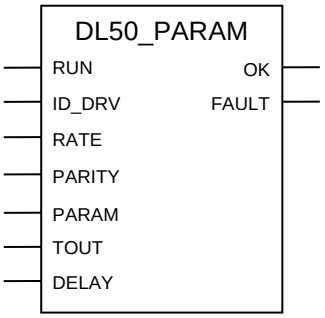
Возможные значения входа ARG1:

1 - Q1 (Диапазон возвращаемого значения 200..50000 мм)
2 - Q2 (Диапазон возвращаемого значения 200..50000 мм)
3 - Q1&Q2 logic.
    Выход Value принимает следующие значения:
    0 - Q1&Q2
    1 - /Q1&Q2
    4 - Q1&/Q2
    5 - /Q1&/Q2
4 - Q1Hyst (Диапазон возвращаемого значения 1..1000 мм)
5 - Q2Hyst (Диапазон возвращаемого значения 1..1000 мм)
6 - Averag.
    Выход Value принимает следующие значения:
    0 - FAST
    1 - MEDIUM
    2 - SLOW
7 - MF.
    Выход Value принимает следующие значения:
    0 - Laser off
    1 - ExternTeach
    2 - Q2
    3 - inactive
8 - Serial.
    Выход Value принимает следующие значения:
    0 - Request
    1 - Continuous
9 - BaudRt.
    Выход Value принимает следующие значения:
    0 - 19200
    1 - 38400
    2 - 57600
    3 - 115200
10 - Parity.
    Выход Value принимает следующие значения:
    0 - None
    1 - Even
    2 - Odd

Вход ARG2 используется только при значении входа ACT, равном 2. Допустимые значения зависят от входа ARG1.

Все значения, кроме установки скорости работы и режима четности, применяются сразу, но действуют до сброса питания. Чтобы сохранить настройки, а также для смены скорости или режима четности, необходимо выполнить команду ACT=5 (Activate and save parameter).

7.3.24 DL50_PARAM



Входы:

RUN	BOOLEAN	Запуск функционального блока
ID_DRV	INTEGER	Идентификатор устройства (COM1..COMn)
RATE	INTEGER	Скорость передачи по последовательной линии
PARITY	INTEGER	Количество стоповых бит и режим контроля четности
PARAM	INTEGER	Режим работы
TOUT	INTEGER	Время ожидания ответа (мс)
DELAY	INTEGER	Длительность паузы между передачами (мс)

Выходы:

OK	BOOLEAN	Признак завершения операции
FAULT	INTEGER	Код завершения

Назначение:

Конфигурация связи с дальномером DL50Hi (на мастер-модуле должна быть запущена задача связи **dl50_rtu_mst**).

Описание:

Изменение конфигурации происходит при единичном значении входа RUN. Новые установки вступают в силу незамедлительно. Текущий приём или передача посылки будет прервана, что может привести к обнаружению ошибок передачи мастер-контроллером.

Вход ID_DRV задаёт идентификатор последовательного устройства, работающего в режиме RS-422. При необходимости вместо одного порта RS-422 может быть задействовано два порта RS-485. Тогда номер принимающего порта задается в младшей части входа ID_DRV, номер передающего - в старшей.
Например:

ID_DRV:=16#010002; // COM1 используется для передачи команд дальномеру, COM2 - для приема ответов.

Вход RATE задает скорость обмена и может принимать одно из следующих значений:

19200, 38400, 57600, 115200 бод.

Количество стоповых битов и режим контроля четности задается входом PARITY. Возможные варианты режима контроля четности представлены в таблице 1.

Таблица 1 – Режим контроля четности

Значение	Количество стоповых бит	Контроль четности
0	1	Отключен
1	2	Отключен
2	1	EVEN
3	1	ODD
4	1	SPACE
5	1	MARK

Для мастеров M1010E, M1011E, M1012E

Формат аргумента PARITY. имеет следующий вид:

MSB X X X X X X X X X X X X X X X X X X X X X X STOP9 STOP8 X X X X X X PARITY1 PARITY0 LSB

PARITY (разряд 0,1), контроль четности задается входом PARITY. Возможные варианты режима контроля четности представлены в таблице 2:

Таблица 2 – Режим контроля чётности

Значение	Контроль четности
0	Отключен
1	EVEN
2	ODD

STOP (разряд 8,9), количество стоповых бит. Возможные варианты представлены в таблице 3:

Таблица 3 – Возможные варианты количества стоповых бит

Значение	Количество стоповых бит
0	1
1	1
2	1.5
3	2

Например:

PARITY :=16#0100; //1-стоповых бит,контроль четности отключен

Формат аргумента PARAM имеет следующий вид:

MSB X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X CONT X SYNC **LSB**

Формат аргумента PARAM

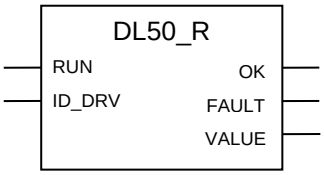
SYNC (разряд 0, synchronous). Если флаг установлен, обмен с дальномером происходит в синхронном режиме: вызванный функциональный блок не возвращает управление технологической программе до тех пор, пока текущая операция не будет завершена. При сброшенном флаге SYNC функциональные блоки работают асинхронном режиме, а момент завершения операции определяется состоянием выходов READY.

CONT (разряд 2, continuous). Если флаг установлен, предполагается использование дальномера как обычный датчик, т.е. когда дальномер работает в режиме не запрос-ответ, а в режиме постоянной выдачи данных в линию. При этом передача управляющих запросов дальномеру в данного порта блокируется. При сброшенном флаге CONT режим работы соответствует режиму дальномера запрос-ответ.

Параметры TOUT и DELAY задают время ожидания ответа от дальномера и задержку после получения

ответа и перед следующей передачей соответственно. При установленном флаге CONT (вход PARAM) параметр TOUT задает максимальный временной интервал между получением данных с дальномера, по истечении которого фиксируется обрыв линии и на выходе FAULT формируется соответствующая ошибка.После завершения операции, выход функционального блока OK устанавливается в состояние TRUE, а на выходе FAULT – код ошибки или нуль, если вызов отработан без ошибок.

7.3.25 DL50_R



Входы:

RUN	BOOLEAN	Запуск функционального блока
ID_DRV	INTEGER	Идентификатор устройства (COM1..COMn)

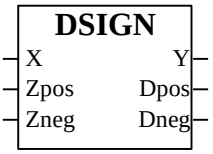
Выходы:

READY	BOOLEAN	Признак завершения операции
FAULT	INTEGER	Код завершения операции
VALUE	INTEGER	Возвращаемое значение (миллиметры)

Назначение:

Чтение данных с дальномера DL50Hi.

7.3.26 *DSIGN*



Входы:

X	REAL	Основной вход
Zpos	BOOLEAN	Команда запрета "плюс"
Zneg	BOOLEAN	Команда запрета "минус"

Выходы:

Y	REAL	Основной выход
Dpos	BOOLEAN	Признак запрета "плюс"
Dneg	BOOLEAN	Признак запрета "минус"

Назначение

Функция DSIGN применяется для запрета перехода сигнала в область положительных или отрицательных значений.

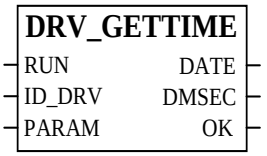
Описание алгоритма

Алгоритм содержит узел запрета, управляемый входными дискретными сигналами Cpos и Cneg. При действии команды Cpos запрещается переход выходного сигнала Y в область положительных значений, команда Cneg запрещает изменение Y в область отрицательных значений. Выход Dplus устанавливается в 1 при режиме запрета ”плюс”, выход Dmin устанавливается в 1 при режиме запрета “минус”.
Работа алгоритма описывается следующей таблицей.

Zpos	Zneg	X	Y	Dpos	Dneg	Режим запрета
0	0	*	Y=X	0	0	-
1	0	X>0	Y=0	1	0	+
1	0	X ≤0	Y=X	0	0	-
0	1	X<0	Y=0	0	1	+
0	1	X≥0	Y=X	0	0	-
1	1	*	Y=0	1	1	+

* - значение сигнала безразлично.

7.3.27 DRV_GETTIME



Входы:

RUN	BOOLEAN	Режим работы: TRUE - выполнить, FALSE - не выполнять
ID_DRV	INTEGER	Идентификатор устройства
PARAM	MESSAGE	Дополнительные параметры

Выходы:

DATE	INTEGER	Дата (день,месяц,год)
DMSEC	INTEGER	Миллисекунда с начала суток
FAULT	INTEGER	Код завершения операции: 0 – выполнено успешно 1 – ошибка выделения памяти (некорректные параметры) 2 – превышено время ожидания ответа 3 – системная ошибка при работе с портом 4 – ошибка инициализации задачи связи 5 – выполняется запрос 6 – ошибка принятых данных 7 – внутренняя ошибка 8 – Локальные часы

Назначение

Получение текущего времени с удаленного устройства.

Описание:

Источник синхронизации задается входом **ID_DRV**. Возможны следующие значения:

0 - GPS-приемник
1 - NTP-сервер
2 - устройство МЭК-104 в режиме master (время, полученное по команде синхронизации 103)

1) При нулевом значении **ID_DRV** функциональный блок аналогичен ***GPS_GETTIME***. Вход PARAM не используется. На контроллере должна быть запущена задача gps.

2) При единичном значении **ID_DRV** выполняется получение времени с ***NTP-сервера*** (на мастер-модуле должна быть запущена задача связи **um_ntp**). Вход **PARAM** задает IP-адрес сервера. Алгоритм действий следующий:

При первом запуске функционального блока выполняется инициализация процедуры синхронизации. При этом, до получения первого ответа от NTP-сервера выход FAULT принимает значение 5 (блок выполняется асинхронно, т.е. корректное значение на выходах DATE и DMSEC будет установлено, когда выход FAULT примет нулевое значение).

После этого получение времени с сервера производится автоматически, а последующие вызовы функционального блока возвращают текущее время и код ошибки последней операции синхронизации или нуль, если запрос отработан без ошибок.

Примечание

Если выход FAULT принимает значение 7, то от сервера пришел ответ-исключение, который сигнализирует о том, что обращения к нему следует прекратить (например, закрытый сервер), или сервер находится в каком-то особенном временном состоянии и не может ответить на запрос. При этом необходимо сменить IP-адрес сервера и перезапустить функциональный блок.

Если выход FAULT принимает значение 8, это означает, что NTP-сервер перешел на локальные часы (поле ClockID равно LOCL). Если в качестве NTP-сервера применяется Сервер единого времени S350,

данный код ошибки сигнализирует о том, что S350 не может синхронизироваться от GPS-приемника и выдает время по своим внутренним часам.

3) При значении входа **ID_DRV**, равному 2, функциональный блок возвращает время, полученное по команде синхронизации часов (идентификатор типа 103). Вход PARAM не используется. На мастер-модуле должна быть запущена задача **iec104_slv**.

7.3.28 DRV_SYNCTIME

Входы:

RUN	BOOLEAN	Режим работы: TRUE - выполнить, FALSE - не выполнять
ID_DRV	INTEGER	Идентификатор устройства
PARAM	#MESSAGE	Дополнительные параметры
TIMEZONE	INTEGER	Часовой пояс
MOD_PERIOD	INTEGER	Период синхронизации модулей ввода-вывода (сек)

Выходы:

FAULT	INTEGER	Код завершения операции: 0 – выполнено успешно 1 – ошибка выделения памяти (некорректные параметры) 2 – превышено время ожидания ответа 3 – системная ошибка при работе с портом 4 – ошибка инициализации задачи связи 5 – выполняется запрос 6 – ошибка принятых данных 7 – внутренняя ошибка 8 – Локальные часы
DATE	INTEGER	Дата (день,месяц,год)
DMSEC	INTEGER	Миллисекунда с начала суток
USEC	INTEGER	Микросекунда с начала миллисекунды
CUR_LINE	INTEGER	Текущая линия
SYNC	BOOLEAN	Признак цикла синхронизации

Назначение

Синхронизация времени с временем удаленного устройства

Описание:

Источник синхронизации задается входом **ID_DRV**. Возможны следующие значения:

0 - GPS-приемник
1 - NTP-сервер
2 - устройство МЭК-104 в режиме master (время, полученное по команде синхронизации 103)

Вход **PARAM** используется в зависимости от значения входа ID_DRV, см. ниже.

Вход **TIMEZONE** задает часовой пояс.

Вход **MOD_PERIOD** используется, если в проекте присутствуют модули с регистрацией событий и задает период (секунды) синхронизации времени модулей с временем мастер-модуля (синхронизация выполняется только при нулевом значении на выходе FAULT).
Если MOD_PERIOD равен нулю, синхронизация времени модулей не выполняется.
Кроме того, время на модулях можно синхронизировать с помощью системного вызова system(1,0).

Выходы **DATE** и **DMSEC** возвращают время во внутреннем формате UnimodPRO. Для расшифровки может использоваться ФБ DATETIME.

Выход **CUR_LINE** имеет значение в режиме ID_DRV=1, если на вход PARAM передается массив строковых переменных (несколько NTP-серверов).

Выход **SYNC** имеет значение в режиме ID_DRV=1 и выставляется в TRUE на один цикл, в момент успешной синхронизации с NTP-сервером.

Подробнее о режимах работы функционального блока:

1) ID_DRV=0.

При нулевом значении **ID_DRV** функциональный блок аналогичен ***GPS_GETTIME***. Вход PARAM не используется. На контроллере должна быть запущена задача gps.

2) ID_DRV=1.

При единичном значении **ID_DRV** выполняется получение времени с ***NTP-сервера*** (на мастер-модуле должна быть запущена задача связи **um_ntp**). Вход **PARAM** задает IP-адрес сервера. Алгоритм действий следующий:

При первом запуске функционального блока выполняется инициализация процедуры синхронизации. При этом, до получения первого ответа от NTP-сервера выход FAULT принимает значение 5 (блок выполняется асинхронно, т.е. корректное значение на выходах DATE и DMSEC будет установлено, когда выход FAULT примет нулевое значение).

После этого получение времени с сервера производится автоматически, а последующие вызовы функционального блока возвращают текущее время и код ошибки последней операции синхронизации или нуль, если запрос отработан без ошибок.

Примечание

При необходимости задать несколько NTP-серверов на вход PARAM может передаваться массив из переменных строкового типа (каждый элемент задает IP-адрес сервера). В этом случае ФБ пытается синхронизироваться с первым сервером. В случае ошибки переходит на второго, и т.д.

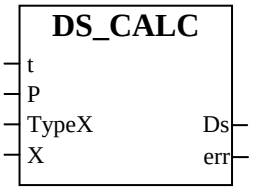
Если выход FAULT принимает значение 7, то от сервера пришел ответ-исключение, который сигнализирует о том, что обращения к нему следует прекратить (например, закрытый сервер), или сервер находится в каком-то особенном временном состоянии и не может ответить на запрос. При этом необходимо сменить IP-адрес сервера и перезапустить функциональный блок.

Если выход FAULT принимает значение 8, это означает, что NTP-сервер перешел на локальные часы (поле ClockID равно LOCL). Если в качестве NTP-сервера применяется Сервер единого времени S350, данный код ошибки сигнализирует о том, что S350 не может синхронизироваться от GPS-приемника и выдает время по своим внутренним часам.

3) ID_DRV=2.

При значении входа **ID_DRV**, равному 2, функциональный блок возвращает время, полученное по команде синхронизации часов (идентификатор типа 103). Вход PARAM не используется. На мастер-модуле должна быть запущена задача **iec104_slv**.

7.3.29 DS_CALC



Входы:

t	REAL	Температура газа (град.С)
P	REAL	Давление газа (кПа)
TypeX	BOOLEAN	Тип компонентного состава газа: FALSE – в молярных долях; TRUE – в объемных долях.
X	#REAL	Ссылка на массив с компонентным составом газа (элементы типа Real)

Выходы:

Ds	REAL	Плотность газа при стандартных условиях (кг/куб.м)
err	INTEGER	Код ошибки 3 – недопустимый размер массива; 4 – недопустимая ссылка на массив.

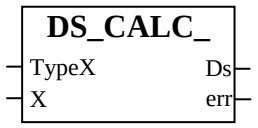
Назначение

Функциональный блок используется для расчета плотности природного газа при стандартных условиях по компонентному составу (ГОСТ 30319.1-96, ГОСТ 8.563.2-97).

Вход X содержит концентрации компонентов газа в следующем порядке:

X[1]	Метан
X[2]	Этан
X[3]	Пропан
X[4]	н-Бутан
X[5]	и-Бутан
X[6]	Азот
X[7]	Диоксид углерода
X[8]	Сероводород
X[9]	Ацетилен
X[10]	Этилен
X[11]	Пропилен
X[12]	н-Пентан
X[13]	и-Пентан
X[14]	н-Гексан
X[15]	Бензол
X[16]	н-Гептан
X[17]	Толуол
X[18]	н-Октан
X[19]	Гелий
X[20]	Водород
X[21]	Монооксид углерода
X[22]	Кислород

7.3.30 DS_CALC_



Входы:

TypeX	BOOLEAN	Тип компонентного состава газа: FALSE – в молярных долях; TRUE – в объемных долях.
X	#REAL	Ссылка на массив с компонентным составом газа (элементы типа Real)

Выходы:

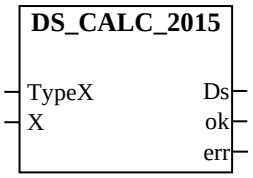
Ds	REAL	Плотность газа при стандартных условиях (кг/куб.м)
err	INTEGER	Код ошибки 3 – недопустимый размер массива; 4 – недопустимая ссылка на массив.

Назначение
Функциональный блок используется для расчета плотности природного газа при стандартных условиях по компонентному составу (ГОСТ 30319.1-96, ГОСТ 8.563.2-97).

Вход X содержит концентрации компонентов газа в следующем порядке:

X[1]	Метан
X[2]	Этан
X[3]	Пропан
X[4]	н-Бутан
X[5]	и-Бутан
X[6]	н-Пентан
X[7]	и-Пентан
X[8]	н-Гексан
X[9]	н-Гептан
X[10]	н-Октан
X[11]	Ацетилен
X[12]	Этилен
X[13]	Пропилен
X[14]	Бензол
X[15]	Толуол
X[16]	Водород
X[17]	Водяной пар
X[18]	Аммиак
X[19]	Метанол
X[20]	Сероводород
X[21]	Метилмеркаптан
X[22]	Диоксид серы
X[23]	Гелий
X[24]	Неон
X[25]	Аргон
X[26]	Монооксид углерода
X[27]	Азот
X[28]	Воздух
X[29]	Кислород
X[30]	Диоксид углерода

7.3.31 DS_CALC_2015



Входы:

TypeX	BOOLEAN	Тип компонентного состава газа: FALSE – в молярных долях; TRUE – в объемных долях.
X	#REAL	Ссылка на массив с компонентным составом газа (элементы типа Real)

Выходы:

Ds	REAL	Плотность газа при стандартных условиях (кг/куб.м)
ok	BOOLEAN	Код завершения FALSE – нет ошибки TRUE – есть ошибки
err	INTEGER	Код ошибки 0 – нет ошибки; 1 – ошибка обмена с задачей связи; 2 – недопустимый тип массива;

3 - недопустимый размер массива;
4 - недопустимая ссылка на массив;
5 - недопустимая сумма компонентов газовой смеси.

При использовании данного функционального блока на контроллере должна быть запущена задача **ds_calc_2015**.

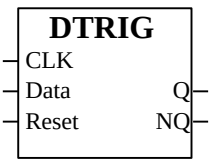
Назначение

Функциональный блок используется для определения плотности смеси газов при стандартных условиях по компонентному составу (ГОСТ 30319.0-96 - ГОСТ 30319.1-96, Изменение №1 к ГОСТ 30319.1-96 от 01.06.2004, ГОСТ 30319.2-2015) при стандартном давлении 0,101325 МПа и стандартной температуре 293,15 K.

Вход X содержит концентрации компонентов газа в следующем порядке:

X[1]	Метан
X[2]	Этан
X[3]	Пропан
X[4]	н-Бутан
X[5]	и-Бутан
X[6]	н-Пентан
X[7]	и-Пентан
X[8]	н-Гексан
X[9]	н-Гептан
X[10]	н-Октан
X[11]	Ацетилен
X[12]	Этилен
X[13]	Пропилен
X[14]	Бензол
X[15]	Толуол
X[16]	Водород
X[17]	Водяной пар
X[18]	Аммиак
X[19]	Метанол
X[20]	Сероводород
X[21]	Метилмеркаптан
X[22]	Диоксид серы
X[23]	Гелий
X[24]	Неон
X[25]	Аргон
X[26]	Монооксид углерода
X[27]	Азот
X[28]	Воздух
X[29]	Кислород
X[30]	Диоксид углерода

7.3.32 *DTRIG*



Входы:

CLK	BOOLEAN	Команда записи
Data	BOOLEAN	Входной сигнал

Reset	BOOLEAN	Команда сброса

Выходы:

Q	BOOLEAN	Основной выход
NQ	BOOLEAN	Инверсный выход

Назначение

Функциональный блок DTRIG используется для запоминания дискретного сигнала. Информация записывается по переднему фронту входного дискретного управляющего сигнала.

Описание алгоритма

По переднему фронту дискретного сигнала CLK (т.е. в момент перехода сигнала CLK из состояния логического 0 в состояние логической 1) информация на входе Data записывается, после чего алгоритм не реагирует на изменение сигнала на входе Data. Записанная информация передается на выход Q. По команде сброса (Reset=TRUE) триггер сбрасывается, выход Q=FALSE.

Инверсный выход NQ равен инвертированному значению основного выхода Q.

7.3.33 EPS_CALC

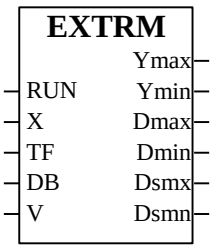
Назначение: Расчет коэффициента расширения газов (ГОСТ 8.586.3-2005, формула.5,2, перечисление к)

Вход	ResType	Integer	Тип сужающего устройства
	ka	Real	Показатель адиабаты
	dP	Real	Перепад давления (кПа)
	P	Real	Давление абсолютное (кПа)
	b	Real	Диаметр относительный
Выход	Eps	Real	Коэффициент расширения
	err	Integer	Код ошибки

Код ошибки:

0 – нет ошибки
1 – недопустимое значение давления
2 – недопустимое значение адиабаты
4 – недопустимый float-формат значения адиабаты
8 – недопустимый float-формат значения перепада давления
16 – недопустимый float-формат значения абсолютного давления
32 – недопустимый float-формат значения диаметра

7.3.34 *EXTRM*



Входы:

RUN	BOOLEAN	Режим работы: FALSE -сброс, TRUE -запуск
X	REAL	Входной сигнал
TF	REAL	Постоянная времени фильтра (сек)
DB	REAL	Зона нечувствительности
V	REAL	Допустимая скорость изменения входного сигнала (1/сек)

Выходы:

Ymax	REAL	Максимальное значение входного сигнала
Ymin	REAL	Минимальное значение входного сигнала
Dmax	BOOLEAN	Фиксация максимума
Dmin	BOOLEAN	Фиксация минимума
Dsmx	BOOLEAN	Признак поиска максимума
Dsmn	BOOLEAN	Признак поиска минимума

Назначение

Функциональный блок EXTRM применяется для поиска и фиксации максимального и/или минимального значения меняющегося во времени сигнала. В частности, алгоритм используется в задачах оптимизации.

Описание алгоритма

Входной сигнал X проходит через фильтр нижних частот. Фильтр нижних частот имеет передаточную функцию:

$$W(p) = \frac{1}{TF \cdot p + 1}$$, где TF - постоянная времени фильтра.

Если TF≤0, то входной сигнал X не фильтруется.

На выходах блока Ymax, Ymin фиксируется последнее соответственно максимальное и минимальное значение отфильтрованного сигнала X_f.

Алгоритм работает следующим образом (Рисунок 1). До тех пор, пока блок находится в состоянии сброса (RUN=FALSE) поиск экстремума не ведется и выходные сигналы Ymax =Ymin =X. При запуске блока (RUN=TRUE) выходные сигналы Ymax и Ymin замораживаются, и начинается поиск экстремума.

Как только найден максимум, выходной сигнал Ymax становится равным максимальному значению X_f. Когда алгоритм фиксирует минимум, выходной сигнал Ymin принимает значение, равное минимальному значению X_f. Выходы Ymax и Ymin остаются неизменными до обнаружения очередного экстремума – соответственно максимума или минимума.

В момент обнаружения максимума или минимума на время Tc, равное времени цикла работы контроллера, на дискретных выходах соответственно Dmax или Dmin формируются дискретные сигналы Dmax =TRUE или Dmin =TRUE.

На дискретных выходах Dsmx (поиск максимума) и Dsmn (поиск минимума) формируются сигналы, свидетельствующие о направлении поиска. В состоянии сброса поиск не ведется и Dsmx =Dsmn =FALSE. Если в текущем цикле ведется поиск максимума, то Dsmx = TRUE, Dsmn= FALSE, если ведется поиск минимума, то Dsmx = FALSE и Dsmn= TRUE.

В алгоритме предусмотрены меры по повышению помехозащищенности процесса поиска. Как отмечалось выше, входной сигнал проходит через фильтр нижних частот. Также имеется возможность введения зоны нечувствительности и предусмотрена возможность анализа на допустимую скорость изменения сигнала в районе экстремума. Эти меры могут применяться по отдельности, а также в любых сочетаниях.

Влияние зоны нечувствительности и допустимой скорости показано на Рисунок 2.

Если на входе DB установлено значение зоны нечувствительности DB≤0, то экстремум фиксируется сразу же, как только знак производной сменился на противоположный (именно такая ситуация представлена на рисунке 4.1). Если введена зона нечувствительности (DB >0), то экстремум фиксируется не сразу, а лишь после того, как, пройдя экстремум, сигнал изменится на величину, большую DB. Если же сигнал изменится на меньшую величину и затем вновь начнет изменяться в исходном направлении, экстремум не фиксируется (Рисунок 2).

На входе V устанавливается допустимая скорость изменения сигнала, при которой фиксируется экстремум. Если на входе V установлено значение V≤0, то ограничения на скорость не накладываются. В противном случае экстремум фиксируется лишь тогда, когда после прохождения экстремума (или, при наличии зоны нечувствительности – после прохождения этой зоны) скорость изменения сигнала Vx≤V (Рисунок 2).

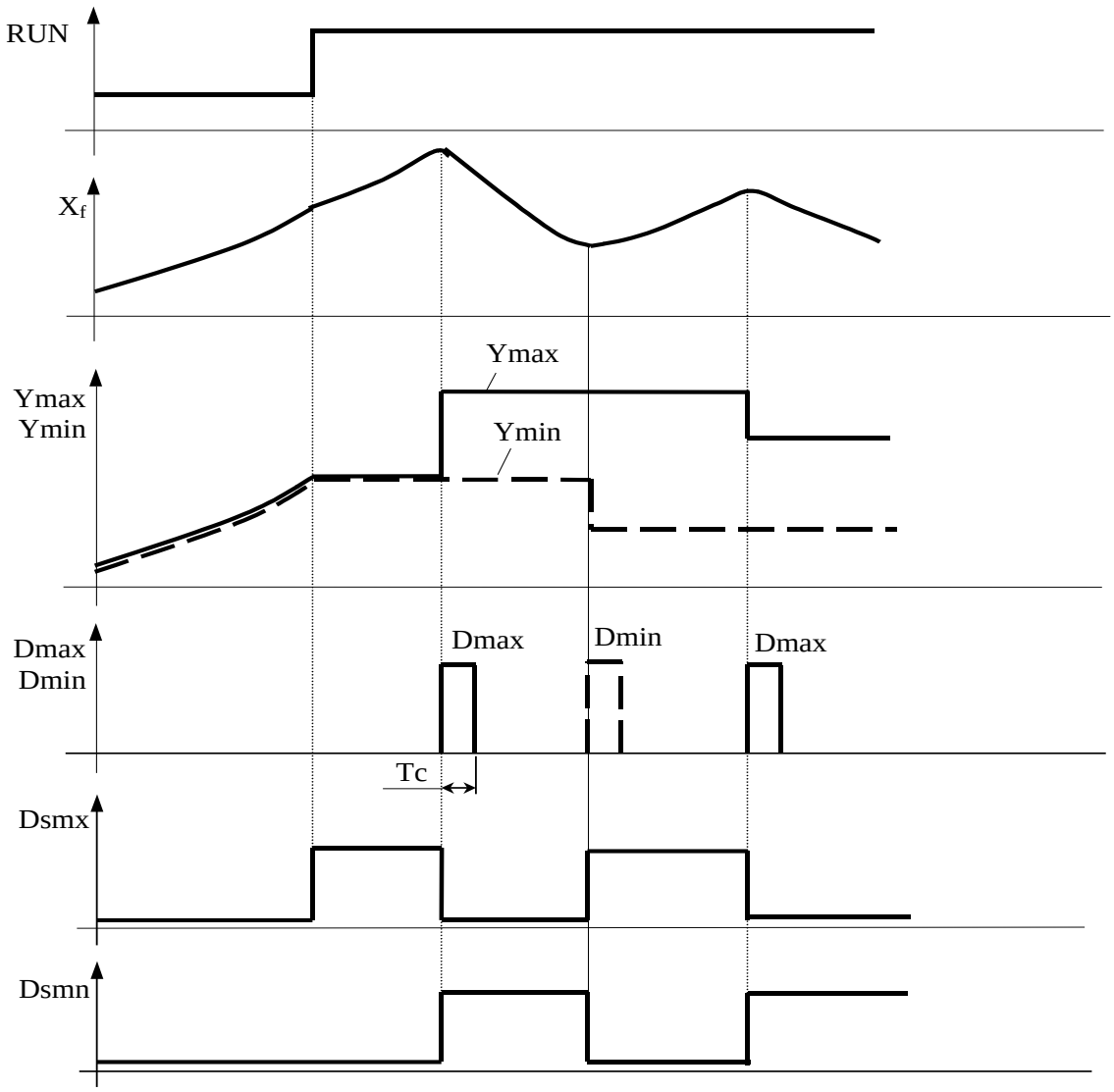
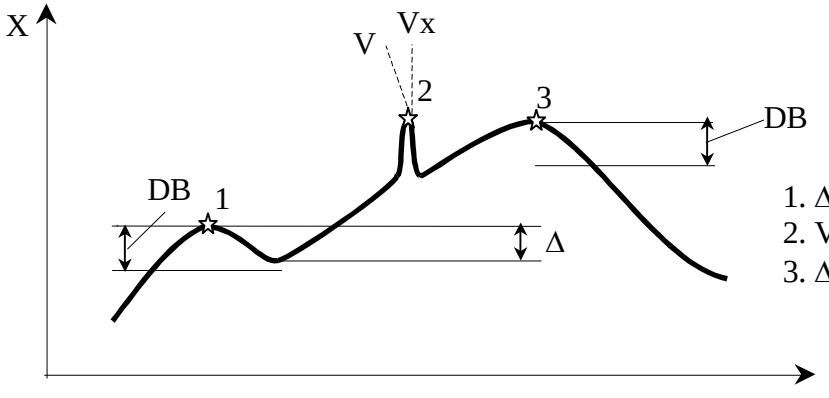


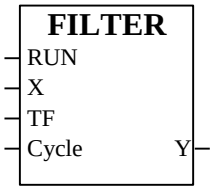
Рисунок 1. Диаграмма работы функционального блока EXTRM.



1. Δ<DB; экстремум не фиксируется;
2. Vx>V; экстремум не фиксируется;
3. Δ>DB; Vx<V; экстремум фиксируется.

Рисунок 2. Влияние зоны нечувствительности и допустимой скорости.

7.3.35 *FILTER*



Входы:

RUN	BOOLEAN	Режим работы
X	REAL	Основной вход
TF	REAL	Постоянная времени фильтра (сек)
Cycle	INTEGER	Период работы блока (мсек)

Выход:

Y	REAL	Отфильтрованный выход

Назначение

Функциональный блок FILTER используется для фильтрации высокочастотных помех, а также для динамической коррекции. Фильтр, имеющий порядок выше первого, можно получить путем последовательного включения нескольких блоков FILTER.

Описание алгоритма

Алгоритм является фильтром нижних частот первого порядка. При состоянии входа RUN=TRUE выходной сигнал Y равен отфильтрованному значению входного сигнала X. Передаточная функция фильтра:

$$W(p) = \frac{Y(p)}{X(p)} = \frac{1}{TF \cdot p + 1},$$

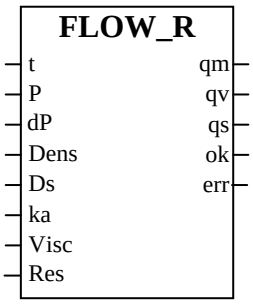
где TF - постоянная времени фильтра.

Параметр Cycle является периодом работы блока.

При состоянии входа RUN=FALSE выход блока Y равен входу X, т.е. входной сигнал не фильтруется.

Примечание: при значении Cycle ≤0, период работы блока равен циклу контроллера.

7.3.36 *FLOW_R*



Входы:

t	REAL	Температура (град.С)
P	REAL	Давление (кПа)
dP	REAL	Перепад давления (кПа)
Dens	REAL	Плотность газа при рабочих условиях (кг / куб.м)
Ds	REAL	Плотность газа при стандартных условиях (кг / куб.м)
ka	REAL	Показатель адиабаты
Visc	REAL	Динамическая вязкость (мкПа * с)
Res	#REAL	Ссылка на массив параметров сужающих устройств (элементы типа Real)

Выходы:

qm	REAL	Массовый расход (кг / с)
qv	REAL	Объемный расход (куб.м / с)
qs	REAL	Объемный расход, приведенный к станд. условиям (куб.м / с)
ok	INTEGER	Код завершения
err	INTEGER	Код ошибки: 0 – нет ошибки; 1 – недопустимое значение температуры; 2 – недопустимое значение давления; 4 – недопустимое значение перепада давления на СУ; 8 – ошибка в параметрах СУ; 16 – недопустимое значение плотности; 32 – недопустимое значение плотности при нормальных усл-ях; 64 – недопустимое значение показателя адиабаты; 128 – недопустимое значение вязкости.

Назначение

Функциональный блок используется для расчета расхода природного газа (ГОСТ 8.563.1-97, ГОСТ 8.563.2-97).

Формат массива параметров сужающих устройств (СУ), 9 переменных вещественного типа:

№	Назначение
0	код типа трубы (не используется, равен 0)
1	код типа СУ (см. Таблица 1)
2	диаметр трубы при 20 град.С (мм)
3	код типа стали трубы (см. Таблица 2)
4	шероховатость трубы (мм)
5	диаметр отверстия СУ при 20 град.С (мм)
6	код типа стали СУ (см. Таблица 2)
7	начальный радиус кромки (для диафрагм) (мм)
8	межповерочный интервал (для диафрагм) (лет)

Таблица 1. Допустимые коды типа СУ:

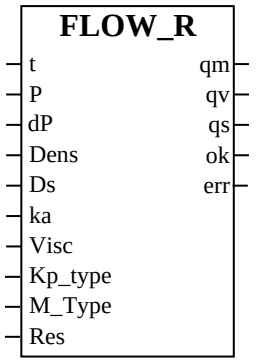
Код	Тип СУ
1	Диафрагма с трехрадиусным отбором давления
2	Диафрагма с фланцевым отбором давления
3	Диафрагма с угловым отбором давления

4	Сопло ИСА 1932
5	Труба Вентури литая
6	Труба Вентури обработанная
7	Труба Вентури сварная
8	Сопло Вентури

Таблица 2. Допустимые коды типа стали:

Код	Тип стали
0	сталь 8
1	сталь 10
2	сталь 15
3	сталь 15М
4	сталь 16М
5	сталь 20
6	сталь 20М
7	сталь 25
8	сталь 30
9	сталь 35
10	Х6СМ
11	Х7СМ
12	12МХ
13	12Х1МФ
14	12Х17
15	12Х18Н9Т
16	12Х18Н10Т
17	14Х17Н2
18	15ХМА
19	15Х1М1Ф
20	15Х5М
21	15Х12ЕМФ
22	17Х18Н9
23	20Х23Н13
24	36Х18Н25С2

7.3.37 *FLOW_R_2005*



Входы:

t	REAL	Температура (град.С)
P	REAL	Давление (кПа) – сумма избыточного давления среды и атмосферного давления
dP	REAL	Перепад давления (кПа)
Dens	REAL	Плотность газа при рабочих условиях (кг / куб.м)
Ds	REAL	Плотность газа при стандартных условиях (кг / куб.м) (используется только если вход M_Type равен 4, в остальных случаях - 0)
ka	REAL	Показатель адиабаты

Visc	REAL	Динамическая вязкость (мкПа * с)
Kp_type	BOOLEAN	Выбор времени эксплуатации или межповерочный интервал (для диафрагм) FALSE - межповерочный интервал (год) TRUE - текущее время эксплуатации с момента определения начального диаметра (год)
M_Type	INTEGER	Тип измеряемой среды: 1 - вода 2 - насыщенный пар 3 - перегретый пар 4 - природный газ
Res	#REAL	Ссылка на массив параметров сужающих устройств (элементы типа Real)

Выходы:

qm	REAL	Массовый расход (кг / с)
qv	REAL	Объемный расход (куб.м / с)
qs	REAL	Объемный расход, приведенный к станд. условиям (куб.м / с)
ok	INTEGER	Код завершения
err	INTEGER	Код ошибки:

При использовании данного функционального блока на контроллере должна быть запущена задача **flow_r_2005**.

Назначение

Функциональный блок используется для измерения расхода и количества жидкостей и газов с помощью стандартных сужающих устройств (ГОСТ 8.586.1-2005 - ГОСТ 8.586.5-2005).

Формат массива параметров сужающих устройств (СУ) (10 переменных вещественного типа):

№	Назначение
0	не используется
1	код типа СУ (целое число от 1 до 9, см. таблица 1)
2	диаметр трубы при 20 град.С (м)
3	код типа стали трубы (целое число от 1 до 62, см. таблица 2 "Марка стали ИТ и СУ")
4	шероховатость трубы (м)
5	диаметр отверстия СУ при 20 град.С (м)
6	код типа стали СУ (целое число от 1 до 62, см. таблица 2 "Марка стали ИТ и СУ")
7	начальный радиус кромки (для диафрагм) (м)
8	межповерочный интервал (для диафрагм) (лет)
9	текущее время эксплуатации с момента определения начального диаметра (для диафрагм) (лет)

Таблица 1. Допустимые коды типа СУ:

Код	Тип СУ
1	***Диафрагма с трехрадиусным отбором давления*** (d>=0,0125 м, 0,050 м<=D<=1,0 м, 0,10 <= b <= 0,75, Re > 5000 при b <= 0,56, Re > 16000*b^2 при b > 0,56);
2	***Диафрагма с фланцевым отбором давления*** (d>=0,0125 м, 0,050 м<=D<=1,0 м, 0,10 <= b <= 0,75, Re > 5000 и Re > 1,7*10^5* b^2*D);
3	***Диафрагма с угловым отбором давления*** (d>=0,0125 м, 0,050 м<=D<=1,0 м, 0,10 <= b <= 0,75, Re > 5000 при b <= 0,56, Re > 16000*b^2 при b > 0,56);
4	***Сопло ИСА 1932*** (0,05 м <= D <= 0,50 м; 0,30 <= b <= 0,8; 7*10^4<=Re <= 10^7 при 0,30 <= b <= 0,44 и 2*10^4<=Re <= 10^7 при 0,44 <= b <= 0,80);
5	***Труба Вентури литая*** (0,10 м <= D <= 0,80 м; 0,30 <= b <= 0,75; Re >= 4*10^4);
6	***Труба Вентури обработанная*** (0,05 м <= D <=

	0,25 м; 0,40 <= b <= 0,75; 4*10^4*b <= Re <= 10^8*b);
7	***Труба Вентури сварная*** (0,20 м <= D <= 1,20 м; 0,40 < b < 0,70; Re >= 4*10^4);
8	***Сопло Вентури*** (0,065 м <= D <= 0,5 м; d=> 0,05м, 0,316 <= b <= 0,775; 1,5*10^5<=Re <= 2*10^6);
9	***Эллипсное Сопла*** (0,05 м <= D <= 0,63 м; 0,2 <= b <= 0,8; 10^4<=Re <= 10^7, Ra/D<=3,2*10^-4).

Таблица 2. Марка стали ИТ и СУ (код типа задается входами Res[3] и Res[6]):

Код	Тип стали
1	сталь 8
2	сталь 10
3	сталь 15
4	сталь 15М
5	сталь 16М
6	сталь 20
7	сталь 20М
8	сталь 25
9	сталь 30
10	сталь 35
11	сталь 40
12	сталь 45
13	Х6СМ
14	Х7СМ
15	12МХ
16	12Х1МФ
17	12Х17
18	12Х18Н9Т
19	12Х18Н10Т
20	14Х17Н2
21	15ХМА
22	15Х1М1Ф
23	15Х5М
24	15Х12ЕМФ
25	17Х18Н9
26	20Х23Н13
27	36Х18Н25С2
28	35Л
29	45Л
30	20ХМЛ
31	12Х18Н9ТЛ
32	15К
33	20К
34	22К
35	16ГС
36	09Г2С
37	10Г2
38	38ХА
39	40Х
40	15ХМ
41	30ХМ
42	30ХМА
43	25Х1МФ
44	25Х2М1Ф
45	18Х2Н4МА
46	38ХН3МФА
47	08Х13
48	12Х13
49	20Х13
50	30Х13

51	10Х14Г14Н4Т
52	08Х18Н10
53	12Х18Н12Т
54	08Х18Н10Т
55	08Х22Н6Т
56	37Х12Н8Г8МФБ
57	31Х19Н9МВБТ
58	06ХН28МДТ
59	20Л
60	25Л
61	20ХМЛ
62	12Х18Н9ТЛ

Код завершения:

FALSE – нет ошибки
TRUE – есть ошибки

Код ошибки:

0 – нет ошибки
1 – ошибка обмена с задачей связи
2 – недопустимый тип массива
3 – недопустимый размер массива (размер массива [0..9]);
4 – недопустимая ссылка на массив
5 – недопустимое значение давления (давление: от 0 кПа);
6 – недопустимое значение температуры (температура: от -200 до +1000 град. С);
7 – неизвестный тип СУ (допустимые коды типа СУ от 1 до 9);
8 – неизвестная марка стали СУ или ИТ (допустимые коды типа СУ или ИТ от 1 до 62);
9 – неправильно задан диаметр СУ или ИТ (диаметр от 0 мм);
10 – недопустимое значение вязкости (вязкость от 0 мкПа * с);
11 – недопустимое значение плотности при рабочих условиях (плотность от 0 кг / куб.м);
12 – недопустимое значение плотности при стандартных условиях (плотность от 0 кг / куб.м);
13 – недопустимое значение показателя адиабаты (показатель адиабаты > 0);
14 – недопустимое значение перепада давления на СУ (0 < 4*dP < P (кПа))
15 – недопустимое значение на входе M_Type (0 < M_Type < 5)
16 – недостоверность входных переменных, проверьте размерость величин
17 – недопустимое значение межповерочного интервала или времени эксплуатации для диафрагм
101 – СУ вышло за границы применения для диафрагма с трехрадиусным отбором давления (диаметр СУ d < 0,0125, диаметр ИТ D < 0,050 м или D > 1,0 м, относительный диаметр СУ b < 0,10 или b > 0,75);
102 – СУ вышло за границы применения для диафрагма с трехрадиусным отбором давления (число Рейнольдса Re < 5000 при относительном диаметре СУ b >0,56);
103 – СУ вышло за границы применения для диафрагма с трехрадиусным отбором давления (число Рейнольдса Re < 16000b^2 при относительном диаметре СУ b <= 0,56);
111 – СУ вышло за границы применения для диафрагма с фланцевым отбором давления (диаметр СУ d < 0,0125, диаметр ИТ D < 0,050 м или D > 1,0 м, относительный диаметр СУ b < 0,10 или b > 0,75);
112 – СУ вышло за границы применения для диафрагма с фланцевым отбором давления (число Рейнольдса Re <= 5000);
113 – СУ вышло за границы применения для диафрагма с фланцевым отбором давления (число Рейнольдса Re <= 1,7*10^5* b^2*D);
121 – СУ вышло за границы применения для диафрагма с угловым отбором давления (диаметр СУ d < 0,0125, диаметр ИТ D < 0,050 м или D > 1,0 м, относительный диаметр СУ b < 0,10 или b > 0,75);
122 – СУ вышло за границы применения для диафрагма с угловым отбором давления (число Рейнольдса Re < 5000 при относительном диаметре СУ b >0,56);
123 – СУ вышло за границы применения для диафрагма с угловым отбором давления (число Рейнольдса Re < 16000b^2 при относительном диаметре СУ b <= 0,56);
131 – СУ вышло за границы применения для Сопла ИСА 1932 (диаметр ИТ D < 0,050 м или D > 0,50 м, относительный диаметр СУ b < 0,30 или b > 0,80);
132 – СУ вышло за границы применения для Сопла ИСА 1932 (число Рейнольдса Re < 7*10^4 или Re > 10^7 при относительном диаметре СУ b<0,3 и b=>0,44);
133 – СУ вышло за границы применения для Сопла ИСА 1932

(число Рейнольдса Re < 2*10^4 или Re > 10^7 при относительном диаметре СУ b<0,44 и b=>0,80);
141 – СУ вышло за границы применения для Трубы Вентури литой
(диаметр ИТ D < 0,01 м или D > 0,8 м, относительный диаметр СУ b < 0,30 или b > 0,75);
142 – СУ вышло за границы применения для Трубы Вентури литой
(число Рейнольдса Re < 4*10^4);
143 – СУ вышло за границы применения для Трубы Вентури литой
(не проходит по условиям шероховатости Ra/D>3,2*10^-4);
151 – СУ вышло за границы применения для Трубы Вентури обработанной
(диаметр ИТ D < 0,05 м или D > 0,25 м, относительный диаметр СУ b < 0,40 или b > 0,75);
152 – СУ вышло за границы применения для Трубы Вентури обработанной
(число Рейнольдса Re < 4*10^4*b или Re > 10^8*b);
153 – СУ вышло за границы применения для Трубы Вентури обработанной
(не проходит по условиям шероховатости Ra/D>3,2*10^-4);
161 – СУ вышло за границы применения для Трубы Вентури сварной
(диаметр ИТ D < 0,2 м или D > 1,2 м, относительный диаметр СУ b < 0,40 или b > 0,70);
162 – СУ вышло за границы применения для Трубы Вентури сварной
(число Рейнольдса Re < 4*10^4);
163 – СУ вышло за границы применения для Трубы Вентури сварной
(не проходит по условиям шероховатости Ra/D>3,2*10^-4);
171 – СУ вышло за границы применения для Сопла Вентури
(диаметр СУ d < 0,05, диаметр ИТ D < 0,065 м или D > 0,5 м, относительный диаметр СУ b < 0,316 или b > 0,775);
172 – СУ вышло за границы применения для Сопла Вентури
(число Рейнольдса Re < 1,5*10^5 и Re > 2*10^6);
181 – СУ вышло за границы применения для Эллипсного Сопла
(диаметр ИТ D < 0,050 м или D > 0,63 м, относительный диаметр СУ b < 0,20 или b > 0,80);
182 – СУ вышло за границы применения для Эллипсного Сопла
(число Рейнольдса Re < 10^4 или число Рейнольдса Re > 10^7);
183 – СУ вышло за границы применения для Эллипсного Сопла
(Ra/D>3.2*10^-4).

7.3.38 FLW_HOLD

Входы:

FLOW	BOOLEAN	Команда слежения
X	REAL	Основной вход

Выходы:

DF	BOOLEAN	Признак слежения
Y	REAL	Основной выход

Назначение

Функциональный блок FLW_HOLD используется для слежения и запоминания аналогового сигнала.

Описание алгоритма

На вход блока X подается аналоговый сигнал, который нужно запомнить. На вход FLOW подается команда слежения.

До тех пор, пока FLOW=TRUE, входной сигнал X передается на выход Y, т.е. выход "следит" за входом. При FLOW=FALSE слежение блокируется, текущее значение Y запоминается и перестает зависеть от входного сигнала X.

Выходной сигнал DF=FLOW, т.е. повторяет входной управляющий сигнал.

Работа алгоритма описывается следующей таблицей.

FLOW	Y	DF
TRUE	Y=X	TRUE

FALSE	Y=X0	FALSE

X0 - значение входного сигнала X в момент снятия команды FLOW.

7.3.39 FS_CLOSE

Входы:

RUN	BOOLEAN	TRUE - выполнить, FALSE не выполнять
DEVICE	INTEGER	Тип накопителя (0 - USB)
TIMEOUT	INTEGER	Таймаут на завершение операции (Сек)

Выходы:

READY	BOOLEAN	Операция завершена – TRUE
FAULT	INTEGER	Код завершения операции: 0 – нет ошибки 1 – устройство недоступно

Назначение

Отключение накопителя (USB).

7.3.40 FS_OPEN

Входы:

RUN	BOOLEAN	TRUE - выполнить, FALSE не выполнять
DEVICE	INTEGER	Тип накопителя (0 - USB)
TIMEOUT	INTEGER	Таймаут на завершение операции (Сек)

Выходы:

READY	BOOLEAN	Операция завершена – TRUE
FAULT	INTEGER	Код завершения операции: 0 – нет ошибки 1 – устройство недоступно

Назначение

Подключение накопителя (USB).

7.3.41 F_TRIG

Вход:

CLK	BOOLEAN	Основной вход

Выход:

Q	BOOLEAN	Основной выход

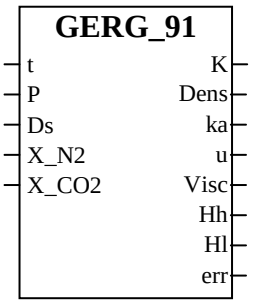
Назначение

Функциональный блок F_TRIG применяется для выделения заднего фронта дискретного сигнала.

Описание алгоритма

Если на входе блока дискретный сигнал CLK изменяет свое состояние с логической 1 на логический 0 (задний фронт), то на выходе алгоритма формируется сигнал Q=1 на время одного цикла работы контроллера. Остальное время Q=0.

7.3.42 GERG_91



Входы:

t	REAL	Температура газа (град.С)
P	REAL	Давление газа (кПа)
Ds	REAL	Плотность газа при стандартных условиях (кг/куб.м)
X_N2	REAL	Молярная плотность азота (%)
X_CO2	REAL	Молярная плотность углекислого газа (%)

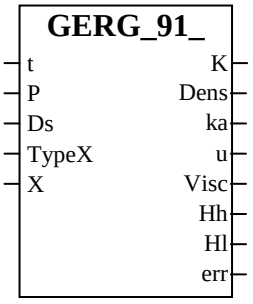
Выходы:

K	REAL	Коэффициент сжимаемости
Dens	REAL	Плотность газа при рабочих условиях (кг/куб.м)
ka	REAL	Показатель адиабаты
u	REAL	Скорость звука в газе (м/с)
Visc	REAL	Динамическая вязкость (мкПа*с)
Hh	REAL	Высшая удельная теплота сгорания (МДж/куб.м)
Hl	REAL	Низшая удельная теплота сгорания (МДж/куб.м)
err	INTEGER	Код ошибки: 0 - нет ошибки; 3 – недопустимый размер массива; 4 – недопустимая ссылка на массив. -1 - недопустимое значение температуры; -2 - недопустимое значение давления; -4 - недопустимое значение плотности; -8 – недопустимое значение доли азота; -16 – недопустимое значение доли углекислого газа.

Назначение

Функциональный блок используется для расчета свойств природного газа по методу GERG-91 мод. (ГОСТ 30319.2-96).

7.3.43 GERG_91_



Входы:

t	REAL	Температура газа (град.С)
P	REAL	Давление газа (кПа)
Ds	REAL	Плотность газа при стандартных условиях (кг/куб.м)
TypeX	BOOLEAN	Тип компонентного состава газа: FALSE – в молярных %; TRUE – в объемных %.
X	#REAL	Имя массива с компонентным составом газа

(элементы типа Real)

Выходы:

K	REAL	Коэффициент сжимаемости
Dens	REAL	Плотность газа при рабочих условиях (кг/куб.м)
ka	REAL	Показатель адиабаты
u	REAL	Скорость звука в газе (м/с)
Visc	REAL	Динамическая вязкость (мкПа*с)
Hh	REAL	Высшая удельная теплота сгорания (МДж/куб.м)
Hl	REAL	Низшая удельная теплота сгорания (МДж/куб.м)
err	INTEGER	Код ошибки: 0 - нет ошибки; 3 – недопустимый размер массива; 4 – недопустимая ссылка на массив. -1 - недопустимое значение температуры; -2 - недопустимое значение давления; -4 - недопустимое значение плотности; -8 – недопустимое значение доли азота; -16 – недопустимое значение доли углекислого газа.

Назначение

Функциональный блок используется для расчета свойств природного газа по методу GERG-91 мод. (ГОСТ 30319.2-96, Изменение №1 к ГОСТ 30319.1-96 от 01.06.2004).

Вход X содержит концентрации компонентов газа в следующем порядке:

X[1]	Метан
X[2]	Этан
X[3]	Пропан
X[4]	н-Бутан
X[5]	и-Бутан
X[6]	н-Пентан
X[7]	и-Пентан
X[8]	н-Гексан
X[9]	н-Гептан
X[10]	н-Октан
X[11]	Ацетилен
X[12]	Этилен
X[13]	Пропилен
X[14]	Бензол
X[15]	Толуол
X[16]	Водород
X[17]	Водяной пар
X[18]	Аммиак
X[19]	Метанол
X[20]	Сероводород
X[21]	Метилмеркаптан
X[22]	Диоксид серы
X[23]	Гелий
X[24]	Неон
X[25]	Аргон
X[26]	Монооксид углерода
X[27]	Азот
X[28]	Воздух
X[29]	Кислород
X[30]	Диоксид углерода

7.3.44 GET_SMS

Входы:

RUN	BOOLEAN	Признак выполнения функционального блока (передача запроса в модем)

Выходы:

READY	BOOLEAN	Признак завершения операции.
FAULT	INTEGER	Код ошибки завершения операции.
AbNum	MESSAGE	Номер абонента - получателя сообщения
Message	MESSAGE	Текст сообщения передаваемого сообщения

Назначение

Функциональный блок используется для приема СМС сообщений через встроенный модем в мастер-модуле M1012E
По переднему фронту на входе RUN, блок читает очередное СМС из модема
и выдает его на выход Message, и номер отправителя на выход AbNum, до тех пор пока RUN= TRUE.
После перехода RUN в FALSE, Message и AbNum очищаются.
По следующему переднему фронту выдается новая СМС.
Если FAULT равно нулю и READY равно TRUE, СМС считана из модема
и установлена на выходах блока Message и AbNum .
Если FAULT не равно нулю, чтение СМС не удалось выполнить.

Примечание
Блок рекомендуется использовать совместно с оператором SYSTEM(20,4),
который возвращает не нулевое значение, при наличие принятых смс в модеме.
Если SYSTEM(20,4) возвращает ноль, блок считывает последнее принятое СМС.

Блок GET_SMC и оператор SYSTEM(20,4) предназначен для использования в мастер-модуле M1012E со встроенным GSM модемом.

Для фильтрации сообщений, принимаются только сообщения начинающиеся с текстовой последовательности: TREI
Корректно распознаются сообщения содержащие только латинские буквы.

Пример на ST:

```
if SYSTEM(20,4) <> 0 then           //наличие принятых смс в модеме
   if i_sms_busy = false then
      i_sms_run:=true;
   end_if;
end_if;

i_sms(i_sms_run);   // i_sms - блок типа GET_SMS

if i_sms_run then
   i_sms_busy:=true;
   i_sms_ready:=i_sms.ready;
   i_sms_fault:=i_sms.fault;

   if (i_sms_ready = true) or (i_sms_fault <> 0) then
      i_sms_run:=false;

      if i_sms_fault = 0 then
         i_sms_msg:=i_sms.Message;  // текст полученного сообщения
         i_sms_num:=i_sms.AbNum;         // номер абонента отправителя сообщения
         end_if;
```

```
     end_if;

   end_if;

else
   i_sms_busy:=false;
end_if;
```

7.3.45 GETTIME

Входы:

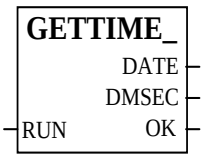
RUN	BOOLEAN	Признак необходимости обновления данных
CMOS	BOOLEAN	Признак чтения времени из CMOS

Выходы:

DATE	INTEGER	Дата (день, месяц, год)
DMSEC	INTEGER	Миллисекунда с начала суток
OK	INTEGER	Результат операции

Назначение
Получение текущего времени.

7.3.46 GETTIME_



Входы:

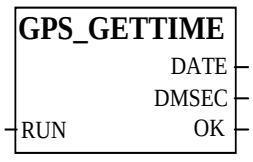
RUN	BOOLEAN	Признак необходимости обновления данных

Выходы:

DATE	INTEGER	Дата (день, месяц, год)
DMSEC	INTEGER	Миллисекунда с начала суток
OK	INTEGER	Результат операции

Назначение
Получение текущего времени с использованием поправки времени.

7.3.47 GPS_GETTIME



Входы:

RUN	BOOLEAN	Запуск блока TRUE - выполнить,FALSE - не выполнять

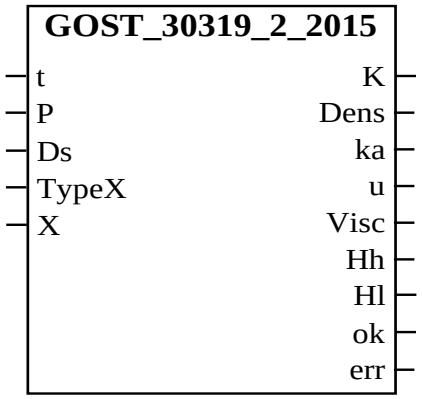
Выходы:

DATE	INTEGER	Текущая дата (дент, месяц, год)
DMSEC	INTEGER	Миллисекунда с начала суток
OK	INTEGER	Результат операции

Назначение
Получение точного времени с GPS приемника. При использовании данного функционального блока на контроллере должна быть запущена задача gps.

Функциональный блок оставлен для совместимости. Более новая версия блока - DRV_GETTIME.

7.3.48 GOST_30319_2_2015



Входы:

t	Integer	Температура среды (град С)
P	Integer	Давление среды (КПа)
Ds	Real	Плотность газа при стандартных условиях (кг / куб.м)
TypeX	Boolean	Тип компонентного состава газа: FALSE - в молярных % TRUE - в объемных %
X	#Real	Ссылка на массив с компонентным составом газа

Выходы:

K	Real	Коэффициент сжимаемости
Dens	Real	Плотность газа при рабочих условиях (кг / куб.м)
ka	Real	Показатель адиабаты
u	Real	Скорость звука в газе (м / с)
Visc	Real	Динамическая вязкость (мкПа * с)
Hh	Real	Высшая удельная теплота сгорания (МДж / куб.м)
Hl	Real	Низшая удельная теплота сгорания (МДж / куб.м)
ok	Boolean	Код завершения FALSE – нет ошибки TRUE – есть ошибки
err	Integer	Код ошибки

При использовании данного функционального блока на контроллере должна быть запущена задача gost_30319_2_2015

Назначение
Расчет свойств природного газа (метод GERG-91 мод.).
(ГОСТ 30319.2-96, Изменение №1 к ГОСТ 30319.1-96 от 01.06.2004, ГОСТ 30319.2-2015).
Диапазон применения 0.1 <= P <= 7.5 МПа, 250 <= T <= 350K.

Вход X содержит концентрации компонентов газа в следующем порядке:

- X[1] Метан
- X[2] Этан
- X[3] Пропан
- X[4] н-Бутан
- X[5] и-Бутан
- X[6] н-Пентан
- X[7] и-Пентан
- X[8] н-Гексан
- X[9] н-Гептан
- X[10] н-Октан
- X[11] Ацетилен
- X[12] Этилен
- X[13] Пропилен
- X[14] Бензол
- X[15] Толуол
- X[16] Водород
- X[17] Водяной пар
- X[18] Аммиак
- X[19] Метанол
- X[20] Сероводород
- X[21] Метилмеркаптан
- X[22] Диоксид серы
- X[23] Гелий
- X[24] Неон
- X[25] Аргон
- X[26] Монооксид углерода
- X[27] Азот
- X[28] Воздух
- X[29] Кислород
- X[30] Диоксид углерода

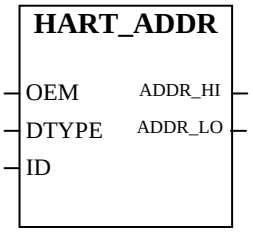
Код завершения:

- FALSE – нет ошибки
- TRUE – есть ошибки

Код ошибки:

- 0 – нет ошибки;
- 1 – ошибка обмена с задачей связи;
- 2 – недопустимый тип массива;
- 3 – недопустимый размер массива;
- 4 – недопустимая ссылка на массив;
- -1 – недопустимое значение температуры (температура газа: от -23.15 до +76.85 град. C);
- -2 – недопустимое значение давления (абсолютное давление газа: от 100 до 7500 кПа);
- -4 – недопустимое значение плотности (значение плотности газа при стандартных условиях: от 0,66 до 1,05 кг / куб.м);
- -8 – недопустимое значение доли азота (молярная доля азота - от 0 до 20 %);
- -16 – недопустимое значение доли углекислого газа (молярная доля углекислого газа - от 0 до 20 %);
- -32 – Недопустимое значение доли метана (молярная доля метана - от 70 до 100 %);
- -64 – Недопустимое значение доли этана (молярная доля этана - от 0 до 20 %);
- -128 – Недопустимое значение доли пропана (молярная доля пропана - от 0 до 3,5 %);
- -256 – Недопустимое значение доли бутанов в сумме (молярная доля бутанов - от 0 до 1,5 %);
- -1024 – Недопустимое значение доли остальных компонентов (молярная доля остальных компонентов - от 0 до 0,25 %)
- -2048 – Недопустимое значение доли Водорода или Гелия (молярная доля водорода или гелия - от 0 до 0,05 %);
- -4096 – Недопустимое значение доли пентанов в сумме (молярная доля пентанов - от 0 до 0,5 %);
- -8192 – Недопустимое значение доли гексана (молярная доля гексана - от 0 до 0,1 %).

7.3.49 HART_ADDR



Входы:

OEM	INTEGER	Код изготовителя устройства HART
DTYPE	INTEGER	Код тип устройства HART
ID	INTEGER	Идентификационный номер устройства HART

Выходы:

ADDR_HI	INTEGER	Адрес устройства HART старшая часть
ADDR_LO	INTEGER	Адрес устройства HART младшая часть

Назначение

Функциональный блок используется для получения адреса устройства HART из его идентификационных данных.

При использовании данного функционального блока входные данные можно получить из результата выполнения функционального блока HART_0 или из документации и маркировок устройства HART.

7.3.50 HART_TRANSIT

Входы

RUN	BOOLEAN	Запуск блока TRUE - выполнить, FALSE - не выполнять
CHN	INTEGER	Номер канала HART
ICOUNT	INTEGER	Количество входных данных в байтах
IADDR	INTEGER	Адрес первой переменной входных данных в словаре обмена модуля
OADDR	INTEGER	Адрес первой переменной выходных данных в словаре обмена модуля

Выходы

OK	BOOLEAN	Признак завершения операции
FAULT	INTEGER	Код ошибки 0 – нет ошибки; 1 – устройство не отвечает (количество повторов превысило допустимое значение); 2 – ошибка интерфейса (ошибка фрейма); 3 – ошибка контрольной суммы; 4 – ошибка ответного пакета; 5 – мезонин не найден; 6 – аппаратная ошибка мезонина; 7 – обрыв линии; 8 - ошибка контрольной суммы входного пакета; 9 - ошибка размера ответного пакета.
OCOUNT	INTEGER	Количество выходных данных в байтах

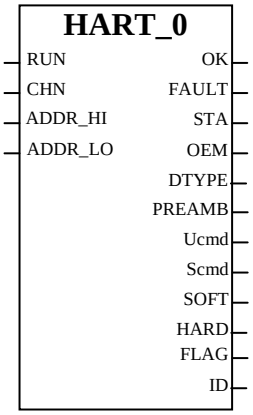
Назначение
Отправка транзитного запроса на устройство HART.

Примечание:
Для экономии памяти в словаре обмена адреса IADDR и OADDR могут быть одинаковы.
Запросы и ответы в словаре обмена модуля лежат в целых переменных в упакованном виде.
По старшему адресу в целой переменной лежит первый байт, по младшему - последний.

Пример: запрос HART [02 80 00 00 82] необходимо упаковать в 2 переменные:
16#02800000
16#82000000

ВНИМАНИЕ: Входные и выходные данные HART не должны и не содержат преамбул.

7.3.51 HART_0



Входы:

RUN	BOOLEAN	Запуск функционального блока
CHN	INTEGER	Номер канала HART
ADDR_HI	INTEGER	Адрес устройства HART старшая часть
ADDR_LO	INTEGER	Адрес устройства HART младшая часть

Выходы:

OK	BOOLEAN	Признак завершения операции
FAULT	INTEGER	Код ошибки (табл.1)
STA	INTEGER	Статус устройства HART (табл.2)
OEM	INTEGER	Код изготовителя устройства HART
DTYPE	INTEGER	Код тип устройства HART
PREAMB	INTEGER	Количество преамбул устройства HART
Ucmd	INTEGER	Версия универсальных команд устройства HART
Scmd	INTEGER	Версия специальных команд устройства HART
SOFT	INTEGER	Версия программного обеспечения устройства HART
HARD	INTEGER	Версия аппаратной части устройства HART
FLAG	INTEGER	Флаги функций устройства HART
ID	INTEGER	Идентификационный номер устройства HART

Назначение

Функциональный блок используется для получения расширенного кода устройства HART, версии и идентификационного номера.

Примечание:

При использовании с S240H/M541HR:

- на вход «CHN» должен подаваться - номер модуля S240H/M541HR.
- на вход «ADDR_HI» должен подаваться адрес канала на модуле в диапазоне [0..15].

Таблица 1. Коды ошибок:

Код	Описание
0	Нет ошибок
1	Устройство не отвечает (количество повторов превысило допустимое значение). Ошибка обмена с драйвером (для S240H/M541HR)
2	Ошибка интерфейса (ошибка фрейма). Нет ответа (отработан таймаут) (для S240H/M541HR)
3	Ошибка контрольной суммы. Ошибка системы (для S240H/M541HR)
4	Ошибка ответного пакета. Ошибка драйвера (для S240H/M541HR)
5	Мезонин не найден. Операция выполняется (при асинхронном режиме обмена) (для S240H/M541HR)
6	Аппаратная ошибка мезонина. Некорректные данные (нарушение формата, протокола и т.п.) (для S240H/M541HR)

Таблица 1. Коды ошибок (продолжение):

7	Обрыв линии. Операция прервана (для S240H/M541HR)
8	Некорректный запрос (превышение допустимой длины) (для S240H/M541HR)
9	Некорректный адрес модуля S240H/M541HR
10	Ошибка выделения памяти
11	Некорректный описатель модуля S240H/M541HR
12	Некорректный формат ответа
13	Некорректная контрольная сумма ответа
14	Некорректная длина ответа
15	Некорректная команда в ответе
16	Модуль S240H/M541HR диагностировал ошибку (байт 0 = 16, байт [1..2] - статус модуля)
17	Модуль S240H/M541HR не в рабочем режиме
18	Обмен запрещен
19	Получен ответ-исключение от HART устройства
20	Время ожидания ответа от датчика истекло
21	Ошибка формата пакета на отправку по HART
22	Ошибка формата принятого пакета по HART
23	Ошибка контрольной суммы принятого по HART пакета
24	Ошибка физического интерфейса HART

Таблица 2. Кодирование статуса

Кодирование статуса.

Первый байт:

БИТ 7 = 1 ОШИБКИ ПЕРЕДАЧИ ДАННЫХ:	
Бит 6	ошибка по четности
Бит 5	ошибка переполнения
Бит 4	ошибка формирования фрейма
Бит 3	ошибка контрольной суммы
Бит 2	(зарезервирован)
Бит 1	переполнен буфер приемника
Бит 0	неопределен

Второй байт:

Бит 7	
Бит 6	
Бит 5	
Бит 4	все 0
Бит 3	
Бит 2	
Бит 1	
Бит 0	

Первый байт:

Бит 7 = 0 ОШИБКИ КОМАНД:	
	Биты с 6 по 0 (не побитовое отображение)
0	ошибка, не характерная для команд
1	(не определено)
2	неверный выбор
3	переданный параметр слишком велик
4	переданный параметр слишком мал
5	получено слишком мало байт данных
6	ошибка команды, специфической для датчика
7	в режиме, защищенном от записи
8-15	ошибки, характерные для команд (см. ниже табл.4)
16	доступ ограничен
32	устройство занято
64	команда не задействована

Второй байт:

Бит 7 (80 в шестнадцатеричном виде) неисправность прибора	
Бит 6	изменена конфигурация
Бит 5	холодный старт
Бит 4	
Бит 3	фиксирован выходной ток
Бит 2	насыщение аналогового выходного сигнала
Бит 1	переменная (не первичная) вышла за ограничения
Бит 0	первичная переменная вышла за ограничения

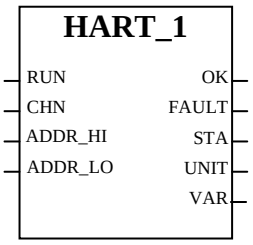
Показаны шестнадцатеричные эквиваленты, в предположении, что присутствует только одна характеристика статуса.

Эти коды имеют различное значение для различных команд. В следующей таблице перечислены некоторые из этих значений. Обращайтесь к полной документации HART за информацией о том, какие коды и значения используются для каждой команды.

Коды ошибок, специфичные для команд.

КОД	ЗНАЧЕНИЕ
8	Неудача при обновлении Обновление в процессе работы Значение установлено равным ближайшему возможному
9	Параметр процесса слишком велик Нижнее значение диапазона слишком велико Не в режиме фиксированного тока
10	Параметр процесса слишком мал Нижнее значение диапазона слишком мало Не поддерживается режим моноканала
11	В режиме моноканала Неверный код переменной датчика Верхнее значение диапазона слишком велико
12	Неверный код единиц измерения Верхнее значение диапазона слишком мало
13	Оба значения диапазона выходят за ограничения
14	Введенное верхнее значение диапазона выходит за ограничение Интервал слишком мал

7.3.52 HART_1



Входы:

RUN	BOOLEAN	Запуск функционального блока
CHN	INTEGER	Номер канала HART
ADDR_HI	INTEGER	Адрес устройства HART старшая часть
ADDR_LO	INTEGER	Адрес устройства HART младшая часть

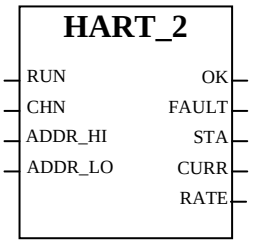
Выходы:

OK	BOOLEAN	Признак завершения операции
FAULT	INTEGER	Код ошибки (см.табл.1 в описании HART_0)
STA	INTEGER	Статус устройства HART (см.табл.2 в описании HART_0)
UNIT	INTEGER	Код единиц измерения первичной переменной
VAR	REAL	Первичная переменная устройства HART

Назначение
Функциональный блок используется для получения первичной переменной устройства HART..

Примечание:
На входы ADDR_HI, ADDR_LO подается результат вызова блока HART_ADDR.
При использовании с S240H/M541HR: на вход «CHN» должен подаваться - номер модуля S240H/M541HR.

7.3.53 HART_2



Входы:

RUN	BOOLEAN	Запуск функционального блока
CHN	INTEGER	Номер канала HART
ADDR_HI	INTEGER	Адрес устройства HART старшая часть
ADDR_LO	INTEGER	Адрес устройства HART младшая часть

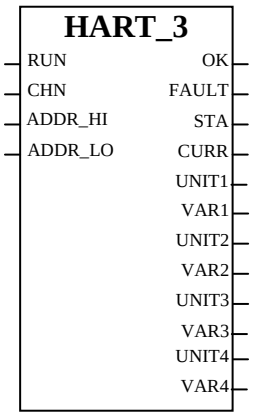
Выходы:

OK	BOOLEAN	Признак завершения операции
FAULT	INTEGER	Код ошибки (см.табл.1 в описании HART_0)
STA	INTEGER	Статус устройства HART (см.табл.2 в описании HART_0)
CURR	REAL	Ток первичной переменной(мА)
RATE	REAL	Процент диапазона (%)

Назначение
Функциональный блок используется для получения тока и процента диапазона первичной переменной устройства HART

Примечание:
На входы ADDR_HI, ADDR_LO подается результат вызова блока HART_ADDR.
При использовании с S240H/M541HR: на вход «CHN» должен подаваться - номер модуля S240H/M541HR.

7.3.54 HART_3



Входы:

RUN	BOOLEAN	Запуск функционального блока
CHN	INTEGER	Номер канала HART
ADDR_HI	INTEGER	Адрес устройства HART старшая часть
ADDR_LO	INTEGER	Адрес устройства HART младшая часть

Выходы:

OK	BOOLEAN	Признак завершения операции
FAULT	INTEGER	Код ошибки (см.табл.1 в описании HART_0)
STA	INTEGER	Статус устройства HART (см.табл.2 в описании HART_0)
CURR	REAL	Ток первичной переменной (мА)
UNIT1	INTEGER	Код единиц измерения первичной переменной
VAR1	REAL	Первичная переменная устройства HART
UNIT2	INTEGER	Код единиц измерения второй переменной
VAR2	REAL	Вторая переменная устройства HART
UNIT3	INTEGER	Код единиц измерения третьей переменной
VAR3	REAL	Третья переменная устройства HART
UNIT4	INTEGER	Код единиц измерения четвертой переменной
VAR4	REAL	Четвертая переменная устройства HART

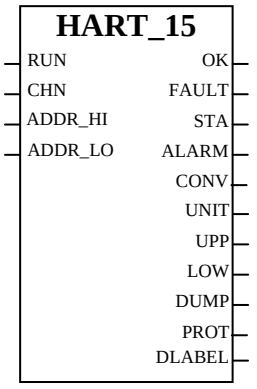
Назначение

Функциональный блок используется для получения значения четырех предопределенных динамических переменных и тока первичной переменной устройства HART.

Примечание:

На входы ADDR_HI, ADDR_LO подается результат вызова блока HART_ADDR.
При использовании с S240H/M541HR: на вход «CHN» должен подаваться - номер модуля S240H/M541HR.

7.3.55 HART_15



Входы:

RUN	BOOLEAN	Запуск функционального блока
CHN	INTEGER	Номер канала HART
ADDR_HI	INTEGER	Адрес устройства HART старшая часть
ADDR_LO	INTEGER	Адрес устройства HART младшая часть

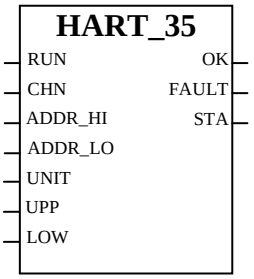
Выходы:

OK	BOOLEAN	Признак завершения операции
FAULT	INTEGER	Код ошибки (см.табл.1 в описании HART_0)
STA	INTEGER	Статус устройства HART (см.табл.2 в описании HART_0)
ALARM	INTEGER	Код выбора тревоги устройства HART
CONV	INTEGER	Код функции преобразования устройства HART
UNIT	INTEGER	Код единиц измерения диапазона первичной переменной
UPP	REAL	Верхнее значение диапазона устройства HART
LOW	REAL	Нижнее значение диапазона устройства HART
DUMP	REAL	Величина демпфирования устройства HART
PROT	INTEGER	Код защиты от записи устройства HART
DLABEL	INTEGER	Код метки дистрибьютора устройства HART

Назначение
Функциональный блок используется для получения информации о выходе первичной переменной устройства HART.

Примечание:
На входы ADDR_HI, ADDR_LO подается результат вызова блока HART_ADDR.
При использовании с S240H/M541HR: на вход «CHN» должен подаваться - номер модуля S240H/M541HR.

7.3.56 HART_35



Входы:

RUN	BOOLEAN	Запуск функционального блока
CHN	INTEGER	Номер канала HART
ADDR_HI	INTEGER	Адрес устройства HART старшая часть
ADDR_LO	INTEGER	Адрес устройства HART младшая часть
UNIT	INTEGER	Код единиц измерения диапазона устройства HART
UPP	REAL	Верхнее значение диапазона устройства HART
LOW	REAL	Нижнее значение диапазона устройства HART

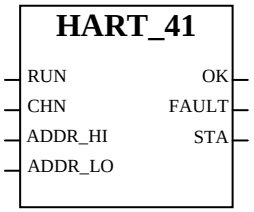
Выходы:

OK	BOOLEAN	Признак завершения операции
FAULT	INTEGER	Код ошибки (см.табл.1 в описании HART_0)
STA	INTEGER	Статус устройства HART (см.табл.2 в описании HART_0)

Назначение
Функциональный блок используется для записи границ диапазона первичной переменной устройства HART.

Примечание:
На входы ADDR_HI, ADDR_LO подается результат вызова блока HART_ADDR.
При использовании с S240H/M541HR: на вход «CHN» должен подаваться - номер модуля S240H/M541HR.

7.3.57 HART_41



Входы:

RUN	BOOLEAN	Запуск функционального блока
CHN	INTEGER	Номер канала HART
ADDR_HI	INTEGER	Адрес устройства HART старшая часть
ADDR_LO	INTEGER	Адрес устройства HART младшая часть

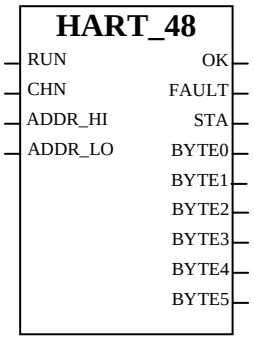
Выходы:

OK	BOOLEAN	Признак завершения операции
FAULT	INTEGER	Код ошибки (см.табл.1 в описании HART_0)
STA	INTEGER	Статус устройства HART (см.табл.2 в описании HART_0)

Назначение
Функциональный блок используется для запуска самотестирования устройства HART.

Примечание:
На входы ADDR_HI, ADDR_LO подается результат вызова блока HART_ADDR.
При использовании с S240H/M541HR: на вход «CHN» должен подаваться - номер модуля S240H/M541HR.

7.3.58 HART_48



Входы:

RUN	BOOLEAN	Запуск функционального блока
CHN	INTEGER	Номер канала HART
ADDR_HI	INTEGER	Адрес устройства HART старшая часть
ADDR_LO	INTEGER	Адрес устройства HART младшая часть

Выходы:

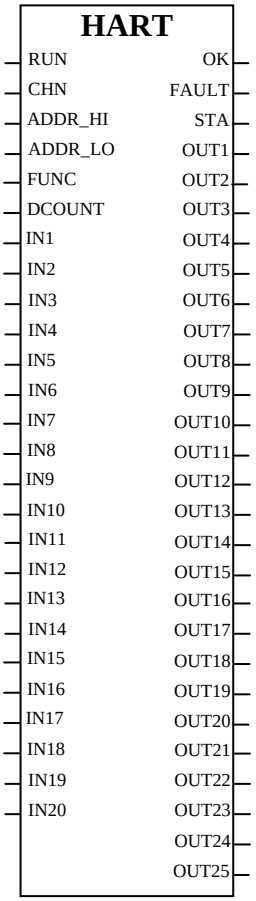
OK	BOOLEAN	Признак завершения операции
FAULT	INTEGER	Код ошибки (см.табл.1 в описании HART_0)
STA	INTEGER	Статус устройства HART (см.табл.2 в описании HART_0)
BYTE0	INTEGER	Байт 0 в ответе (см. описание устройства HART)
BYTE1	INTEGER	Байт 1 в ответе (см. описание устройства HART)
BYTE2	INTEGER	Байт 2 в ответе (см. описание устройства HART)
BYTE3	INTEGER	Байт 3 в ответе (см. описание устройства HART)
BYTE4	INTEGER	Байт 4 в ответе (см. описание устройства HART)
BYTE5	INTEGER	Байт 5 в ответе (см. описание устройства HART)

Назначение
Функциональный блок используется для получения дополнительного статуса устройства HART.

Примечание:

На входы ADDR_HI, ADDR_LO подается результат вызова блока HART_ADDR.
При использовании с S240H/M541HR: на вход «CHN» должен подаваться - номер модуля S240H/M541HR.

7.3.59 HART



Входы:

RUN	BOOLEAN	Запуск функционального блока
CHN	INTEGER	Номер канала HART
ADDR_HI	INTEGER	Адрес устройства HART старшая часть
ADDR_LO	INTEGER	Адрес устройства HART младшая часть
FUNC	INTEGER	Код запроса
DCOUNT	INTEGER	Количество передаваемых данных
IN1	INTEGER	Байт 1 передаваемых данных
…..		…..
IN20	INTEGER	Байт 20 передаваемых данных

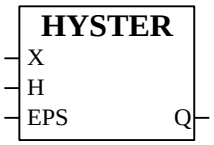
Выходы:

OK	BOOLEAN	Признак завершения операции
FAULT	INTEGER	Код ошибки (см.табл.1 в описании HART_0)
STA	INTEGER	Статус устройства HART (см.табл.2 в описании HART_0)
OUT1	INTEGER	Байт 1 принятых данных
….		….
OUT20	INTEGER	Байт 25 принятых данных

Назначение
Функциональный блок используется для формирования и отправки пользовательского запроса к устройству HART.

Примечание:
На входы ADDR_HI, ADDR_LO подается результат вызова блока HART_ADDR.
При использовании с S240H/M541HR: на вход «CHN» должен подаваться - номер модуля S240H/M541HR.

7.3.60 HYSTER



Входы:

X	REAL	Основной вход
H	REAL	Порог срабатывания
EPS	REAL	Гистерезис

Выход:

Q	BOOLEAN	Тревога: TRUE, если пороговый элемент сработал

Назначение

Функциональный блок HYSTER применяется для контроля за выходом сигнала из ограниченной сверху области допустимых значений.

Описание алгоритма

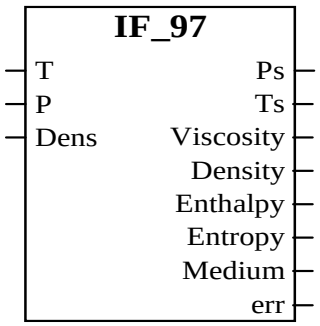
Входной сигнал X подается на звено порогового элемента с порогом срабатывания H и гистерезисом EPS. Звено порогового элемента срабатывает, когда X>H+EPS, при этом появляется дискретный сигнал на выходе Q=TRUE до тех пор, пока входной сигнал не уменьшится до значения X<H-EPS. Логика работы блока описывается таблицей:

X	Q
X < H - EPS	0
X > H + EPS	1
$H - EPS \le X \le EPS + H$	Q_{i-1}

Здесь Q_{i-1} – предыдущее значение выходного сигнала.

На входах H, EPS задается соответственно порог срабатывания и гистерезис. Значение параметра EPS необходимо задавать только положительным (EPS≥0).

7.3.61 IF_97



Входы:

T	INTEGER	Температура среды (град С)
P	INTEGER	Давление среды (КПа)
Dens	INTEGER	Плотность среды (кг/м3)

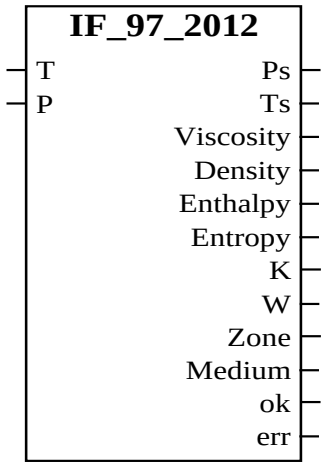
Выходы:

Ps	REAL	Давление насыщения (МПа)
Ts	REAL	Температура насыщения (К)
Viscosity	REAL	Динамическая вязкость (Па * с)
Density	REAL	Плотность (кг/м3)

Enthalpy	REAL	Энтальпия (КДж / кг)
Entropy	REAL	Энтропия (КДж / (кг*К))
Medium	INTEGER	Тип измеряемой среды: 1 - вода 2 - насыщенный пар 3 - перегретый пар
err	INTEGER	Код завершения операции 0 - выполнено успешно 1 - данные не достоверны

Назначение
Расчет термодинамических свойств воды и пара по методу IAPWS-IF97.

7.3.62 IF_97_2012



Входы:

T	INTEGER	Температура среды (град С)
P	INTEGER	Давление среды (КПа)

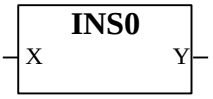
Выходы:

Ps	REAL	Давление насыщения (МПа)
Ts	REAL	Температура насыщения (К)
Viscosity	REAL	Динамическая вязкость (Па * с)
Density	REAL	Плотность (кг/м3)
Enthalpy	REAL	Энтальпия (КДж / кг)
Entropy	REAL	Энтропия (КДж / (кг*К))
K	REAL	Показатель адиабаты
W	REAL	Скорость звука (м / с)
Zone	INTEGER	Зона расчета
Medium	INTEGER	Тип измеряемой среды: 1 - вода 2 - насыщенный пар 3 - перегретый пар
ok	BOOLEAN	Код завершения FALSE – нет ошибки TRUE – есть ошибки
err	INTEGER	Код ошибки 0 – нет ошибки; 1 – ошибка обмена с задачей связи; 2 – недопустимое значение давления; 3 – недопустимое значение температуры; 4 – ошибка вычислений

При использовании данного функционального блока на контроллере должна быть запущена задача **if_97_2012**

Назначение
Расчет термодинамических свойств воды и пара по методу IAPWS-IF97 (2012). ГСССД 187-99
Диапазон применения 0 <= T <=800 C при P <= 100000 КПа, 800 <= T <=2000 C при P <= 50000 КПа.

7.3.63 INS0



Вход:

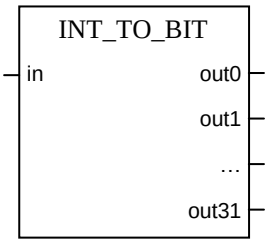
X	REAL	Основной вход блока

Выход:

Y	REAL	Основной выход блока

Описание алгоритма:
Этот блок воспроизводит на выходе входное значение. Однако при каждом изменении входного значения выход обнуляется на один цикл работы модуля и только в следующем цикле становится равным входу.

7.3.64 INT_TO_BIT



Входы:

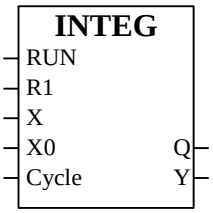
in	INTEGER	Упакованный ввод

Выход:

out0	INTEGER	Дискретный выход 0
out1	INTEGER	Дискретный выход 1
…		
out31	INTEGER	Дискретный выход 31

Назначение
Распаковка целого значения в булевские.

7.3.65 INTEG



Входы:

RUN	BOOLEAN	Режим работы: TRUE - интегрировать, FALSE - держать
R1	BOOLEAN	Сигнал сброса
X	REAL	Основной вход
X0	REAL	Начальное значение
Cycle	INTEGER	Период работы блока (мсек)

Выходы:

Q	BOOLEAN	Инверсия от R1
Y	REAL	Основной выход

Назначение

Функциональный блок INTEG используется для интегрирования и (или) запоминания сигнала.

Описание алгоритма

В режиме работы, когда RUN=TRUE и R1=FALSE, сигнал с входа X проходит через интегрирующее звено и поступает на выход Y. Передаточная функция интегрирующего звена:

$$W(p) = \frac{Y(p)}{X(p)} = \frac{1}{p},$$

При состоянии входа RUN=FALSE интегрирования не происходит, значение выхода Y_i не изменяется и равно предыдущему значению Y_{i-1}, если R1=FALSE. Y_i – значение выхода в текущем цикле, Y_{i-1} – значение выхода в предыдущем цикле. Если R1=TRUE выход блока Y равен значению входа X0. На выходе Q формируется инверсия от входного сигнала R1.

Параметр Cycle является периодом работы блока и периодом интегрирования. При значении Cycle≤0 период работы блока равен циклу контроллера.

7.3.66 LATCH

Входы:

CLK	BOOLEAN	Команда запоминания
X	REAL	Основной вход

Выходы:

Q	BOOLEAN	Признак запоминания
Y	REAL	Основной выход

Назначение

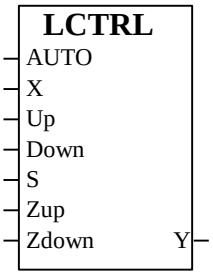
Функциональный блок LATCH используется для запоминания аналогового сигнала.

Описание алгоритма

На вход блока X подается аналоговый сигнал, который нужно запомнить. На вход CLK подается команда запоминания.

Запись осуществляется по переднему фронту дискретного сигнала CLK, т.е. в момент перехода CLK из состояния логического 0 в состояние логической 1. В этот момент текущее значение входного сигнала X запоминается и выход Q принимает состояние TRUE на 1 цикл контроллера. Запомненное значение остается неизмененным вплоть до прихода нового фронта сигнала CLK. Сигнал на выходе Y соответствует запомненному значению сигнала X.

7.3.67 LCTRL



Входы:

AUTO	BOOLEAN	Режим работы: TRUE -автоматический, FALSE -ручной
X	REAL	Значение выхода в автоматическом режиме
Up	BOOLEAN	Увеличить управляющее воздействие
Down	BOOLEAN	Уменьшить управляющее воздействие
S	REAL	Приращение выхода в ручном режиме
Zup	BOOLEAN	Сигнал запрета в направлении "Больше"
Zdown	BOOLEAN	Сигнал запрета в направлении "Меньше"

Выход:

Y	REAL	Управляющее воздействие

Назначение

Функциональный блок LCTRL используется в составе каскадного регулятора. Он необходим, если предусматриваться ручное управление импульсным регулятором RIM.

Описание алгоритма

При автоматическом режиме работы AUTO=TRUE выход блока Y равен входу X. При ручном режиме выход Y (управляющее воздействие) увеличивается по абсолютному значению на величину S в каждом цикле контроллера в соответствии со следующей зависимостью:

$$Y_i=Y_{i-1}+S,$$

где Y_i – значение Y в текущем цикле, Y_{i-1} – значение Y в предыдущем цикле, S – величина приращения.

Знак управляющего воздействия Y зависит от состояния дискретных входов Up и Down. Если Up=TRUE управляющее воздействие Y положительно, если Down=TRUE управляющее воздействие Y отрицательно. В ситуации, когда Up=Down (оба входа равны логическому 0 или логической 1) выход Y обнуляется. Если Zup=TRUE и Up=TRUE или Zdown=TRUE и Down=TRUE, то управляющее воздействие Y=0.

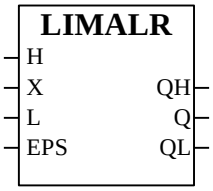
Сигнал с выхода блока Y подается на вход X0 импульсного регулятора RIM. На вход X подается сигнал с выхода RIM. Следовательно, в автоматическом режиме выход блока равен выходу регулятора.

В момент переключения на ручной режим управляющее воздействие Y равно 0, следовательно переход на ручной режим выполняется безударно. Поскольку сигнал с выхода RIM поступает на вход блока широтно-импульсной модуляции PWM, увеличение абсолютного значения управляющего воздействия Y приведет к увеличению длительности импульсов. Таким образом, длительность импульсов будет увеличиваться в каждом цикле до тех пор, пока временное расстояние между ними больше минимально допустимой паузы Tmps, установленной для блока PWM. Как только расстояние между импульсами станет меньше, чем Tmps, импульсы слипнутся и выход, соответствующий знаку управляющего воздействия Y будет постоянно включен. Направление перемещения ИМ будет зависеть от состояния входных сигналов Up и Down блока LCTRL. При состоянии входа Up=TRUE ИМ перемещается по направлению “больше”, если Down=TRUE по направлению “меньше”. Если Up=Down ИМ не перемещается.

Дискретные входы Zup и Zdown блока LCTRL предназначены для запрещения перемещения ИМ по одному из направлений. Если Zup=TRUE или Zdown=TRUE ИМ не будет перемещаться по направлению “больше” или “меньше” соответственно.

Примечание: значение $S \leq 0$ воспринимается алгоритмом как S=0.

7.3.68 LIMALR



Входы:

H	REAL	Верхний порог срабатывания
X	REAL	Основной вход
L	REAL	Нижний порог срабатывания
EPS	REAL	Гистерезис

Выходы:

QH	BOOLEAN	Верхняя тревога: TRUE -X выше границы H
Q	BOOLEAN	Тревога: TRUE -X вне границ
QL	BOOLEAN	Нижняя тревога: TRUE -X ниже границы L

Назначение

Функциональный блок LIMALR или «нуль-орган» используется для контроля за выходом сигнала из области допустимых значений, ограниченной верхним и нижним пределами.

Описание алгоритма

Функциональный блок LIMALR реализует контроль входного аналогового сигнала X на достоверность. Значение сигнала считается достоверным, если выполняется условие:

$$L \le X \le H,$$

где H, L – верхняя и нижняя границы достоверности. Если $H \le L$ блок работать не будет.

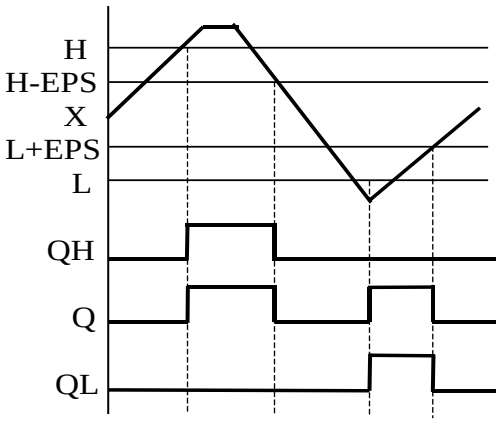
При выходе аналогового сигнала за любую из предварительно заданных границ логический признак QH или QL, сигнализирующий о выходе за конкретную границу, переходит в состояние TRUE. Также в состояние TRUE переходит другой логический признак Q, сигнализирующий о том, что сигнал находится вне границ. При нахождении в пределах границ все выходы функционального блока находятся в состоянии FALSE.

Логика работы блока поясняется таблицей и временной диаграммой:

X	QH	QL
X < H-EPS	0	*
X ≥ H	1	*
H-EPS ≤ X < H	QH_{i-1}	*
X > L+EPS	*	0
X ≤ L	*	1
L+EPS ≥ X > L	*	QL_{i-1}

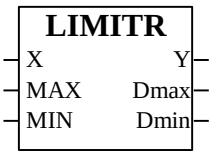
* - выходной сигнал не зависит от данного условия

Временная диаграмма



Примечание: Значение параметра EPS задается только положительное (EPS≥0).

7.3.69 LIMITR



Входы:

X	REAL	Основной вход алгоритма
MAX	REAL	Верхняя граница
MIN	REAL	Нижняя граница

Выходы:

Y	REAL	Основной выход алгоритма
Dmax	BOOLEAN	Достижение верхней границы
Dmin	BOOLEAN	Достижение нижней границы

Назначение

Функция LIMITR используется для ограничения верхней и (или) нижней границы диапазона изменения сигнала.

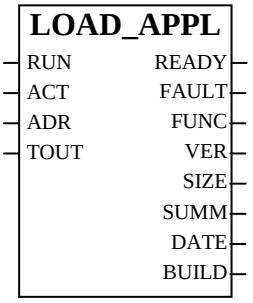
Описание алгоритма

Алгоритм содержит ограничитель верхнего и нижнего значения сигнала. На двух дискретных выходах Dmax и Dmin фиксируется достижение сигналом верхней и нижней границы ограничения.
Работа алгоритма определяется следующей таблицей:

X	Y	Dmax	Dmin
MAX>X>MIN	X	0	0
X≥MAX	MAX	1	0
X≤MIN	MIN	0	1

Уровни ограничений задаются входами MAX, MIN. Если MAX≤MIN, то выход Y=X.

7.3.70 LOAD_APPL



Входы:

RUN	BOOLEAN	Запуск функционального блока
ACT	INTEGER	Действие
ADR	INTEGER	Адрес модуля
TOUT	INTEGER	Время ожидания ответа от модуля

Выходы:

READY	BOOLEAN	Признак завершения операции
FAULT	INTEGER	Код ошибки
FUNC	INTEGER	Текущая функция
VER	INTEGER	Версия приложения
SIZE	INTEGER	Размер приложения

SUMM	INTEGER	Контрольная сумма приложения
DATE	INTEGER	Дата сборки (кол-во секунд с 1970г)
BUILD	INTEGER	Идентификатор сборки приложения

Назначение:

Загрузка/чтение приложения с модуля ввода/вывода.

Описание:

Функциональный блок по фронту входа RUN выполняет действие в соответствии с аргументом ACT.

При единичном значении на входе ACT происходит чтение сохраненного приложения модуля в/в, определяемого входом ADR. Поиск файла приложения осуществляется в директории, задаваемой параметром APPL_SAVEDIR=X (в master.ini). В случае успешного выполнения операции на выходы функционального блока выдается информация о приложении. Выход FAULT возвращает следующие коды ошибок:

0 – Нет ошибки
1 – Запуск функционального блока невозможен. Выполняется загрузка /чтение с одного из модулей в/в
2 – Приложение не найдено
3 – Ошибка чтения файла
4 – Нехватка памяти
5 – Неверный формат приложения
6 – Несовпадение контрольной суммы приложения
7 – Неверно задан аргумент ACT

При значении входа ACT, равному 2, выполняется чтение сохраненного приложения и его загрузка на указанный модуль в/в. Вход TOUT определяет время ожидания ответа от модуля в/в (в миллисекундах). При поддержании единичного значения на входе RUN функциональный блок на выход FUNC возвращает код текущего запроса к модулю (функция по протоколу PC – Мастер-модуль), на выход FAULT – код ошибки:

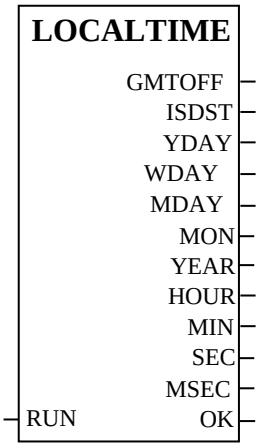
0 – Нет ошибки
1 – Неверный ответ от модуля в/в. Получен ответ с кодом функции, отличным от запроса
2 – Неверные данные в запросе
15 – Превышено время ожидания ответа от модуля в/в
(0x80 | код ошибки) – От модуля в/в получен ответ исключения с данным кодом ошибки

При значении входа ACT, равному 3, выполняется чтение приложения с указанного модуля в/в и его сохранение в директории, задаваемой параметром APPL_SAVEDIR=X (в master.ini). Вход TOUT определяет время ожидания ответа от модуля в/в (в миллисекундах). При поддержании значения 3 на входе RUN функциональный блок на выход FUNC возвращает код текущего запроса к модулю (функция по протоколу PC - Мастер-модуль), на выход FAULT - код ошибки при выполнении данного запроса (аналогично ошибкам при загрузке приложения).

При нулевом значении входа ACT функциональный блок возвращает состояние последней операции.

Примечание: во время загрузки/чтения приложения все “транзитные” запросы к данному модулю в/в блокируются.

7.3.71 LOCALTIME



Входы:

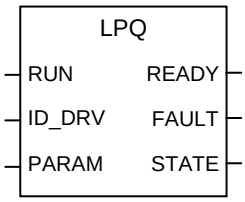
RUN	Boolean	Признак необходимости обновления данных

Выходы:

GMTOFF	Integer	Смещение от GMT
ISDST	Integer	Летнее время
YDAY	Integer	День с начала года (с 1 января) [0..365]
WDAY	Integer	День с начала недели (с воскресенья) [0..6]
MDAY	Integer	День с начала месяца [1..31]
MON	Integer	Месяц с начала года [1..12]
YEAR	Integer	Год
HOUR	Integer	Час с начала суток [0..23]
MIN	Integer	Минута с начала часа [0..59]
SEC	Integer	Секунда с начала минуты [0..59]
MSEC	Integer	Миллисекунда с начала секунды
OK	Integer	Результат операции (0 - выполнено успешно)

Назначение
Получение текущего времени и даты с днем недели. (с использованием поправки).

7.3.72 LPQ



Входы:

RUN	BOOLEAN	Триггер запуска функционального блока
ID_DRV	INTEGER	Идентификатор принтера
PARAM	INTEGER	Параметры запроса 0 – короткая форма (код команды 3) 1 – расширенная форма (код команды 4)

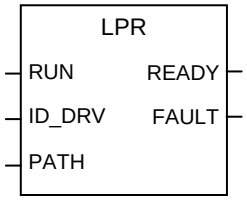
Выходы:

READY	BOOLEAN	Признак завершения операции
FAULT	INTEGER	Код завершения операции: 0 – выполнено успешно 1 – ошибка выделения памяти (некорректные параметры) 2 – превышено время ожидания ответа 3 – системная ошибка при работе с портом

		4 – ошибка инициализации задачи связи 5 – выполняется запрос 6 – ошибка принятых данных 7 – запуск блока невозможен, выполняется запрос
STATE	MESSAGE	Прочитанное состояние

Назначение:
Чтение состояния принтера (на мастер-модуле должна быть запущена задача связи **um_pd**).

7.3.73 LPR



Входы:

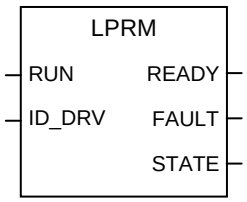
RUN	BOOLEAN	Триггер запуска функционального блока
ID_DRV	INTEGER	Идентификатор принтера
PATH	MESSAGE	Полное имя файла

Выходы:

READY	BOOLEAN	Признак завершения операции
FAULT	INTEGER	Код завершения операции: 0 – выполнено успешно 1 – ошибка выделения памяти (некорректные параметры) 2 – превышено время ожидания ответа 3 – системная ошибка при работе с портом 4 – ошибка инициализации задачи связи 5 – выполняется запрос 6 – ошибка принятых данных 7 – запуск блока невозможен, выполняется запрос

Назначение:
Печать файла (на мастер-модуле должна быть запущена задача связи **um_pd**).

7.3.74 LPRM



Входы:

RUN	BOOLEAN	Триггер запуска функционального блока
ID_DRV	INTEGER	Идентификатор принтера

Выходы:

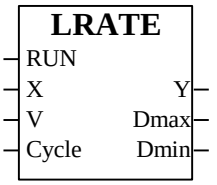
READY	BOOLEAN	Признак завершения операции
FAULT	INTEGER	Код завершения операции: 0 – выполнено успешно 1 – ошибка выделения памяти (некорректные параметры) 2 – превышено время ожидания ответа

3 – системная ошибка при работе с портом
4 – ошибка инициализации задачи связи
5 – выполняется запрос
6 – ошибка принятых данных
7 – запуск блока невозможен, выполняется запрос

STATE	MESSAGE	Прочитанное состояние

Назначение:
Чистка очереди принтера (на мастер-модуле должна быть запущена задача связи **um_pd**).

7.3.75 LRATE



Входы:

RUN	BOOLEAN	Режим работы
X	REAL	Основной вход алгоритма
V	REAL	Максимальная скорость (1/сек)
Cycle	INTEGER	Период работы блока (мсек)

Выход:

Y	REAL	Основной выход алгоритма
Dmax	BOOLEAN	Признак увеличения с превышенной скоростью
Dmin	BOOLEAN	Признак уменьшения с превышенной скоростью

Назначение
Функциональный блок LRATE используется в тех случаях, когда необходимо ограничить скорость изменения сигнала.

Описание алгоритма
При состоянии входа RUN=TRUE сигнал с входа X проходит на выход Y в соответствии со следующей последовательностью преобразований. Определяется скорость изменения значения входа X относительно выхода:

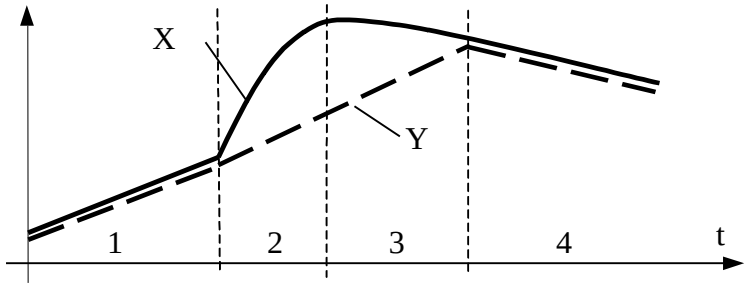
$$Vx = \frac{X - Y}{Tc},$$ где Tc – время цикла.

Если |Vx|≤V, то сигнал X проходит на выход без изменений. Если |Vx|>V (при этом если V>0), то выход Y будет изменяться в соответствии со следующей зависимостью:

$$Y_i = Y_{i-1} + sign(Vx) \cdot V \cdot Tc,$$

где Y_{i-1} – значение выхода в предыдущем цикле; Y_i – обновленное значение выхода в текущем цикле; V – заданное значение ограничения скорости.

Выход Y будет изменяться с ограниченной скоростью V до тех пор, пока не достигнет значения X.



1 - |Vx|<V; Vy=Vx
2 - |Vx|>V; Vy=V
3 - |Vx|<V; Vy=V
4 - |Vx|<V; Vy=Vx

В любой момент времени выходной сигнал Y стремится сравниться с входным сигналом X. Если скорость изменения входного сигнала Vx меньше заданного ограничения V (т.е. Vx < V), то выходной сигнал Y изменяется со скоростью Vy=Vx, оставаясь в каждый момент времени равным сигналу X. Если Vx>V, сигнал Y начинает изменяться с ограниченной скоростью V до тех пор, пока не сравняется с сигналом X.

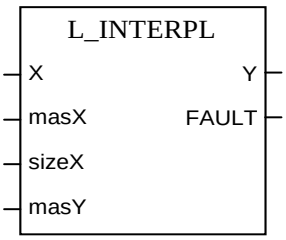
Блок имеет два дискретных выхода Dmax и Dmin. Если скорость Vy меньше заданного ограничения V, сигналы на обоих выходах равны нулю. В противном случае появляется сигнал на выходе Dmax или Dmin, в зависимости от того, увеличивается (изменяется в направлении "больше") или уменьшается (изменяется в направлении "меньше") выходной сигнал Y.

Vy	Dmax	Dmin
|Vy|<V	0	0
Vy=V	1	0
Vy= - V	0	1

При состоянии входа RUN=FALSE выход Y равен входу X, выходы Dmax и Dmin в состоянии логического нуля.

Примечание: при значении Cycle≤0 период работы блока равен циклу контроллера.

7.3.76 L_INTERPL



Входы:

X	REAL	Фактический аргумент
masX	#REAL	Массив аргументов (элементы типа REAL)
sizeX	INTEGER	Размер массива X
masY	#REAL	Массив функций от аргументов X (элементы типа REAL)

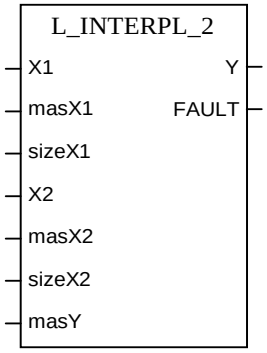
Выход:

Y	REAL	
FAULT	INTEGER	Код ошибки: 0 – нет ошибок; 1 – ошибка размера массива; 2 – недопустимая ссылка на массив.

Назначение

Линейная одномерная интерполяция.

7.3.77 L_INTERPL_2



Входы:

X1	REAL	Фактический аргумент 1
masX1	#REAL	Массив аргументов 1 (элементы типа REAL)
sizeX1	INTEGER	Размер массива X1
X2	REAL	Фактический аргумент 2
masX2	#REAL	Массив аргументов 2 (элементы типа REAL)
sizeX2	INTEGER	Размер массива X2

masY	#REAL	Массив функций от аргументов X1 и X2 (элементы типа REAL)

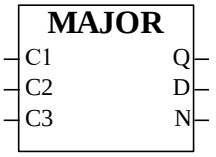
Выход:

Y	REAL	
FAULT	INTEGER	Код ошибки: 0 – нет ошибок; 1 – ошибка размера массива; 2 – недопустимая ссылка на массив.

Назначение

Линейная двумерная интерполяция.

7.3.78 MAJOR



Входы:

C1	BOOLEAN	1-й вход
C2	BOOLEAN	2-й вход
C3	BOOLEAN	3-й вход

Выходы:

Q	BOOLEAN	Основной выход
D	BOOLEAN	Признак несовпадения
N	INTEGER	Номер несовпадающего входа

Назначение

Функциональный блок MAJOR используется для повышения достоверности дискретных сигналов, поступающих, например, от модулей дискретного ввода контроллера. Алгоритм работает по правилу "два из трех".

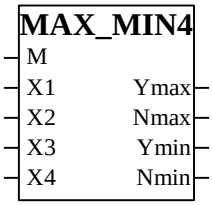
Описание алгоритма

Алгоритм содержит узел мажорирования, а также логику, определяющую номер сигнала, состояние которого не совпадает с состоянием двух других сигналов.
Работа алгоритма описывается следующей таблицей.

Состояние входных сигналов	Q	D	N
C1 = C2 = C3 = C	C	0	0
C2 = C3 = C; C1 != C	C	1	1
C1 = C3 = C; C2 != C	C	1	2
C1 = C2 = C; C3 != C	C	1	3

Выход D устанавливается в 1, если состояние одного из входных сигналов не совпадает с состоянием двух других сигналов. На выходе N формируется номер несовпадающего входного сигнала.

7.3.79 MAX_MIN4



Входы:

M	INTEGER	Количество используемых входов от 1 до 4
X1	REAL	1-й вход
X2	REAL	2-й вход
X3	REAL	3-й вход
X4	REAL	4-й вход

Выходы:

Ymax	REAL	Максимум из входов
Nmax	INTEGER	Номер входа с максимальным значением
Ymin	REAL	Минимум из входов
Nmin	INTEGER	Номер входа с минимальным значением

Назначение.

Функция MAX_MIN4 используется для выделения максимального и минимального из нескольких (до 4) сигналов.

Описание алгоритма.

На вход алгоритма поступают сигналы, число которых 0<M<4 и задается на входе M. Выходной сигнал Ymax равен максимальному из этих сигналов, выходной сигнал Ymin равен минимальному из этих сигналов:

$$Ymax=min\ \{X1;X2;...XM\};$$
$$Ymin=min\ \{X1;X2;...XM\}$$

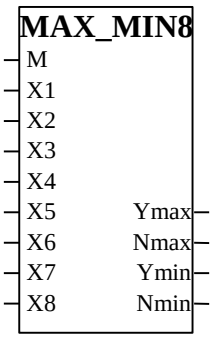
При этом входные сигналы, номер которых больше M не учитываются.

Алгоритм имеет дополнительные выходы Nmax и Nmin. На выходе Nmax формируется число, равное номеру входного сигнала, прошедшего на выход Ymax (т.е. являющегося максимальным). На выходе Nmin формируется число, равное номеру входного сигнала, прошедшего на выход Ymin (т.е. являющегося минимальным). Если имеется группа равных между собой сигналов и эти сигналы являются максимальными, то номер Nmax равен минимальному номеру сигналов в этой группе. Если имеется группа равных между собой сигналов и эти сигналы являются минимальными, то номер Nmin равен минимальному номеру сигналов в этой группе.

При M<1 все выходы алгоритма равны нулю. Значение M>4 алгоритм воспринимает как M=4.

Примечание: На неиспользуемые входы можно подавать любые значения, т.к. эти значения не будут учитываться при определении максимума и минимума.

7.3.80 MAX_MIN8



Входы:

M	INTEGER	Количество используемых входов от 1 до 8
X1	REAL	1-й вход

X2	REAL	2-й вход
X3	REAL	3-й вход
X4	REAL	4-й вход
X5	REAL	5-й вход
X6	REAL	6-й вход
X7	REAL	7-й вход
X8	REAL	8-й вход

Выходы:

Ymax	REAL	Максимум из входов
Nmax	INTEGER	Номер входа с максимальным значением
Ymin	REAL	Минимум из входов
Nmin	INTEGER	Номер входа с минимальным значением

Назначение.

Функция MAX_MIN8 используется для выделения максимального и минимального из нескольких (до 8) сигналов.

Описание алгоритма.

На вход алгоритма поступают сигналы, число которых 0<M<8 и задается на входе M. Выходной сигнал Ymax равен максимальному из этих сигналов, выходной сигнал Ymin равен минимальному из этих сигналов:

$$Ymax=min\ \{X1;X2;...XM\};$$
$$Ymin=min\ \{X1;X2;...XM\}$$

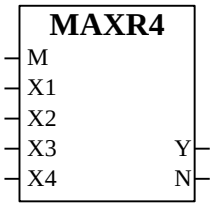
При этом входные сигналы, номер которых больше M не учитываются.

Алгоритм имеет дополнительные выходы Nmax и Nmin. На выходе Nmax формируется число, равное номеру входного сигнала, прошедшего на выход Ymax (т.е. являющегося максимальным). На выходе Nmin формируется число, равное номеру входного сигнала, прошедшего на выход Ymin (т.е. являющегося минимальным). Если имеется группа равных между собой сигналов и эти сигналы являются максимальными, то номер Nmax равен минимальному номеру сигналов в этой группе. Если имеется группа равных между собой сигналов и эти сигналы являются минимальными, то номер Nmin равен минимальному номеру сигналов в этой группе.

При M<1 все выходы алгоритма равны нулю. Значение M>8 алгоритм воспринимает как M=8.

Примечание: На неиспользуемые входы можно подавать любые значения, т.к. эти значения не будут учитываться при определении максимума и минимума.

7.3.81 MAXR4



Входы:

M	INTEGER	Количество используемых входов от 1 до 4
X1	REAL	1-й вход
X2	REAL	2-й вход
X3	REAL	3-й вход
X4	REAL	4-й вход

Выходы:

Y	REAL	Максимум из входов
N	INTEGER	Номер входа с максимальным значением

Назначение.

Функция MAXR4 используется для выделения максимального из нескольких (до 4) сигналов.

Описание алгоритма.

На вход алгоритма поступают сигналы, число которых 0<M<4 и задается на входе M. Выходной сигнал равен максимальному из этих сигналов:

Y=max {X1;X2;...XM}

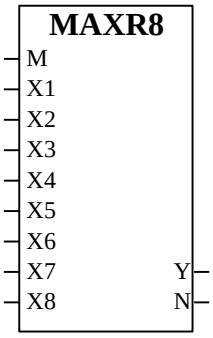
При этом входные сигналы, номер которых больше M не учитываются.

Алгоритм имеет дополнительный выход N, на котором формируется число, равное номеру входного сигнала, прошедшего на выход (т.е. являющегося максимальным). Если имеется группа равных между собой сигналов, причем эти сигналы являются максимальными, то номер N равен минимальному номеру сигналов в этой группе.

При M<1 выходы Y и N равны нулю. Значение M>4 алгоритм воспринимает как M=4.

Примечание: На неиспользуемые входы можно подавать любые значения, т.к. эти значения не будут учитываться при определении максимума.

7.3.82 MAXR8



Входы:

M	INTEGER	Количество используемых входов от 1 до 8
X1	REAL	1-й вход
X2	REAL	2-й вход
X3	REAL	3-й вход
X4	REAL	4-й вход
X5	REAL	5-й вход
X6	REAL	6-й вход
X7	REAL	7-й вход
X8	REAL	8-ой вход

Выходы:

Y	REAL	Максимум из входов
N	INTEGER	Номер входа с максимальным значением

Назначение.

Функция MAXR8 используется для выделения максимального из нескольких (до 8) сигналов.

Описание алгоритма.

На вход алгоритма поступают сигналы, число которых 0<M<8 и задается на входе M. Выходной сигнал равен максимальному из этих сигналов:

Y=max {X1;X2;...XM}

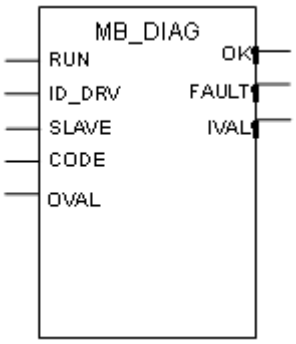
При этом входные сигналы, номер которых больше M не учитываются.

Алгоритм имеет дополнительный выход N, на котором формируется число, равное номеру входного сигнала, прошедшего на выход (т.е. являющегося максимальным). Если имеется группа равных между собой сигналов, причем эти сигналы являются максимальными, то номер N равен минимальному номеру сигналов в этой группе.

При M<1 выходы Y и N равны нулю. Значение M>8 алгоритм воспринимает как M=8.

Примечание: На неиспользуемые входы можно подавать любые значения, т.к. эти значения не будут учитываться при определении максимума.

7.3.83 MB_DIAG



Входы:

RUN	BOOLEAN	Запуск функционального блока
ID_DRV	INTEGER	Идентификатор устройства Modbus. Допустимые значения входа ID_DRV приведены в документе TREI_MODBUS.pdf.
SLAVE	INTEGER	Адрес устройства для режима Slave
CODE	INTEGER	Диагностический код
OVAL	INTEGER	Отправляемые данные

Выходы:

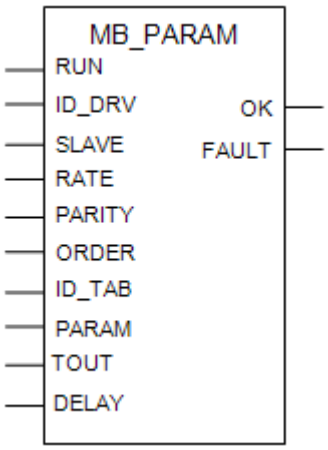
OK	BOOLEAN	Признак завершения операции
FAULT	INTEGER	Код завершения операции
IVAL	INTEGER	Принятые данные

Назначение:

Проверяется система коммуникации с подчиненным устройством. Используется функция Modbus с кодом 8 (Loopback Test). Возможные значения диагностического кода определяются документацией на конкретное устройство. В общем случае, при нулевом значении диагностического кода подчиненное устройство возвращает значение IVAL, отправленное через вход OVAL.

Возможные значения выхода Fault (код завершения операции) приведены в описании ФБ MB_PARAM.

7.3.84 MB_PARAM



Входы:

RUN	BOOLEAN	Запуск функционального блока
ID_DRV	INTEGER	Идентификатор устройства Modbus

SLAVE	INTEGER	Адрес устройства для режима Slave
RATE	INTEGER	Скорость передачи по последовательной линии
PARITY	INTEGER	Количество стоповых бит и режим контроля четности
ORDER	INTEGER	Установка режимов адресации и замены байт
ID_TAB	INTEGER	Идентификатор таблицы смещения
PARAM	INTEGER	Режим работы
TOUT	INTEGER	Время ожидания ответа от Slave-устройства
DELAY	INTEGER	Длительность паузы между передачами

Выходы:

OK	BOOLEAN	Признак завершения асинхронной операции
FAULT	INTEGER	Код ошибки

Назначение:
Конфигурация связи по протоколу Modbus.

Описание:

Подробнее об особенностях реализации MB_PARAM для различных модулей см. документ "Unimod Pro. Протокол Modbus".

Изменение установок Modbus происходит по фронту входа **RUN** (для мастер-модулей ***M501E/M841E/M902E/M903E/M915E/M921E*** – при единичном значении). Новые установки Modbus вступают в силу незамедлительно. Текущий приём или передача посылки будет прервана, что может привести к обнаружению ошибок передачи мастер-контроллером.

Вход **ID_DRV** задаёт идентификатор физической линии или устройства Modbus. Недопустимое значение на входе ID_DRV устанавливает на выходе FAULT код ошибки 139 – "неверный идентификатор устройства".

Аргументы **RATE** и **PARITY** используются для конфигурирования последовательной линии RS-485. Через вход RATE задается скорость обмена. RATE может принимать одно из следующих значений:

1200, 1800, 2400, 4800, 7200, 9600, 14400, 19200, 38400, 57600, 115200,
128000, 250000, 625000, 1250000, 2500000, бод.

Если заданная скорость не поддерживается модулем, функциональный блок возвращает ошибку 142 – "некорректное значение скорости передачи".

Количество стоповых битов и режим контроля четности задается входом PARITY. Возможные варианты режима контроля четности представлены в таблице 1.

Таблица 1 – Режим контроля четности

Значение	Количество стоповых бит	Контроль четности
0	1	Отключен
1	2	Отключен
2	1	EVEN
3	1	ODD
4	1	SPACE
5	1	MARK

Значение входа **RATE** используется также для задания номера IP порта в режиме *ModbusTCP Slave* на мастер-модулях **M1011E(E2).**

Значение входа **ORDER** используется только в режиме Slave и устанавливает режим адресации и формат передачи для целых и вещественных переменных и должно выбираться в зависимости от текущей конфигурации мастер-контроллера Modbus (при инициализации работы в режиме Master, **ORDER** задается отдельно для тех функциональных блоков, где этот параметр имеет значение). Возможные значения и соответствующие им режимы работы представлены в таблице 2.

Таблица 2 – Режимы адресации и замены байт

Значение	Ширина переменной	Порядок байт	Замена байт	Замена слов	Без знаковый обмен	Контроль доступа
0	32 бита	1-0-3-2	Да	Нет		Нет
1	32 бита	0-1-2-3	Нет	Нет		Нет
2	32 бита	3-2-1-0	Да	Да		Нет
3	32 бита	2-3-0-1	Нет	Да		Нет
4	16 бит	1-0	Да		Нет	Нет
5	16 бит	0-1	Нет		Нет	Нет
6	16 бит	1-0	Да		Да	Нет
7	16 бит	0-1	Нет		Да	Нет
8	32 бита	1-0-3-2	Да	Нет		Есть
9	32 бита	0-1-2-3	Нет	Нет		Есть
10	32 бита	3-2-1-0	Да	Да		Есть
11	32 бита	2-3-0-1	Нет	Да		Есть
12	16 бит	1-0	Да		Нет	Есть
13	16 бит	0-1	Нет		Нет	Есть
14	16 бит	1-0	Да		Да	Есть
15	16 бит	0-1	Нет		Да	Есть

Для конфигурирования 32-х разрядного режима работы используется установка порядка следования для 4-х байт. На 16-ти разрядный режим влияет опция замены байт и выбор знакового или без знакового режима обмена.

Формат аргумента **PARAM** имеет следующий вид:

MSB X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X ASC X HDUP SYNC **LSB**

Формат аргумента PARAM

SYNC (разряд 0, synchronous). Если флаг установлен, обмен со Slave-устройствами происходит в синхронном режиме: вызванный функциональный блок не возвращает управление технологической программе до тех пор, пока текущая операция Modbus не будет завершена. При сброшенном флаге SYNC функциональные блоки работают асинхронном режиме, а момент завершения операции определяется состоянием выходов READY. Для **мастер-модуля M911E** сброшенный флаг соответствует синхронному режиму!

HDUP (разряд 1, half-duplex). Флаг контролирует режим обмена по линии RS-485. Сброшенный флаг HDUP устанавливает полнодуплексный режим обмена по двум дифференциальным линиям, отдельно для приема и передачи (RS-422). Установка флага HDUP разрешает при отсутствии посылки Modbus перевод передатчика RS-485 в третье состояние; таким образом, становится возможна двунаправленный обмен пакетами по одной дифференциальной линии – режим half-duplex.

ASC (разряд 3, ascii). (мастер-модули ***M841E/M902E/M921E/M915E***) Если флаг установлен, то обмен с удаленными устройствами ведется в режиме ASCII.

Параметр **ID_TAB** в режиме ***Slave*** задает идентификатор ресурса, в котором хранится таблица трансляции адресов Modbus. Если ресурс с заданным номером не найден, или его тип не совпадает с запрошенным значением, функциональный блок MB_PARAM возвращает код ошибки 140 – “неверный ресурс”.

Опционально, в старших 16 разрядах номера ресурса может быть указан тип запрашиваемого ресурса (рисунок 1).

Описание структуры таблицы трансляции адресов в зависимости от типа модуля приведено в документе “Unimod PRO. Протокол Modbus.”.

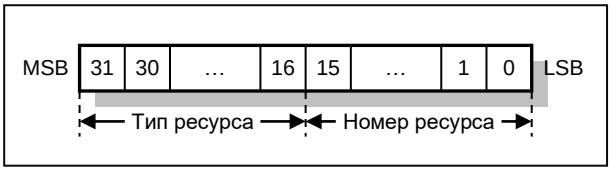


Рисунок 1 - Формат номера ресурса

Перед тем как использовать переданную таблицу трансляции адресов Modbus, функциональный блок производит проверку всех ее полей. При обнаружении каких-либо ошибок выхода за текущий диапазон адресов словаря обмена, функциональный блок

возвращает ошибку с кодом 140 – “неверный ресурс”. Если номер ресурса равен нулю, таблица трансляции адресов не используется, а каждой из групп памяти Modbus в соответствии с ее типом выделяется общее доступное адресное пространство словаря обмена.

Параметр **ID_TAB** в режиме ***Modbus-TCP Master*** (мастер-модули ***M841E/M921E/M902E/M915E*** и модуль ***M932C2***) задает идентификатор ресурса, в котором хранится IP-адрес и порт Slave-устройства (тип ресурса “Параметры Modbus-TCP Master”):

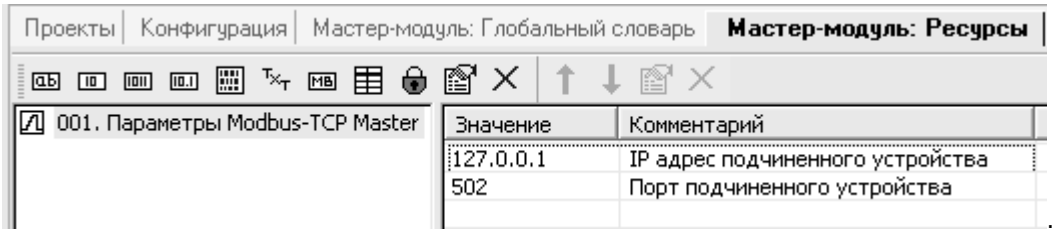


Рисунок 2 - Параметры Modbus-TCP Master

В режиме Master параметры **TOUT** и **DELAY** задают время ожидания ответа от Slave-устройства и задержку после получения ответа и перед следующей передачей соответственно. Для режима Slave параметры TOUT и DELAY задают время обнаружение обрыва (окончания) пакета и задержку перед отправкой ответа. Все значения задаются в миллисекундах.

После завершения операции, выход функционального блока **OK** устанавливается в состояние TRUE, а на выходе **FAULT** – код ошибки или нуль, если вызов отработан без ошибок.

Приведенные коды ошибок актуальны для всех блоков MB_*.

Таблица 6 - Значения выхода FAULT (для M401E/M501E/M841E/M902E/M903E/M915E/M921E)

Код ошибки	Описание
0	Выполнено успешно
1	Ошибка инициализации (некорректные параметры)
2	Удаленное устройство не отвечает (таймаут истек)
3	Системная ошибка при работе с портом / сервер Modbus-TCP недоступен
4	Системная ошибка при обращении к задаче связи
5	Выполняется запрос
6	Ошибка принятых данных
7	Выполнение блока прервано (актуально в асинхронном режиме)
9	Ошибка открытия порта
13	Некорректный ответ от slave-устройства (Адрес устройства/Код функции/Количество данных в ответном пакете отличается от запрашиваемого)
138	Служба/драйвер не был инициализирован (выключен запуск задачи связи или для данного ID_DRV не вызван ФБ MB_PARAM)
141	Параметр Slave задан некорректно
144	Параметр Order задан некорректно (например, для вещественной переменной указан 16-разрядный режим)
149	Неизвестный код функции или функция не поддерживается (параметр FUNC задан неверно)
150	Параметр Number задан некорректно (к этой ошибке также относится обращение в режиме Master по несуществующему адресу на стороне мастер-модуля (связка параметров Addrb и Number))
151	Параметр Type задан некорректно. К этой ошибке также относится: • несоответствие разрядности переменных (задан 32-разрядный тип, но переменные в словаре технологического приложения расположены подряд (подробнее см. документ "TREI_MODBUS.pdf", раздел "Особенности реализации MB_PARAM", пункт "MB_PARAM на мастер-модуле"), и наоборот) • обращение отдельно к старшей части 32-разрядной

	переменной
152	Внутренняя ошибка при обращении к базе технологического приложения
201-211	Пришел ответ исключения от Slave устройства (подробнее см. описание протокола Modbus или документацию на Slave-устройство)
201	Illegal function
202	Illegal data address
203	Illegal data value
204	Server device failure
205	Acknowledge
206	Server device busy
208	Memory parity error
210	Gateway path unavailable
211	Gateway target device failed to respond

Таблица 5 - Значения выхода FAULT (кроме **M401E/M501E/*M841E/M902E/M903E/M915E/M921E***)

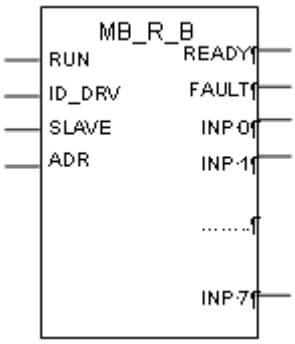
Код ошибки	Описание
0	Выполнено успешно
1	Пришел ответ исключения
2	Удаленное устройство не отвечает (таймаут истек)
3	Неисправность на линии (ошибки паритета, фрейма)
4	Внутренняя ошибка на плате модуля (возможно на модуле M911, если коммуникационный адаптер не отвечает)
5	Нет ответа на широковещательное сообщение
6	Был принят неполный пакет
10	Переполнение аппаратного буфера
11	Ошибка паритета
12	Несовпадение контрольной суммы пакета
13	Некорректный ответ от slave–устройства
14	Неверная длинна пакета от slave–устройства
138	Служба/драйвер не был инициализирован
139	Неверный код устройства
140	Неверный формат ресурса или ресурс не найден
141	Неверный адрес slave-устройства
142	Заданная скорость передачи не поддерживается
143	Заданный режим контроля четности (parity mode) не поддерживается
144	Неверно задан аргумент “ORDER” в MB_PARAM или заданный режим не поддерживается
145	Неверно задан аргумент “PARAM” в MB_PARAM или заданный режим не поддерживается
146	Неверно задан аргумент “TOUT” в MB_PARAM
147	Неверно задан аргумент “DELAY” в MB_PARAM
148	Некорректный стартовый адрес
149	Неизвестный код функции или функция не поддерживается

В реализации MB-блоков на всех интеллектуальных модулях, добавлена следующая возможность: при получении правильного ответа-исключения от slave-устройства, соответствующий MB-блок устанавливает на своем выходе FAULT сумму кодов.

Сумма кодов состоит из базового кода D1000 – определяющего, что получен ответ-исключение. И собственно кода исключения в соответствии со спецификацией MODBUS.

Например, код D1002, означает – получен ответ исключение с кодом 2(ILLEGAL DATA ADDRESS).

7.3.85 MB_R_B



Входы:

RUN	BOOLEAN	Запуск функционального блока
ID_DRV	INTEGER	Идентификатор устройства Modbus. Допустимые значения входа ID_DRV приведены в документе TREI_MODBUS.pdf.
SLAVE	INTEGER	Адрес устройства для режима Slave
ADR	INTEGER	Адрес первого входа

Выходы:

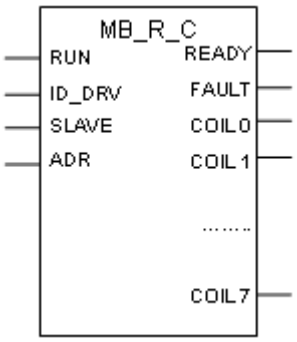
READY	BOOLEAN	Признак завершения операции
FAULT	INTEGER	Код завершения операции
INP0	BOOLEAN	Состояние входа по адресу ADR+0
INP1	BOOLEAN	Состояние входа по адресу ADR+1
INP2	BOOLEAN	Состояние входа по адресу ADR+2
INP3	BOOLEAN	Состояние входа по адресу ADR+3
INP4	BOOLEAN	Состояние входа по адресу ADR+4
INP5	BOOLEAN	Состояние входа по адресу ADR+5
INP6	BOOLEAN	Состояние входа по адресу ADR+6
INP7	BOOLEAN	Состояние входа по адресу ADR+7

Назначение:

Чтение состояния 8-ми последовательных дискретных входов. Используется функция Modbus с кодом 2 (Read Input Status).

Возможные значения выхода Fault (код завершения операции) приведены в описании ФБ MB_PARAM.\

7.3.86 MB_R_C



Входы:

RUN	BOOLEAN	Запуск функционального блока
ID_DRV	INTEGER	Идентификатор устройства Modbus. Допустимые значения входа ID_DRV приведены в документе TREI_MODBUS.pdf.
SLAVE	INTEGER	Адрес подчиненного устройства
ADR	INTEGER	Адрес первой ячейки

Выходы:

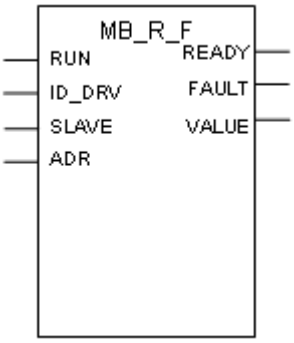
READY	BOOLEAN	Признак завершения операции
FAULT	INTEGER	Код завершения операции
COIL0	BOOLEAN	Состояние ячейки по адресу Adr+0
COIL1	BOOLEAN	Состояние ячейки по адресу Adr+1
COIL2	BOOLEAN	Состояние ячейки по адресу Adr+2
COIL3	BOOLEAN	Состояние ячейки по адресу Adr+3
COIL4	BOOLEAN	Состояние ячейки по адресу Adr+4
COIL5	BOOLEAN	Состояние ячейки по адресу Adr+5
COIL6	BOOLEAN	Состояние ячейки по адресу Adr+6
COIL7	BOOLEAN	Состояние ячейки по адресу Adr+7

Назначение:

Чтение состояния 8-ми последовательных бинарных ячеек памяти. Используется функция Modbus с кодом 1 (Read Coil Status).

Возможные значения выхода Fault (код завершения операции) приведены в описании ФБ MB_PARAM.

7.3.87 MB_R_F



Входы:

RUN	BOOLEAN	Запуск функционального блока
ID_DRV	INTEGER	Идентификатор устройства Modbus. Допустимые значения входа ID_DRV приведены в документе TREI_MODBUS.pdf.
SLAVE	INTEGER	Адрес устройства для режима Slave
ADR	INTEGER	Адрес регистра

Выходы:

READY	BOOLEAN	Признак завершения операции
FAULT	INTEGER	Код завершения операции
VALUE	REAL	Значение пары регистров в формате с плавающей точкой

Назначение:

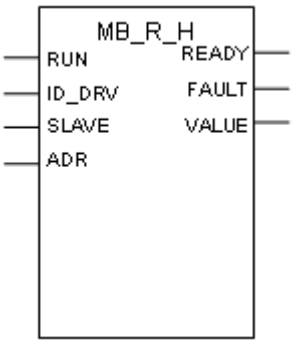
Чтение пары последовательных регистров. Значение возвращается в формате числа с плавающей точкой. Используется функция Modbus с кодом 3 (Read Holding Registers).

В параметре SLAVE в качестве старшего байта можно добавить порядок чтения байт ORDER. Значение которого определяется в соответствии с таблицей 1. Например:

```
slave:= 16#0301;          (*режим: 32х разрядный с заменой байт и заменой слов,  адрес: 1  *)
```

Возможные значения выхода Fault (код завершения операции) приведены в описании ФБ MB_PARAM.

7.3.88 MB_R_H



Входы:

RUN	BOOLEAN	Запуск функционального блока
ID_DRV	INTEGER	Идентификатор устройства Modbus. Допустимые значения входа ID_DRV приведены в документе TREI_MODBUS.pdf.
SLAVE	INTEGER	Адрес устройства для режима Slave
ADR	INTEGER	Адрес регистра

Выходы:

READY	BOOLEAN	Признак завершения операции
FAULT	INTEGER	Код завершения операции
VALUE	INTEGER	Значение целочисленного регистра

Назначение:

Чтение целочисленного регистра. Используется функция Modbus с кодом 3 (Read Holding Registers).

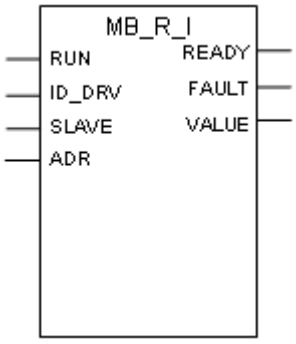
В параметре SLAVE в качестве старшего байта можно добавить порядок чтения байт ORDER. Значение которого определяется в соответствии с таблицей 1. Например:

```
slave:=1;          (*режим: 32х разрядный с заменой байт,  адрес: 1  *)
slave:=16#0601;  (*режим: 16х разрядный с  заменой байт беззнаковый,   адрес: 1  *)
```

При установке 32х разрядного режима производится последовательное чтение 2-х регистров начиная с адреса ADR.

Возможные значения выхода Fault (код завершения операции) приведены в описании ФБ MB_PARAM.

7.3.89 MB_R_I



Входы:

RUN	BOOLEAN	Запуск функционального блока
ID_DRV	INTEGER	Идентификатор устройства Modbus. Допустимые значения входа ID_DRV приведены в документе TREI_MODBUS.pdf.
SLAVE	INTEGER	Адрес устройства для режима Slave
ADR	INTEGER	Адрес регистра

Выходы:

READY	BOOLEAN	Признак завершения операции
FAULT	INTEGER	Код завершения операции
VALUE	INTEGER	Значение целочисленного регистра

Назначение:

Чтение целочисленного регистра. Используется функция Modbus с кодом 4 (Read Input Registers).

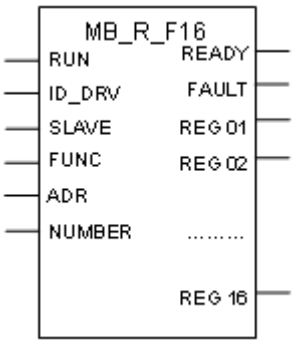
В параметре Slave в качестве старшего байта можно добавить порядок чтения байт ORDER. Значение которого определяется в соответствии с таблицей 1. Например:

```
slave:=1;          (*режим: 32х разрядный с заменой байт,  адрес: 1  *)
slave:=16#0601;  (*режим: 16х разрядный с  заменой байт беззнаковый,   адрес: 1  *)
```

При установке 32х разрядного режима производится последовательное чтение 2-х регистров начиная с адреса ADR.

Возможные значения выхода Fault (код завершения операции) приведены в описании ФБ MB_PARAM.

7.3.90 MB_R_F16



Входы:

RUN	BOOLEAN	Запуск функционального блока
ID_DRV	INTEGER	Идентификатор устройства Modbus. Допустимые значения входа ID_DRV приведены в документе TREI_MODBUS.pdf.
SLAVE	INTEGER	Адрес устройства для режима Slave
FUNC	INTEGER	Код Modbus функции (3/4)
ADR	INTEGER	Начальный адрес группы регистров
NUMBER	INTEGER	Количество пар регистров

Выходы:

READY	BOOLEAN	Признак завершения операции
FAULT	INTEGER	Код завершения операции
REG01	INTEGER	Значение пары регистров с адресом ADR+0
REG02	INTEGER	Значение пары регистров с адресом ADR+1
….		
REG16	INTEGER	Значение пары регистров с адресом ADR+15

Назначение:

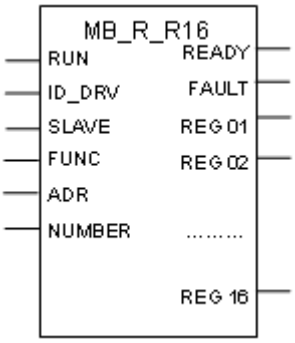
Чтение 16-ти пар регистров подряд начиная с адреса ADR. Вход Func задает код функции Modbus для запроса данных от устройства. Допускаются следующие функции: 3 (Read Holding Registers) и 4 (Read Input Registers). Значение пары регистров представляется в формате с плавающей точкой.

В параметре SLAVE в качестве старшего байта можно добавить порядок чтения байт ORDER. Значение которого определяется в соответствии с таблицей 1. Например:

```
slave:= 16#0301;          (*режим: 32х разрядный с заменой байт и заменой слов,  адрес: 1  *)
```

Возможные значения выхода Fault (код завершения операции) приведены в описании ФБ MB_PARAM.

7.3.91 MB_R_R16



Входы:

RUN	BOOLEAN	Запуск функционального блока
ID_DRV	INTEGER	Идентификатор устройства Modbus. Допустимые значения входа ID_DRV приведены в документе TREI_MODBUS.pdf.
SLAVE	INTEGER	Адрес устройства для режима Slave
FUNC	INTEGER	Код Modbus функции (3/4)
ADR	INTEGER	Адрес регистра
NUMBER	INTEGER	Количество регистров

Выходы:

READY	BOOLEAN	Признак завершения операции
FAULT	INTEGER	Код завершения операции
REG01	INTEGER	Значение целочисленного регистра с адресом Adr+0
REG02	INTEGER	Значение целочисленного регистра с адресом Adr+1
….		
REG16	INTEGER	Значение целочисленного регистра с адресом Adr+15

Назначение:

Чтение 16-ти целочисленных регистров подряд начиная с адреса ADR. Вход Func задает код функции Modbus для запроса данных от устройства. Допускаются следующие функции: 3 (Read Holding Registers) и 4 (Read Input Registers).

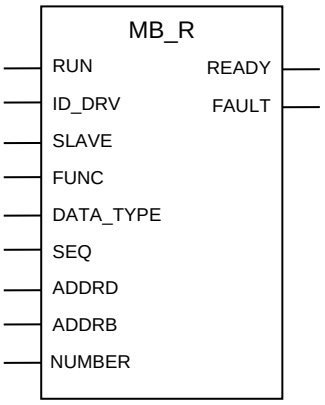
В параметре Slave в качестве старшего байта можно добавить порядок чтения байт Order. Значение которого определяется в соответствии с таблицей 1. Например:

```
slave:=1;          (*режим: 32х разрядный с  заменой байт,  адрес: 1  *)
slave:=16#0601;  (*режим: 16х разрядный с  заменой байт беззнаковый,   адрес: 1  *)
```

При установке 32х разрядного режима производится последовательное чтение 2-х регистров начиная с адреса ADR.

Возможные значения выхода Fault (код завершения операции) приведены в описании ФБ MB_PARAM.

7.3.92 MB_R



Входы:

RUN	BOOLEAN	Запуск функционального блока
ID_DRV	INTEGER	Идентификатор устройства Modbus. Допустимые значения входа ID_DRV приведены в документе TREI_MODBUS.pdf.
SLAVE	INTEGER	Адрес подчиненного устройства
FUNC	INTEGER	Код Modbus функции: 1,2,3,4
DATA_TYPE	INTEGER	Тип данных: 0 булевское 1 2х байтовое целое без знака 2 2х байтовое целое со знаком 3 4х байтовое целое без знака 4 4х байтовое целое со знаком 5 4х байтовое вещественное 6 8и байтовое вещественное
SEQ	INTEGER	Порядок байтов: 0 По умолчанию 1 “1-0” – 2х байтовое, замена байт (типовое значение) 2 “0-1” – 2х байтовое 3 “1-0-3-2” – 4х байтовое, замена байт (типовое значение) 4 “0-1-2-3” – 4х байтовое 5 “2-3-0-1” – 4х байтовое, замена слов 6 “3-2-1-0” – 4х байтовое, замена байт, замена слов 7 "1-0-3-2-5-4-7-6" – 8и байтовое вещественное, замена байт (типовое значение) 8 "0-1-2-3-4-5-6-7" – 8и байтовое вещественное 9 "2-3-0-1-6-7-4-5" – 8и байтовое вещественное, замена слов 10 "3-2-1-0-7-6-5-4" – 8и байтовое вещественное, замена байт, замена слов 11 "5-4-7-6-1-0-3-2" – 8и байтовое вещественное, замена двойных слов, замена байт 12 "4-5-6-7-0-1-2-3" – 8и байтовое вещественное, замена двойных слов 13 "6-7-4-5-2-3-0-1" – 8и байтовое вещественное, замена двойных слов, замена слов 14 "7-6-5-4-3-2-1-0" – 8и байтовое вещественное, замена двойных слов, замена байт, замена слов
ADDRD	INTEGER	Начальный адрес данных на подчиненном устройстве
ADDRB	INTEGER	Начальный адрес данных в базе
NUMBER	INTEGER	Количество

Выходы:

READY	BOOLEAN	Признак завершения операции
FAULT	INTEGER	Код завершения операции

Назначение:
Чтение группы данных с подчиненного устройства непосредственно в базу UnimodPRO. Используются функции Modbus с кодами 1,2,3,4.

Особенности работы блока на мастер-модулях M841E/M902E/M921E/M915E/M903E:

После получения ответного пакета от подчиненного устройства, функциональный блок выполняет проверку соответствия типов переменных базы (начальный адрес которых задается входом ADDRB) параметру ***DATA_TYPE***:

Если DATA_TYPE=0, расположение Modbus-адресов не проверяется;
Если DATA_TYPE=1..2, Modbus-адреса переменных в технологическом приложении UnimodPRO должны идти подряд;
Если DATA_TYPE=3..5, Modbus-адреса должны идти “через один”
Если DATA_TYPE=6, Modbus-адреса должны идти “через три”

Аналогичным образом переменные располагаются, если мастер-модуль находится в режиме Modbus Slave:

Адрес	Переменная	Тип
0001	var_modbus1	Целая
0002	var_modbus2	Целая
0003	var_modbus3	Целая
0004	var_modbus4	Целая
0005	var_modbus5	Целая
0006	var_end	Целая
0007		
0008		
0009		
000A		

DATA_TYPE=1..2

Адрес	Переменная	Тип
0001	var_modbus1	Целая
0002		
0003	var_modbus2	Целая
0004		
0005	var_modbus3	Целая
0006		
0007	var_modbus4	Целая
0008		
0009	var_modbus5	Целая
000A		

DATA_TYPE=3..4

Адрес	Переменная	Тип
0001	var_modbus1	Вещественная
0002		
0003	var_modbus2	Вещественная
0004		
0005	var_modbus3	Вещественная
0006		
0007	var_modbus4	Вещественная
0008		
0009	var_modbus5	Вещественная
000A		
000B		
000C		
000D		
000E		
000F		
0010		
0011		
0012		

DATA_TYPE=5

Адрес	Переменная	Тип
0001	var_modbus1	Вещ. дв. точности
0002		
0003		
0004		
0005	var_modbus2	Вещ. дв. точности
0006		
0007		
0008		
0009	var_modbus3	Вещ. дв. точности
000A		
000B		
000C		
000D	var_modbus4	Вещ. дв. точности
000E		
000F		
0010		
0011	var_modbus5	Вещ. дв. точности
0012		

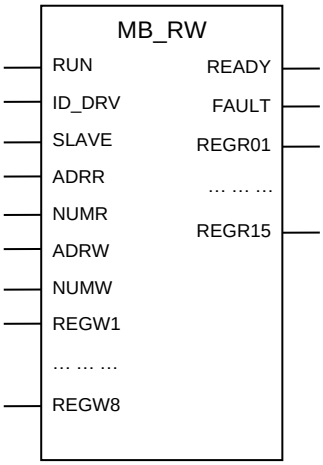
DATA_TYPE=6

На рисунках приведены примеры расположения переменных в базе UnimodPRO при чтении 5 переменных с подчиненного устройства.
Также следует обратить внимание, что в 2х байтовом режиме список переменных оканчивается переменной *var_end*, так как без нее переменная *var_modbus5* будет считаться 4х байтовой, и функциональный блок будет возвращать ошибку.

Параметр ***ADDRB*** должен иметь значение на ***единицу меньше*** адреса переменной на карте адресов Modbus, т.е. для обращения к переменной var_modbus1 параметр ADDRB должен иметь нулевое значение.

Возможные значения выхода Fault (код завершения операции) приведены в описании ФБ MB_PARAM.

7.3.93 MB_RW



Входы:

RUN	BOOLEAN	Запуск функционального блока
ID_DRV	INTEGER	Идентификатор устройства Modbus. Допустимые значения входа ID_DRV приведены в документе TREI_MODBUS.pdf.
SLAVE	INTEGER	Адрес подчиненного устройства
ADRR	INTEGER	Адрес входного регистра
NUMR	INTEGER	Количество входных регистров
ADRW	INTEGER	Адрес выходного регистра
NUMW	INTEGER	Количество выходных регистров
REGW1	INTEGER	Значение целочисленного регистра с адресом ADRW+0
. . .		
REGW8	INTEGER	Значение целочисленного регистра с адресом ADRW+7

Выходы:

READY	BOOLEAN	Признак завершения операции
FAULT	INTEGER	Код завершения операции
REGR01	INTEGER	Значение целочисленного регистра с адресом ADRR+0
. . .		
REGR15	INTEGER	Значение целочисленного регистра с адресом ADRR+14

Назначение:

Запись группы регистров с адреса ADRW. Чтение группы регистров с адреса ADRR. Используется функция Modbus с кодом 23 (Read/Write Multiple Registers).

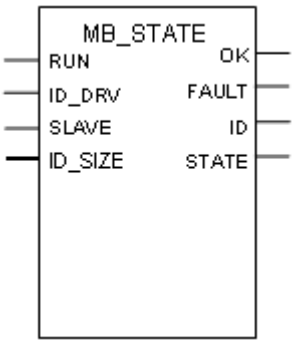
В параметре SLAVE в качестве старшего байта можно добавить порядок чтения байт ORDER. Значение которого определяется в соответствии с таблицей 1. Например:

```
slave:=1;          (*режим: 32х разрядный с заменой байт,  адрес: 1  *)
slave:=16#0601;  (*режим: 16х разрядный с  заменой байт беззнаковый,   адрес: 1  *)
```

При установке 32х разрядного режима производится последовательное чтение 2-х регистров начиная с адреса ADR.

Возможные значения выхода Fault (код завершения операции) приведены в описании ФБ MB_PARAM.

7.3.94 MB_STATE



Входы:

RUN	BOOLEAN	Запуск функционального блока
ID_DRV	INTEGER	Идентификатор устройства Modbus. Допустимые значения входа ID_DRV приведены в документе TREI_MODBUS.pdf.
SLAVE	INTEGER	Адрес устройства для режима Slave
ID_SIZE	INTEGER	Размер ID подчиненного в байтах

Выходы:

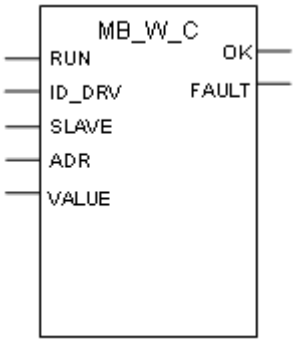
OK	BOOLEAN	Признак завершения операции
FAULT	INTEGER	Код завершения операции
ID	INTEGER	ID подчиненного устройства
STATE	INTEGER	Состояние подчиненного устройства

Назначение:

Получение ID подчиненного устройства и его рабочего состояния. Используется функция Modbus с кодом 17 (Report Slave ID). Возможные значения ID и рабочего состояния определяются документацией на конткретное устройство.

Возможные значения выхода Fault (код завершения операции) приведены в описании ФБ MB_PARAM.

7.3.95 MB_W_C



Входы:

RUN	BOOLEAN	Запуск функционального блока
ID_DRV	INTEGER	Идентификатор устройства Modbus. Допустимые значения входа ID_DRV приведены в документе TREI_MODBUS.pdf.
SLAVE	INTEGER	Адрес устройства для режима Slave
ADR	INTEGER	Начальный адрес группы регистров
VALUE	BOOLEAN	Состояние ячейки памяти

Выходы:

OK	BOOLEAN	Признак завершения операции

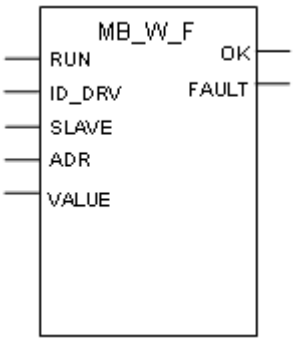
FAULT	INTEGER	Код завершения операции

Назначение:

Запись в бинарную ячейку памяти. Используется функция Modbus с кодом 5 (Force Single Coil).

Возможные значения выхода Fault (код завершения операции) приведены в описании ФБ MB_PARAM.

7.3.96 MB_W_F



Входы:

RUN	BOOLEAN	Запуск функционального блока
ID_DRV	INTEGER	Идентификатор устройства Modbus. Допустимые значения входа ID_DRV приведены в документе TREI_MODBUS.pdf.
SLAVE	INTEGER	Адрес устройства для режима Slave
ADR	INTEGER	Адрес регистра
VALUE	REAL	Значение пары регистров

Выходы:

OK	BOOLEAN	Признак завершения операции
FAULT	INTEGER	Код завершения операции

Назначение:

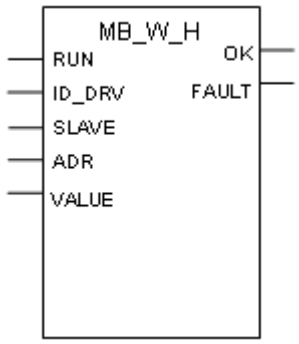
Запись пары последовательных регистров. Значение представляется в формате с плавающей точкой. Используется функция Modbus с кодом 16 (Preset Multiple Registers).

В параметре SLAVE в качестве старшего байта можно добавить порядок чтения байт ORDER. Значение которого определяется в соответствии с таблицей 1. Например:

```
slave:= 16#0301;        (*режим: 32х разрядный с заменой байт и заменой слов,  адрес: 1  *)
```

Возможные значения выхода Fault (код завершения операции) приведены в описании ФБ MB_PARAM.

7.3.97 MB_W_H



Входы:

RUN	BOOLEAN	Запуск функционального блока
ID_DRV	INTEGER	Идентификатор устройства Modbus. Допустимые значения входа ID_DRV приведены в документе TREI_MODBUS.pdf.
SLAVE	INTEGER	Адрес устройства для режима Slave
ADR	INTEGER	Адрес регистра
VALUE	BOOLEAN	Значение регистра

Выходы:

OK	BOOLEAN	Признак завершения операции
FAULT	INTEGER	Код завершения операции

Назначение:

Запись целочисленного регистра. Используется функция Modbus с кодом 6 (Preset Single Register).

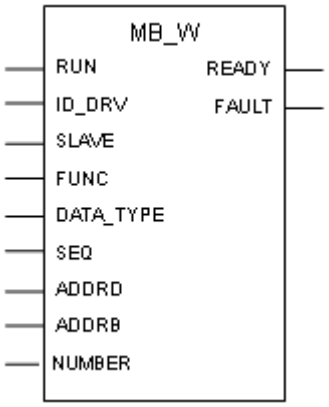
В параметре Slave в качестве старшего байта можно добавить порядок чтения байт ORDER. Значение которого определяется в соответствии с таблицей 1. Например:

```
slave:=1;          (*режим: 32х разрядный с  заменой байт,  адрес: 1  *)
slave:=16#0601;  (*режим: 16х разрядный с  заменой байт беззнаковый,   адрес: 1  *)
```

При установке 32х разрядного режима производится последовательная запись 2-х регистров начиная с адреса ADR. В этом случае используется функция Modbus с кодом 16 (Preset Multiple Registers).

Возможные значения выхода Fault (код завершения операции) приведены в описании ФБ MB_PARAM.

7.3.98 MB_W



Входы:

RUN	BOOLEAN	Запуск функционального блока

ID_DRV	INTEGER	Идентификатор устройства Modbus. Допустимые значения входа ID_DRV приведены в документе TREI_MODBUS.pdf.
SLAVE	INTEGER	Адрес подчиненного устройства
FUNC	INTEGER	Код Modbus функции: 15,16
DATA_TYPE	INTEGER	Тип данных: 0 булевское 1 2х байтовое целое без знака 2 2х байтовое целое со знаком 3 4х байтовое целое без знака 4 4х байтовое целое со знаком 5 4х байтовое вещественное 6 8и байтовое вещественное
SEQ	INTEGER	Порядок байтов: 0 По умолчанию 1 “1-0” – 2х байтовое, замена байт (типовое значение) 2 “0-1” – 2х байтовое 3 “1-0-3-2” – 4х байтовое, замена байт (типовое значение) 4 “0-1-2-3” – 4х байтовое 5 “2-3-0-1” – 4х байтовое, замена слов 6 “3-2-1-0” – 4х байтовое, замена байт, замена слов 7 "1-0-3-2-5-4-7-6" - 8и байтовое вещественное, замена байт (типовое значение) 8 "0-1-2-3-4-5-6-7" – 8и байтовое вещественное 9 "2-3-0-1-6-7-4-5" – 8и байтовое вещественное, замена слов 10 "3-2-1-0-7-6-5-4" – 8и байтовое вещественное, замена байт, замена слов 11 "5-4-7-6-1-0-3-2" – 8и байтовое вещественное, замена двойных слов, замена байт 12 "4-5-6-7-0-1-2-3" – 8и байтовое вещественное, замена двойных слов 13 "6-7-4-5-2-3-0-1" – 8и байтовое вещественное, замена двойных слов, замена слов 14 "7-6-5-4-3-2-1-0" – 8и байтовое вещественное, замена двойных слов, замена байт, замена слов
ADDRD	INTEGER	Начальный адрес данных на подчиненном устройстве
ADDRB	INTEGER	Начальный адрес данных в базе
NUMBER	INTEGER	Количество

Выходы:

READY	BOOLEAN	Признак завершения операции
FAULT	INTEGER	Код завершения операции

Назначение:

Запись группы данных на подчиненное устройство непосредственно из базы UnimodPRO. Используются функции Modbus с кодами 15,16.

Особенности работы блока на мастер-модулях M841E/M902E/M921E/M915E/M903E:

Перед формированием пакета на запись функциональный блок выполняет проверку соответствия типов переменных базы (начальный адрес которых задается входом ADDRB) параметру ***DATA_TYPE***:

Если DATA_TYPE=0, расположение Modbus-адресов не проверяется;

Если DATA_TYPE=1..2, Modbus-адреса переменных в технологическом приложении UnimodPRO должны идти подряд;

Если DATA_TYPE=3..5, Modbus-адреса должны идти “через один”

Если DATA_TYPE=6, Modbus-адреса должны идти “через три”

Аналогичным образом переменные располагаются, если мастер-модуль находится в режиме Modbus Slave:

Адрес	Переменная	Тип
0001	var_modbus1	Целая
0002	var_modbus2	Целая
0003	var_modbus3	Целая
0004	var_modbus4	Целая
0005	var_modbus5	Целая
0006	var_end	Целая
0007		
0008		
0009		
000A		

DATA_TYPE=1..2

Адрес	Переменная	Тип
0001	var_modbus1	Целая
0002		
0003	var_modbus2	Целая
0004		
0005	var_modbus3	Целая
0006		
0007	var_modbus4	Целая
0008		
0009	var_modbus5	Целая
000A		

DATA_TYPE=3..4

Адрес	Переменная	Тип
0001	var_modbus1	Вещественная
0002		
0003	var_modbus2	Вещественная
0004		
0005	var_modbus3	Вещественная
0006		
0007	var_modbus4	Вещественная
0008		
0009	var_modbus5	Вещественная
000A		
000B		
000C		
000D		
000E		
000F		
0010		
0011		
0012		

DATA_TYPE=5

Адрес	Переменная	Тип
0001	var_modbus1	Вещ. дв. точности
0002		
0003		
0004		
0005	var_modbus2	Вещ. дв. точности
0006		
0007		
0008		
0009	var_modbus3	Вещ. дв. точности
000A		
000B		
000C		
000D	var_modbus4	Вещ. дв. точности
000E		
000F		
0010		
0011	var_modbus5	Вещ. дв. точности
0012		

DATA_TYPE=6

На рисунках приведены примеры расположения переменных в базе UnimodPRO при записи 5 переменных на подчиненное устройство.
Также следует обратить внимание, что в 2х байтовом режиме список переменных оканчивается переменной *var_end*, так как без нее переменная *var_modbus5* будет считаться 4х байтовой, и функциональный блок будет возвращать ошибку.

Параметр ***ADDRB*** должен иметь значение на ***единицу меньше*** адреса переменной на карте адресов Modbus, т.е. для обращения к переменной var_modbus1 параметр ADDRB должен иметь нулевое значение.

Возможные значения выхода Fault (код завершения операции) приведены в описании ФБ MB_PARAM.

7.3.99 MBRUTTO

Входы:

mode	Integer	Режим работы линии (0-учет,1-резерв,2-ремонт)
TYPE_PR	Integer	Тип расходомера (1-массомер,2-турбинка)
kf_	Real	Кфактор расходомера
MF_	Real	Мфактор расходомера
freq_01_	Real	Частота расходомера
dflow_01_	Real	Разница импульсов расходомера
dens_01_	Real	Плотность в линии
temp_01_	Real	Температура в линии
press_01_	Real	Давление в линии
dens_	Real	Плотность в БИКе
dens15_	Real	Плотность при 15 град.С в БИКе

dens20_	Real	Плотность при 20 град.С в БИКе
temp_	Real	Температура в БИКе
press_	Real	Давление в БИКе
water_	Real	Влажность
mex_prim	Real	Мех. примеси
M_soli	Real	Мин. соли
M2_01_reset	Real	Масса накоплен. за предыд. часы
V2_01_reset	Real	Объем накоплен. за предыд. часы
V215_01_reset	Real	Объем при 15 град.С накоплен. за предыд. часы
V220_01_reset	Real	Объем при 20 град.С накоплен. за предыд. часы
Msm_01_reset	Real	Масса накоплен. за предыд. часы
Vsm_01_reset	Real	Объем накоплен. за предыд. часы
Vsm15_01_reset	Real	Объем при 15 град.С накоплен. за предыд. часы
Vsm20_01_reset	Real	Объем при 20 град.С накоплен. за предыд. часы
Md_01_reset	Real	Масса накоплен. за предыд. часы
Vd_01_reset	Real	Объем накоплен. за предыд. часы
Vd15_01_reset	Real	Объем при 15 град.С накоплен. за предыд. часы
Vd20_01_reset	Real	Объем при 20 град.С накоплен. за предыд. часы
Mmsum_01_reset	Real	Масса накоплен. за предыд. часы
Vmsum_01_reset	Real	Объем накоплен. за предыд. часы
Msum_01_reset	Integer	Масса накоплен. за предыд. часы
Vsum_01_reset	Integer	Объем накоплен. за предыд. часы
Vsum15_01_reset	Integer	Объем накоплен. за предыд. часы
Vsum20_01_reset	Integer	Объем накоплен. за предыд. часы
CTL15_	Real	Коэффициент CTL15
CPL15_	Real	Коэффициент CPL15
CTL20_	Real	Коэффициент CTL20
Reset_Hour	Integer	Сброс значений
M1_01	Real	Масса в линии за 1 час
V1_01	Real	Объем в линии за 1 час
V115_01	Real	Объем в линии за 1 час
V120_01	Real	Объем в линии за 1 час

Выходы:

M_01	Real	Масса в линии
V_01	Real	Объем в линии
V15_01	Real	Объем при 15 град.С в линии
V20_01	Real	Объем при 20 град.С в линии
Mn_	Real	Масса нетто в линии
Qm_	Real	Расход массыв линии
Qv_	Real	Расход объема в линии
M1_	Real	Масса в линии за 1 час
V1_	Real	Объем в линии за 1 час
V115_	Real	Объем при 15 град.С за 1 час
V120_	Real	Объем при 20 град.С за 1 час
M2_	Real	Масса в линии за 2 часа
V2_	Real	Объем в линии за 2 часа
V215_	Real	Объем при 15 град.С за 2 часа
V220_	Real	Объем при 20 град.С за 2 часа
Msm_	Real	Масса в линии за смену
Vsm_	Real	Объем в линии за смену
Vsm15_	Real	Объем при 15 град.С за смену
Vsm20_	Real	Объем при 20 град.С за смену
Md_	Real	Масса в линии за сутки
Vd_	Real	Объем в линии за сутки
Vd15_	Real	Объем при 15 град.С за сутки
Vd20_	Real	Объем при 20 град.С за сутки
PRESS_1sr_	Real	Ср. взвешен. давление в ИЛ за 1 час
TEMP_1sr_	Real	Ср. взвешен. температура в ИЛ за 1 час
DENS_1sr_	Real	Ср. взвешен. плотность в ИЛ за 1 час
PRESS_2sr_	Real	Ср. взвешен. давление в ИЛ за 2 часа
TEMP_2sr_	Real	Ср. взвешен. температура в ИЛ за 2 часа
DENS_2sr_	Real	Ср. взвешен. плотность в ИЛ за 2 часа

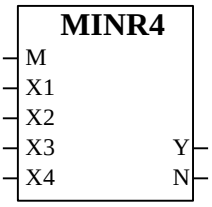
PRESS_smsr_	Real	Ср. взвешен. давление в ИЛ за смену
TEMP_smsr_	Real	Ср. взвешен. температура в ИЛ за смену
DENS_smsr_	Real	Ср. взвешен. плотность в ИЛ за смену
PRESS_dsr_	Real	Ср. взвешен. давление в ИЛ за сутки
TEMP_dsr_	Real	Ср. взвешен. температура в ИЛ за сутки
DENS_dsr_	Real	Ср. взвешен. плотность в ИЛ за сутки
Mmsum_	Real	Масса в линии за месяц
Vmsum_	Real	Объем в линии за месяц
Msum_	Integer	Глобальный счетчик массы в линии
Vsum_	Integer	Глобальный счетчик объема в линии
Vsum15_	Integer	Глобальный счетчик объема в линии
Vsum20_	Integer	Глобальный счетчик объема в линии
err	Integer	Код ошибки: 0 – нет ошибки 1 – ошибка обмена с задачей связи

Назначение

Функциональный блок используется для расчета свойств нефти.

При использовании данного функционального блока на контроллере должна быть запущена задача mbrutto (правила запуска приводятся в документе “Unimod PRO. Исполнительная система”).

7.3.100 MINR4



Входы:

M	INTEGER	Количество используемых входов от 1 до 4
X1	REAL	1-й вход
X2	REAL	2-й вход
X3	REAL	3-й вход
X4	REAL	4-й вход

Выходы:

Y	REAL	Минимум из входов
N	INTEGER	Номер входа с минимальным значением

Назначение

Функция MINR4 используется для выделения минимального из нескольких (до 4) сигналов.

Описание алгоритма

На вход алгоритма поступают сигналы, число которых 0<M<4 и задается на входе M. Выходной сигнал Y равен минимальному из этих сигналов:

$$Y=min\ \{X1;X2;...XM\}$$

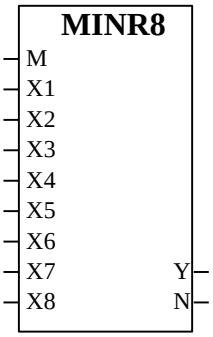
При этом входные сигналы, номер которых больше M не учитываются.

Алгоритм имеет дополнительный выход N, на котором формируется число, равное номеру входного сигнала, прошедшего на выход (т.е. являющегося минимальным). Если имеется группа равных между собой сигналов, причем эти сигналы являются минимальными, то номер N равен минимальному номеру сигналов в этой группе.

При M<1 выходы Y и N равны нулю. Значение M>4 алгоритм воспринимает как M=4.

Примечание: На неиспользуемые входы можно подавать любые значения, т.к. эти значения не будут учитываться при определении минимума.

7.3.101 MINR8



Входы:

M	INTEGER	Количество используемых входов от 1 до 8
X1	REAL	1-й вход
X2	REAL	2-й вход
X3	REAL	3-й вход
X4	REAL	4-й вход
X5	REAL	5-й вход
X6	REAL	6-й вход
X7	REAL	7-й вход
X8	REAL	8-й вход

Выходы:

Y	REAL	Минимум из входов
N	INTEGER	Номер входа с минимальным значением

Назначение

Функция MINR используется для выделения минимального из нескольких (до 8) сигналов.

Описание алгоритма

На вход алгоритма поступают сигналы, число которых 0<M<8 и задается на входе M. Выходной сигнал Y равен минимальному из этих сигналов:

$$Y=min\ \{X1;X2;...XM\}$$

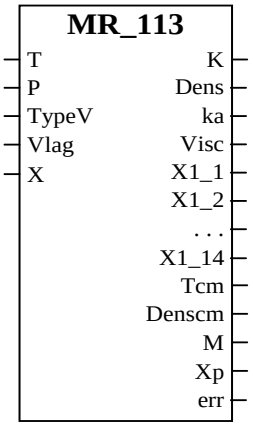
При этом входные сигналы, номер которых больше M не учитываются.

Алгоритм имеет дополнительный выход N, на котором формируется число, равное номеру входного сигнала, прошедшего на выход (т.е. являющегося минимальным). Если имеется группа равных между собой сигналов, причем эти сигналы являются минимальными, то номер N равен минимальному номеру сигналов в этой группе.

При M<1 выходы Y и N равны нулю. Значение M>8 алгоритм воспринимает как M=8.

Примечание: На неиспользуемые входы можно подавать любые значения, т.к. эти значения не будут учитываться при определении минимума.

7.3.102 MR_113



Входы:

T	REAL	Температура газа (град.К)
P	REAL	Давление газа (МПа)
TypeV	BOOLEAN	Тип влажности: FALSE - относительная (%) TRUE - абсолютная (г/куб.м)
Vlag	REAL	Значение влажности
X	#REAL	Ссылка на массив с компонентным составом газа

Выходы:

K	REAL	Коэффициент сжимаемости
Dens	REAL	Плотность газа при рабочих условиях (кг / куб.м)
ka	REAL	Показатель адиабаты
Visc	REAL	Динамическая вязкость (мкПа * с)
X1_1	REAL	Массив концентраций компонентов газа (%)
X1_2	REAL	Массив концентраций компонентов газа (%)
. . .		
X1_14	REAL	Массив концентраций компонентов газа (%)
Tcm	REAL	Критическая температура смеси (град.К)
Denscm	REAL	Критическая плотность смеси (кг/куб.м)
M	REAL	Молярная масса смеси (кг/кмоль)
Xp	REAL	Предельная равновесная концентрация (%)
err	INTEGER	Код ошибки: 0 – нет ошибки 2 - недопустимый размер массива 3 - недопустимая ссылка на массив +4 – недопустимое значение температуры +8 – недопустимое значение давления +16 – недопустимое значение доли компонентов

Внимание! Блок оставлен для совместимости. Вместо данного блока следует использовать ФБ MR_113_V2.

Назначение

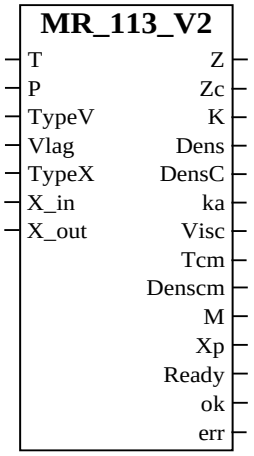
Функциональный блок используется для расчета свойств природного газа по методу ГСССД МР 113-03.
При использовании данного функционального блока на контроллере должна быть запущена задача mr_113 (правила запуска приводятся в документе “Unimod PRO. Исполнительная система”).

Вход X, выходы X1_1 - X1_14 содержат концентрации компонентов газа в следующем порядке:

X[1]	Метан
X[2]	Этан
X[3]	Пропан
X[4]	и-Бутан
X[5]	н-Бутан
X[6]	и-Пентан
X[7]	н-Пентан

X[8]	Гексан
X[9]	Гептан
X[10]	Кислород
X[11]	Азот
X[12]	Диоксид углерода
X[13]	Водяной пар
X[14]	Сероводород

7.3.103 MR_113_V2



Входы:

T	REAL	Температура газа (град.К)
P	REAL	Давление газа (МПа)
TypeV	BOOLEAN	Тип влажности: FALSE - относительная (%) TRUE - абсолютная (г/куб.м)
Vlag	REAL	Значение влажности
TypeX	BOOLEAN	Тип компонентного состава газа: FALSE - в молярных % TRUE - в объемных %
X_in	# REAL	Ссылка на массив с компонентным составом газа (вход)
X_out	# REAL	Ссылка на массив с компонентным составом газа (выход)

Выходы:

Z	REAL	Фактор сжимаемости в рабочих условиях
Zc	REAL	Фактор сжимаемости в стандартных условиях
K	REAL	Коэффициент сжимаемости
Dens	REAL	Плотность газа при рабочих условиях (кг / куб.м)
DensC	REAL	Плотность газа при стандартных условиях (кг / куб.м)
ka	REAL	Показатель адиабаты
Visc	REAL	Динамическая вязкость (мкПа * с)
Tcm	REAL	Критическая температура смеси (град.К)
Denscm	REAL	Критическая плотность смеси (кг/куб.м)
M	REAL	Молярная масса смеси (кг/кмоль)
Xp	REAL	Предельная равновесная концентрация (%)
Ready	BOOLEAN	Признак завершения (TRUE – операция завершена)
ok	BOOLEAN	Код завершения: FALSE – нет ошибок TRUE – есть ошибки
err	INTEGER	Код ошибки

При использовании данного функционального блока на контроллере должна быть запущена задача **mr_113_v2**.

Назначение

Функциональный блок используется для расчета свойств природного газа по методу ГСССД МР 113-03.
Значение влажности должно быть вычислено в нормальных условиях - температура 20град.С и абс. давление 0,101325 кПа.
Выполнение вычислений выполняется асинхронно, не влияя на основной цикл приложения. Анализировать результат можно, когда выход READY примет значение TRUE.

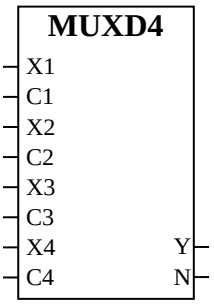
Вход X_in и выход X_out содержат концентрации компонентов газа в следующем порядке:

X[1]	Метан
X[2]	Этан
X[3]	Пропан
X[4]	и-Бутан
X[5]	н-Бутан
X[6]	и-Пентан
X[7]	н-Пентан
X[8]	Гексан
X[9]	Гептан
X[10]	Кислород
X[11]	Азот
X[12]	Диоксид углерода
X[13]	Водяной пар
X[14]	Сероводород

Код ошибки:

- 0 – нет ошибки
- 1 - ошибка обмена с задачей связи
- 2 - недопустимый тип или размер массива (вещественный, [1..14])
- 3 - недопустимая ссылка на массив
- +4 – недопустимое значение температуры (температура газа: от 263 до 500 К);
- +8 – недопустимое значение давления (давление газа: 15 мПа);
- +16 – недопустимое значение доли компонентов
- +32 – недопустимое значение суммы компонентов газовой смеси
- +64 – недопустимое значение относительный влажности
- +128 – недопустимое значение абсолютной влажности

7.3.104 MUXD4



Входы:

X1	REAL	1-й вход
C1	BOOLEAN	Команда выбора входа 1
X2	REAL	2-й вход
C2	BOOLEAN	Команда выбора входа 2
X3	REAL	3-й вход
C3	BOOLEAN	Команда выбора входа 3
X4	REAL	4-й вход
C4	BOOLEAN	Команда выбора входа 4

Выходы:

Y	REAL	Основной выход

N	INTEGER	Порядковый номер выбранного входа от 1 до 4

Назначение

Блок MUXD4 представляет собой многополюсный переключатель аналоговых сигналов, положение которого определяется дискретными сигналами, поступающими на вход алгоритма. Алгоритм используется для выбора одного из нескольких (до 4) сигналов. Если на аналоговых входах алгоритма заданы константы, то алгоритм может использоваться для дискретной установки требуемой константы.

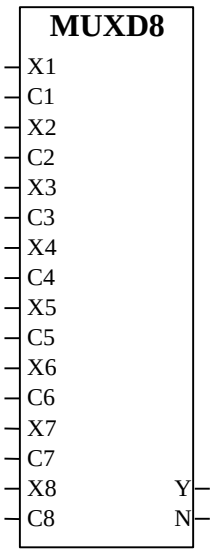
Описание алгоритма

Функциональный блок является переключателем аналоговых сигналов. Если на всех дискретных входах C, управляющих положением переключателя, сигнал отсутствует (лог. 0), выходной сигнал Y=0. Если на какой-либо из дискретных входов подается дискретный сигнал Ci =1, выход алгоритма Y подключается к одноименному (по номеру индекса) аналоговому входу Xi.

Если дискретные сигналы подаются одновременно на несколько входов, приоритетен вход с младшим номером.

Сигнал на выходе Y равен сигналу на выбранном входе. Число на выходе N указывает номер выбранного входа. Если ни один вход не выбран, оба выхода равны нулю Y=0, N=0.

7.3.105 MUXD8



Входы:

X1	REAL	1-й вход
C1	BOOLEAN	Команда выбора входа 1
X2	REAL	2-й вход
C2	BOOLEAN	Команда выбора входа 2
X3	REAL	3-й вход
C3	BOOLEAN	Команда выбора входа 3
X4	REAL	4-й вход
C4	BOOLEAN	Команда выбора входа 4
X5	REAL	5-й вход
C5	BOOLEAN	Команда выбора входа 5
X6	REAL	6-й вход
C6	BOOLEAN	Команда выбора входа 6
X7	REAL	7-й вход
C7	BOOLEAN	Команда выбора входа 7
X8	REAL	8-й вход
C8	BOOLEAN	Команда выбора входа 8

Выходы:

Y	REAL	Основной выход
N	INTEGER	Порядковый номер выбранного входа от 1 до 8

Назначение

Функция MUXD8 представляет собой многополюсный переключатель аналоговых сигналов, положение которого определяется дискретными сигналами, поступающими на вход алгоритма. Алгоритм используется для выбора одного из нескольких (до 8) сигналов. Если на аналоговых входах алгоритма заданы константы, то алгоритм может использоваться для дискретной установки требуемой константы.

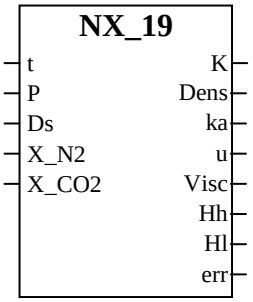
Описание алгоритма

Функциональный блок является переключателем аналоговых сигналов. Если на всех дискретных входах C, управляющих положением переключателя, сигнал отсутствует (лог. 0), выходной сигнал Y=0. Если на какой-либо из дискретных входов подается дискретный сигнал Ci =1, выход алгоритма Y подключается к одноименному (по номеру индекса) аналоговому входу Xi.

Если дискретные сигналы подаются одновременно на несколько входов, приоритетен вход с младшим номером.

Сигнал на выходе Y равен сигналу на выбранном входе. Число на выходе N указывает номер выбранного входа. Если ни один вход не выбран, оба выхода равны нулю Y=0, N=0.

7.3.106 NX_19



Входы:

t	REAL	Температура газа (град.С)
P	REAL	Давление газа (кПа)
Ds	REAL	Плотность газа при стандартных условиях (кг / куб.м)
X_N2	REAL	Молярная плотность азота (%)
X_CO2	REAL	Молярная плотность углекислого газа (%)

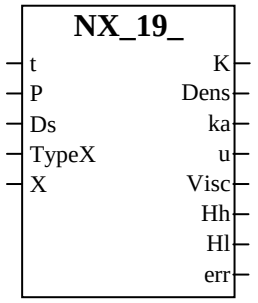
Выходы:

K	REAL	Коэффициент сжимаемости
Dens	REAL	Плотность газа при рабочих условиях (кг / куб.м)
ka	REAL	Показатель адиабаты
u	REAL	Скорость звука в газе (м / с)
Visc	REAL	Динамическая вязкость (мкПа * с)
Hh	REAL	Высшая удельная теплота сгорания (МДж / куб.м)
Hl	REAL	Низшая удельная теплота сгорания (МДж / куб.м)
err	INTEGER	Код ошибки: 0 - нет ошибки; 3 – недопустимый размер массива; 4 – недопустимая ссылка на массив; -1 - недопустимое значение температуры; -2 - недопустимое значение давления; -4 - недопустимое значение плотности; -8 – недопустимое значение доли азота; -16 – недопустимое значение доли углекислого газа.

Назначение

Функциональный блок используется для расчета свойств природного газа по методу NX_19 (ГОСТ 30319.2-96).

7.3.107 NX_19_



Входы:

t	REAL	Температура газа (град.С)
P	REAL	Давление газа (кПа)
Ds	REAL	Плотность газа при стандартных условиях (кг/куб.м)
TypeX	BOOLEAN	Тип компонентного состава газа: FALSE – в молярных %; TRUE – в объемных %.
X	#REAL	Имя массива с компонентным составом газа (элементы типа Real)

Выходы:

K	REAL	Коэффициент сжимаемости
Dens	REAL	Плотность газа при рабочих условиях (кг / куб.м)
ka	REAL	Показатель адиабаты
u	REAL	Скорость звука в газе (м / с)
Visc	REAL	Динамическая вязкость (мкПа * с)
Hh	REAL	Высшая удельная теплота сгорания (МДж / куб.м)
Hl	REAL	Низшая удельная теплота сгорания (МДж / куб.м)
err	INTEGER	Код ошибки: 0 - нет ошибки; 3 – недопустимый размер массива; 4 – недопустимая ссылка на массив; -1 - недопустимое значение температуры; -2 - недопустимое значение давления; -4 - недопустимое значение плотности; -8 – недопустимое значение доли азота; -16 – недопустимое значение доли углекислого газа.

Назначение

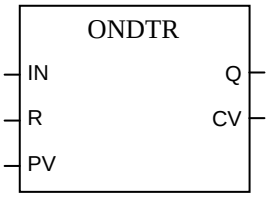
Функциональный блок используется для расчета свойств природного газа по методу NX_19 (ГОСТ 30319.2-96, Изменение №1 к ГОСТ 30319.1-96 от 01.06.2004).

Вход X содержит концентрации компонентов газа в следующем порядке:

X[1]	Метан
X[2]	Этан
X[3]	Пропан
X[4]	н-Бутан
X[5]	и-Бутан
X[6]	н-Пентан
X[7]	и-Пентан
X[8]	н-Гексан
X[9]	н-Гептан
X[10]	н-Октан
X[11]	Ацетилен
X[12]	Этилен
X[13]	Пропилен
X[14]	Бензол
X[15]	Толуол
X[16]	Водород

X[17]	Водяной пар
X[18]	Аммиак
X[19]	Метанол
X[20]	Сероводород
X[21]	Метилмеркаптан
X[22]	Диоксид серы
X[23]	Гелий
X[24]	Неон
X[25]	Аргон
X[26]	Монооксид углерода
X[27]	Азот
X[28]	Воздух
X[29]	Кислород
X[30]	Диоксид углерода

7.3.108 ONDTR



Входы:

IN	BOOLEAN	TRUE – разрешить таймер
R	BOOLEAN	TRUE – сбросить таймер
PV	INTEGER	Максимальное время отсчета (мсек)

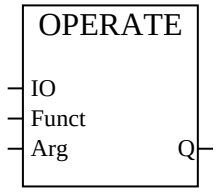
Выход:

Q	BOOLEAN	TRUE, если время RV истекло
CV	INTEGER	Текущее время, отсчитываемое таймером

Назначение

Таймер задержки по включению со сбросом.

7.3.109 OPERATE, OPERATE_F



Входы:

IO	#Ввод/вывод	Переменная ввода/вывода
Funct	INTEGER	Код запроса
Arg	INTEGER	Аргумент

Выход:

Q	INTEGER	Результат вызова

Назначение:
Тестирование и инициализация юнита/мезонина

Описание:

ВНИМАНИЕ! ФБ OPERATE используется для совместимости со старыми версиями. Рекомендуется использовать OPERATE_F.

Вызов функции ***operate*** используется для специальных запросов к физическим каналам, которые связаны с переменными ввода/вывода. Функция возвращает значение целого типа и имеет следующий формат: **operate(переменная В/В, code, arg)**, где:

- переменная В/В - переменная ввода/вывода, переменная модульной структуры;
- code - код запроса;
- arg - аргумент, для выполнения запроса.

Примечание.

Применение не указанных в настоящем руководстве запросов не допускается.

В результате вызова OPERATE с неизвестным кодом команды или некорректным номером юнита, функция возвращает нулевое значение. OPERATE возвращает ноль, если юнит с указанным номером не установлен или не поддерживает данную функцию.

Аргумент “Arg” должен быть сброшен в ноль, если не используется командой явно.

Для Мастер-M911E поддерживается только код **16#0002 (2),** выполняющий проверку достоверности чтения диагностических переменных и переменных ввода/вывода.

Для Мастер-M841E в результате вызова OPERATE с неизвестным кодом команды, функция возвращает “-1”.

16#0001 (1): Запрос на переинициализацию юнита

Команда производит переинициализацию юнита. Обновление входных переменных для юнитов ввода задерживается на время, необходимое для инициализации юнита. Функция возвращает единицу, если команда выполнена успешно, ноль в противном случае.

Флаг ошибки работы юнита, если был установлен перед вызовом данной функции, сбрасывается только в случае (и после) успешной переинициализации юнита.

16#0002 (2): Чтение состояния юнита / операции ввода/вывода

Функция возвращает код завершения последней операции ввода/вывода по физическому каналу, к которому привязана переменная, указанная в качестве первого аргумента. Анализируя возвращаемый код, можно определить степень достоверности информации, находящейся в переменной ввода/вывода.

Диагностируемые ошибки имеют следующие коды:
Для мастер-модуля **M911E** и для модулей **M800**/**M900**:

- 0 – ошибок не обнаружено;

- 1 – команда не поддерживается модулем (несоответствие версии модуля);
- 2 – неверный формат запроса (несоответствие версии модуля);
- 3 – выполнение запроса недопустимо в текущем режиме работы модуля;
- 4 – выполнение запроса невозможно, модуль остановлен переключателем RUN/STOP;
- 5 – превышение допустимого размера приложения, загрузка невозможна;
- 15 – таймаут при ожидании ответа;
- 255 – выполнение запроса невозможно, модуль занят;
- 300 – перегрузка цепей юнита;
- 400 – короткое замыкание во внешней цепи;
- 500 – обрыв внешней цепи;
- 600 – отсутствие напряжения на внешнем источнике питания;
- 700 – аппаратная ошибка в работе канала;
- 800 – ошибка метрологических констант;
- +1000 – значение не достоверно;
- +2000 – отладочный режим, переменная заблокирована отладчиком;

Для мастер-модулей **M841E/M902E/M921E/M915E/M903E**:

a) Диагностика обмена с модулями ввода/вывода:

- 0 – ошибок не обнаружено;
- +2 – ошибка переполнения UART;
- +4 – ошибка на линии;
- +8 – ошибка адреса (ответ идет с отличным от запроса адресом);
- +16 – ошибочное количество байт данных;
- +32 – повторная стартовая комбинация;
- +64 – ошибка контрольной суммы;
- +128 – таймаут при ожидании очередного байта на приеме;
- +256 – нет параметров (перезапуск модуля)
- +1000 – значение не достоверно;
- +2000 – отладочный режим, переменная заблокирована отладчиком;

b) Диагностика межконтроллерного обмена:

- 0 – ошибок не обнаружено;

Ошибки пакета:

- 101 – превышение размера фрейма;
- 102 – ошибочный формат пакета;
- 103 – ошибка контрольной суммы заголовка;
- 104 – ошибка контрольной суммы пакета;

Ошибки данных:

- 201 – объем данных локального и удаленного узла не совпадает;
- 202 – ошибочная длина данных;
- 203 – ошибочный идентификатор сеанса;
- 204 – ошибочный порядок пакетов;
- 205 – превышение количества пакетов;
- 206 – превышение объема данных;
- 207 – ошибка контрольной суммы;
- 208 – состав переменных локального и удаленного узла не совпадает
- 209 – ошибка внутренних идентификаторов;
- 210 – ошибка режимов работы;

- 1000 – значение не достоверно;

Примечание (относится только к диагностике обмена с модулями M500).
Кроме указанных кодов, для мастер-модулей ***M915E/M903E*** при обмене с модулями ***M500*** функция анализирует поканальную диагностику. Т.е. если в качестве первого аргумента указать переменную, содержащую значение канала, то даже в случае успешного чтения/записи значения функция operate_f может вернуть ненулевое значение, если в поканальной диагностике для данного канала содержится ненулевое значение:

- 256+Err, где Err – значение поканальной диагностики для данного канала.

16#0003 (3): Установка недостоверного значения

Функция устанавливает значение, равное arg, принимаемое переменной, указанной в качестве первого аргумента, при возникновении ошибки выполнения операции ввода/вывода. Используется для булевских и аналоговых переменных. В случае успешного выполнения функция возвращает установленное значение.

16#0004 (4): Отмена установки недостоверного значения

Аргумент “Arg” не используется. Функция отменяет действие вызова функции OPERATE с кодом 3 для переменной, указанной в качестве первого аргумента. При возникновении ошибки выполнения операции ввода/вывода значение переменной остается предыдущим. В результате вызова OPERATE в “Q” прописывается единица.

16#0005 (5): Чтение информации из расширенного состояния модуля

Результат вызова функции зависит от значения аргумента:	
01h(1)	Режим работы
02h(2)	Глобальные флаги ошибок
03h(3)	Критические ошибки приложения
04h(4)	Код сброса
05h(5)	Версия ПО
06h(6)	Тип модуля
07h(7)	Аппаратная конфигурация
08h(8)	Динамические ошибки приложения
09h(9)	Флаги ошибок окружения
0Ah(10)	Аппаратные ошибки работы модулей
0Bh(11)	Ошибки конфигурации модулей
0Ch(12)	Аппаратные ошибки в работе юнитов
0Dh(13)	Ошибки чтения метрологических констант
0Eh(14)	Ошибки внешних цепей юнитов
0Fh(15)	Флаги ошибок связи

16#0006 (6): Чтение/запись модульных переменных с режимом “по запросу”

Функция выполняет действие в соответствии со значением аргумента “Arg”:

1 – Выполнить чтение переменной;
2 – Выполнить запись переменной.

Если присутствует ошибка во входных параметрах (например, попытка чтения/записи переменной с несоответствующим режимом), результат вызова функции будет равен нулю. В случае успешного выполнения функция возвращает единицу.

16#000A (10): Чтение типа юнита (регистры CCR0…CCR1).

Функция возвращает код типа юнита. В случае, когда юнит не установлен или если номер юнита выходит за номинальный диапазон, результат вызова функции будет равен нулю.

16#000B (11): Чтение состояния физических выводов юнита.

Физический интерфейс юнита состоит из четырёх двунаправленных линий D0..D3 и двух входных диагностических линий S0 и S1. Данная команда выполняет чтение непосредственно физических выводов юнита и возвращает маску их состояния. Результат вызова функции:

MSB 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 S1 S0 D3 D2 D1 D0 **LSB**

Если адресуемый юнит не имеет физического интерфейса (например, виртуальные юниты термометра термокомпенсации, юнит состояния и диагностики модуля и т.п.), результат вызова функции будет равен нулю. При возникновении аппаратной ошибки во время доступа к регистрам юнита, результат устанавливается в -1.

16#000C (12): Управление индикатором юнита.

Эта команда разрешает или запрещает управление, а также устанавливает состояние индикатора юнита. Верхние 24 бита аргумента “Arg” не используются и могут принимать любые значения. Младшие 8 бит описаны подробнее:

MSB X X X X X X X X X X X X X X X X X X X X X X X X M1 M0 DF EN S3 S2 S1 S0 **LSB**

Биты S3..S0: Собственно новое состояние светодиодной матрицы индикатора.
Бит EN: Включить/отключить (1/0) управление индикатором.
Бит DF: Разрешить отображение ошибки в работе юнита по умолчанию. Если бит DF установлен, в случае возникновения аппаратной ошибки юнита, контроль над индикатором принимает интеллектуальный модуль и сигнализирует ошибку миганием светодиодов юнита синхронно с красным светодиодом состояния модуля.
Биты M1, M0: Управление режимами мигания светодиодов. 00 – мигание выключено; 01 – мигают включенные светодиоды, а выключенные остаются выключенными; 10 – мигают выключенные светодиоды, включенные остаются включенными; 11 – попеременно зажигаются то включенные, то включенные светодиоды индикатора.
Биты, обозначенные как X: не используются, могут принимать любые значения.

В случае использования приложением модуля энергонезависимой памяти, состояние индикатора, установленное данной функцией OPERATE, не сохраняется.

16#0011 (17): Запрос на переинициализацию юнита с аргументом.

Команда производит переинициализацию юнита. Значение аргумента “Arg” зависит от типа установленного юнита (по умолчанию - ноль). Обновление входных переменных для юнитов ввода задерживается на время, необходимое для инициализации юнита. Функция возвращает единицу, если команда выполнена успешно, ноль в противном случае.
Флаг ошибки работы юнита, если был установлен перед вызовом данной функции OPERATE, сбрасывается только после успешной переинициализации юнита.

16#0015 (21): Блокировка опроса модуля ввода/вывода.

В зависимости от значения аргумента “Arg” функция выполняет следующие действия:

1 – блокировать опрос модуля в/в;
0 – возобновить опрос модуля в/в.

На транзитные запросы блокировка не распространяется. Также, вызов функции с аргументом, равным 1, сбрасывает невыполненные запросы на чтение/запись по всем переменным данного модуля с режимом чтения/записи «по запросу».

В случае успешного выполнения функция возвращает установленное значение.

16#001A (26): Чтение регистров состояния юнита (CCR2…CCR3).

Аргумент “Arg” не используется. Результатом вызова функции является битовая маска состояния выбранного юнита. Значение неописанных здесь битов не всегда равно нулю и зависит от типа юнита. Список общих флагов, находящихся в регистре состояния юнита:

Бит 4: Юнит находится в режиме инициализации. Устанавливается при возникновении аппаратной ошибки или запросом с кодом команды 01h через функцию OPERATE.
Бит 5: Отсутствие напряжения на внешнем источнике питания. Фиксируется при отсутствии напряжения на диагностических выводах юнита.
Бит 6: Обрыв внешней цепи.
Бит 7: Обнаружена аппаратная ошибка работы юнита.
Бит 8: Ошибка метрологических констант канала. Несоответствие существующих констант выбранному типу юнита.
Бит 9: Переменная, подключенная к каналу, заблокирована.
Бит 10: Короткое замыкание внешней цепи.

Полный список флагов зависит от реализации драйвера юнита, а также описание битов регистра состояния и находится в спецификации конкретного юнита.

16#001B (27): Запись специальных регистров юнита (CSR0…CSR3).

16#002A (42): Чтение входных регистров юнита (CIR0…CIR3).

16#002B (43): Запись выходных регистров юнита (COR0…COR3).

16#003A (58): Чтение входных регистров юнита (CIR4…CIR7).

16#003B (59): Запись выходных регистров юнита (COR4…COR7).

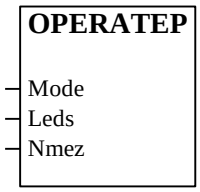
16#004A (74): Чтение специальных регистров юнита (CSR0…CSR3).

Реализация данной функции, как и возвращаемое вызовом OPERATE значение зависит от типа установленного юнита. Если не сказано иначе, OPERATE возвращает ноль.

16#0064 (100): Конфигурация интерфейса ввода (подробное описание в документе «Unimod Pro. Пульт оператора M920L»).

16#0065 (101): Конфигурация стандартного вывода (подробное описание в документе «Unimod Pro. Пульт оператора M920L»).

7.3.110 OPERATEP



Входы:

Mode	INTEGER	Режим работы мезонина
Leds	INTEGER	Индикация
NMez	INTEGER	Номер мезонина

Назначение:

Устанавливает режим работы и индикации пожарного юнита/мезонина

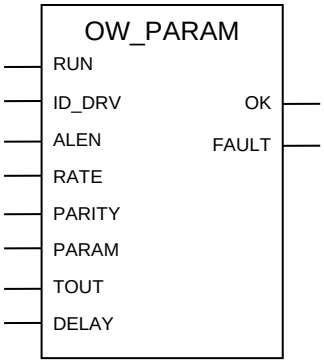
Mode - Управление режимом работы

- 16#00 - перейти в нормальный режим работы
- 16#10 - выдавать всегда положительную полярность (поверка)
- 16#20 - дать отрицательную полярность на 3 секунды (сброс датчиков)
- 16#40 - изменить состояние светодиодов

Leds - Индикация светодиодами

- 1 - внимание "сработал 1 датчик"
- 2 - "пожар"
- 4 - дежурный режим "шлейф в порядке"
- 8 - обрыв или КЗ в линии
- 0 - ошибка юнита/мезонина

7.3.111 OW_PARAM



Входы:

RUN	BOOLEAN	Запуск функционального блока
ID_DRV	INTEGER	Идентификатор коммуникационного адаптера
ALEN	INTEGER	Длина адреса устройства (0 = 8 бит, 1 = 11 бит)
RATE	INTEGER	Скорость передачи по последовательной линии
PARITY	INTEGER	Количество стоповых бит и режим контроля четности
PARAM	INTEGER	Режим работы синхронный/асинхронный
TOUT	INTEGER	Время ожидания ответа от Slave-устройства (мс)
DELAY	INTEGER	Длительность паузы между передачами (мс)

Выходы:

OK	BOOLEAN	Признак завершения операции
FAULT	INTEGER	Код ошибки

Назначение:
Конфигурация связи по протоколу ОВЕН (на мастер-модуле должна быть запущена задача связи **ow_rtu_mst**).

Описание:

Изменение установок Овен происходит при единичном значении входа RUN. Новые установки Овен вступают в силу незамедлительно. Текущий приём или передача посылки будет прервана, что может привести к обнаружению ошибок передачи мастер-контроллером.

Вход ID_DRV задаёт идентификатор физической линии (COM-порт).

В протоколе используется 11-и и 8-и битная адресация. Длина адреса задается входом ALEN (0 = 8 бит, 1 = 11 бит). Соответственно, максимальное значение базового адреса при 8-и битной адресации - 248, при 11-и битной адресации - 2040.

Аргументы RATE и PARITY используются для конфигурирования последовательной линии RS-485. Через вход RATE задается скорость обмена. RATE может принимать одно из следующих значений:

1200, 2400, 4800, 9600, 14400, 19200, 28800, 38400, 57600, 15200, бод.

Количество стоповых битов и режим контроля четности задается входом PARITY. Возможные варианты режима контроля четности представлены в таблице 1.

Таблица 1 – Режим контроля четности

Значение	Количество стоповых бит	Контроль четности
0	1	Отключен
1	2	Отключен
2	1	EVEN
3	1	ODD
4	1	SPACE
5	1	MARK

Формат аргумента PARAM имеет следующий вид:

MSB X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X HDUP SYNC **LSB**

Формат аргумента PARAM

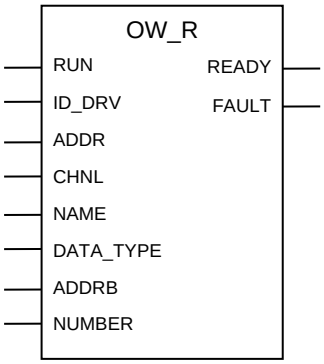
SYNC (разряд 0, synchronous). Если флаг установлен, обмен со Slave-устройствами происходит в синхронном режиме: вызванный функциональный блок не возвращает управление технологической программе до тех пор, пока текущая операция не будет завершена. При сброшенном флаге SYNC функциональные блоки работают асинхронном режиме, а момент завершения операции определяется состоянием выходов READY.

HDUP (разряд 1, half-duplex). Флаг контролирует режим обмена по линии RS-485. Сброшенный флаг HDUP устанавливает полнодуплексный режим обмена по двум дифференциальным линиям, отдельно для приема и передачи (RS-422). Установка флага HDUP разрешает при отсутствии посылки перевод передатчика RS-485 в третье состояние; таким образом, становится возможна двунаправленный обмен пакетами по одной дифференциальной линии – режим half-duplex.

Параметры TOUT и DELAY задают время ожидания ответа от Slave-устройства и задержку после получения ответа и перед следующей передачей соответственно. Все значения задаются в миллисекундах.

После завершения операции, выход функционального блока OK устанавливается в состояние TRUE, а на выходе FAULT – код ошибки или нуль, если вызов отработан без ошибок.

7.3.112 OW_R



Входы:

RUN	BOOLEAN	Запуск функционального блока
ID_DRV	INTEGER	Идентификатор коммуникационного адаптера
ADDR	INTEGER	Базовый адрес удаленного устройства (0 - 2040 через 8)
CHNL	INTEGER	Номер канала устройства (0 - 7)
NAME	INTEGER	Сетевое имя параметра
DATA_TYPE	INTEGER	Тип данных: 0 по умолчанию 1 4х байтовое целое без знака 2 4х байтовое целое со знаком (с десятичной точкой) 3 4х байтовое вещественное 4 3х байтовое вещественное
ADDRB	INTEGER	Начальный адрес данных в базе (адрес Modbus)
NUMBER	INTEGER	Количество каналов

Выходы:

READY	BOOLEAN	Признак завершения операции
FAULT	INTEGER	Код ошибки

Назначение:

Чтение группы данных с подчиненного устройства по протоколу ОВЕН непосредственно в базу UnimodPRO.

Вход ADDR задает базовый адрес подчиненного устройства ОВЕН. В протоколе используется 11-и и 8-и битная адресация. Длина адреса задается входом ALEN функционального блока OW_PARAM. Соответственно, максимальное значение базового адреса при 8-и битной адресации - 248, при 11-и битной адресации - 2040. Значение базового адреса должно быть кратным 8.

Функциональный блок автоматически определяет тип параметра по его имени и выполняет соответствующие преобразования (при этом вход Data_type равен нулю). Ненулевое значение на входе Data_type устанавливает тип принудительно. Типы протокола Овен преобразуются в типы UnimodPRO следующим образом:

4х байтовые целые числа без знака представляются в базе целыми числами, остальные - вещественными.

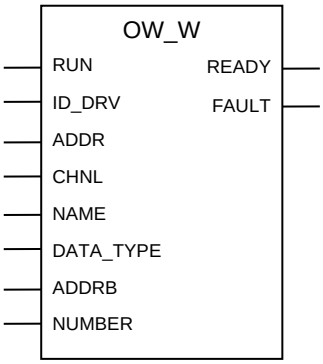
После завершения операции, выход функционального блока READY устанавливается в состояние TRUE, а на выходе FAULT - код ошибки или нуль, если вызов отработан без ошибок.

Примечание

Для корректного отображения данных, адреса переменных в базе, куда выполняется чтение, должны следовать через 1, так как данные преобразуются к 32х разрядному виду.

Протокол ОВЕН не предусматривает групповых запросов, поэтому если аргумент Number имеет значение больше единицы, блок сам выполняет последовательное чтение каналов несколькими запросами. Запись в базу происходит только после успешного чтения всех каналов. При этом выход READY устанавливается в TRUE.

7.3.113 OW_W



Входы:

RUN	BOOLEAN	Запуск функционального блока
ID_DRV	INTEGER	Идентификатор коммуникационного адаптера
ADDR	INTEGER	Базовый адрес удаленного устройства (0 - 2040 через 8)
CHNL	INTEGER	Номер канала устройства (0 - 7)
NAME	INTEGER	Сетевое имя параметра
DATA_TYPE	INTEGER	Тип данных: 0 по умолчанию 1 4х байтовое целое без знака 2 4х байтовое целое со знаком (с десятичной точкой) 3 4х байтовое вещественное 4 3х байтовое вещественное
ADDRB	INTEGER	Начальный адрес данных в базе (адрес Modbus)
NUMBER	INTEGER	Количество каналов

Выходы:

READY	BOOLEAN	Признак завершения операции
FAULT	INTEGER	Код ошибки

Назначение:

Запись группы данных на подчиненное устройство по протоколу ОВЕН непосредственно из базы UnimodPRO.

Вход ADDR задает базовый адрес подчиненного устройства ОВЕН. В протоколе используется 11-и и 8-и битная адресация. Длина адреса задается входом ALEN функционального блока OW_PARAM. Соответственно, максимальное значение базового адреса при 8-и битной адресации - 248, при 11-и битной адресации - 2040. Значение базового адреса должно быть кратным 8.

Функциональный блок автоматически определяет тип параметра по его имени и выполняет соответствующие преобразования (при этом вход Data_type равен нулю). Ненулевое значение на входе Data_type устанавливает тип принудительно. Типы протокола Овен преобразуются в типы UnimodPRO следующим образом:

> 4х байтовые целые числа без знака представляются в базе целыми числами, остальные - вещественными.

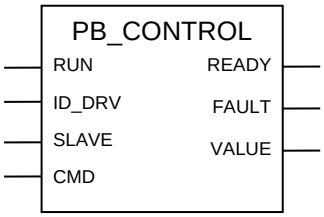
После завершения операции, выход функционального блока READY устанавливается в состояние TRUE, а на выходе FAULT - код ошибки или нуль, если вызов отработан без ошибок.

Примечание

Для корректного преобразования данных, адреса переменных в базе, откуда выполняется запись, должны следовать через 1, так как данные преобразуются к 32х разрядному виду.

Протокол ОВЕН не предусматривает групповых запросов, поэтому если аргумент Number имеет значение больше единицы, блок сам выполняет последовательную запись каналов несколькими запросами. Только после успешной записи всех каналов выход READY устанавливается в TRUE.

7.3.114 PB_CONTROL



Входы:

RUN	Boolean	Запуск функционального блока
ID_DRV	Integer	Идентификатор устройства (1..4)
Slave	Integer	Адрес подчиненного устройства
Cmd	Integer	Команда

Выходы:

Ready	Boolean	Признак завершения операции
Fault	Integer	Код завершения операции 0 – выполнено успешно 1 – ошибка выделения памяти (некорректные параметры) 4 – ошибка инициализации задачи связи
Value	Integer	Возвращаемое значение

Назначение:
Управление интерфейсом Profibus-DP.

Описание:

Функциональный блок отправляет служебные команды плате Profibus-DP Master.

Команды 0-6 выполняются по фронту входа **RUN**, команда 7 – при единичном значении **RUN**.

Аргумент **Slave** должен сброшен в ноль, если не используется блоком явно.

Код команды задается входом **CMD**. Возможны следующие значения:

0 - Сброс платы. Обмен прекращается
1 - Перезапуск платы с переконфигурированием
2 - Запуск обмена данными
3 - Останов обмена данными (все выходы устанавливаются в ноль, чтение продолжается)
4 - Прочитать входы (из коммуникационного устройства в память драйвера).
Команда используется совместно с параметром “t” (см. web-конфигуратор)
5 - Записать выходы (из памяти драйвера в коммуникационное устройство).
Команда используется совместно с параметром “t” (см. web-конфигуратор)
6 - Выполнить цикл обмена данными.
Команда используется совместно с параметром "c" (см. web-конфигуратор)
7 - Чтение состояния обмена с ведомым устройством (адрес устройства задается аргументом Slave). После успешного выполнения блока, выход **Value** принимает нулевое значение, если прошлый цикл обмена с устройством выполнен успешно; 1 – в противном случае.

Драйвер платы Profibus-DP Master может запускаться в нескольких режимах:

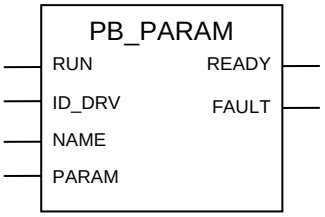
1) Стандартный циклический обмен. Драйвер при старте выделяет разделяемую память, инициализирует обмен и в каждом цикле выполняет обмен с ведомыми устройствами: отправляет ведомому устройству данные из разделяемой памяти и записывает в нее считанные данные.
2) Синхронный обмен между платой и драйвером (см. web-конфигуратор, **параметр “t”**). Драйвер также в каждом цикле выполняет обмен, но считанные данные хранит в своей внутренней памяти и необходимо подать команду “4”, чтобы драйвер переложил считанные данные в разделяемую память. Аналогично для команды “5”.
3) Синхронный обмен с подчиненными устройствами (см. web-конфигуратор, **параметр “c”**). Драйвер инициализирует память, но обмен не начинает. Чтобы выполнить однократный цикл обмена, необходимо подать команду “6”.

Примечание: интервалы между вызовами команды “6” в таком случае должны быть не более времени **T_wd=(*10мс*wdfact1*wdfact2*)**, где wdfact1 и wdfact2 – значения, задаваемые в конфигураторе DPconf для каждого ведомого устройства.

4) Совмещение режимов 2 и 3.

На контроллере должна быть запущена задача pb_mst.

7.3.115 PB_PARAM



Входы:

RUN	Boolean	Триггер запуска функционального блока
ID_DRV	Integer	Идентификатор устройства (1..4)
Name	Message	Имя сервера (по умолчанию – PROFIBUS)
Param	Integer	Дополнительные параметры

Выходы:

READY	Boolean	Признак завершения операции
FAULT	Integer	Код завершения операции 0 – выполнено успешно 1 – ошибка выделения памяти (некорректные параметры) 4 – ошибка инициализации задачи связи

Назначение:
Конфигурация связи по протоколу Profibus-DP.

Описание:

Изменение установок Profibus происходит по фронту входа **RUN**. Функциональный блок открывает доступ к плате с номером ID_DRV и именем Name.

Вход **ID_DRV** задает номер PCI-слота платы (должен соответствовать параметру “b” в настройках Profibus-DP Master в web-конфигураторе). Номер устанавливается перемычкой на плате.

Вход **Name** определяет имя сервера Profibus. Должно соответствовать названию проекта в конфигураторе DPconf.

Формат аргумента **PARAM** имеет следующий вид:

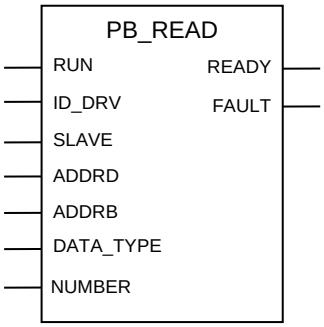
MSB X X X X X X X … X X X X X X X X X SYNC LSB

SYNC (разряд 0, synchronous). Если флаг установлен, вызванный функциональный блок не возвращает управление технологической программе до тех пор, пока текущая операция не будет завершена. При сброшенном флаге SYNC функциональные блоки работают в асинхронном режиме, а момент завершения операции определяется состоянием выходов READY. В целях безопасности не все операции могут выполняться синхронно. Долгие операции, как например «Перезапуск платы», принудительно выполняются асинхронно, во избежание срабатывания Watchdog-таймера. Синхронно могут выполняться, например, чтение/запись данных и чтение диагностики.

После успешного выполнения блока с платой можно взаимодействовать.

На контроллере должна быть запущена задача **pb_mst**.

7.3.116 PB_READ



Входы:

RUN	BOOLEAN	Запуск функционального блока
ID_DRV	INTEGER	Идентификатор устройства (1..4)
Slave	INTEGER	Адрес подчиненного устройства
AddrD	INTEGER	Начальный адрес данных на подчиненном устройстве
AddrB	INTEGER	Начальный адрес данных в базе (Адрес Modbus – 1)
DATA_TYPE	INTEGER	Тип данных: 0 дискретное (1 байт) 1 дискретное (2 байта) 2 дискретное (3 байта) 3 дискретное (4 байта) 4 2х байтовое целое без знака 5 2х байтовое целое со знаком 6 4х байтовое целое 7 4х байтовое вещественное
Number	INTEGER	Количество данных

Выходы:

Ready	BOOLEAN	Признак завершения операции
Fault	INTEGER	Код завершения операции 0 – выполнено успешно 1 – ошибка выделения памяти (некорректные параметры) 4 – ошибка инициализации задачи связи

Назначение:
Чтение группы данных с подчиненного устройства непосредственно в базу UnimodPRO.

Описание:

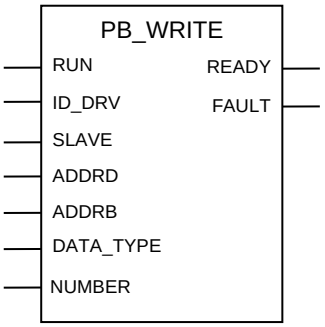
ФБ не выполняет команд чтения с подчиненным устройством, а просто читает данные из разделяемой памяти драйвера и преобразует их для использования в технологическом приложении в соответствии с аргументом **DATA_TYPE**.

При чтении 2х байтовых целых, 4х байтовых целых и 4х байтовых вещественных, формат чисел автоматически преобразуется из формата Profibus в формат UnimodPRO.

Примечание: Modbus- адреса всех переменных в базе должны идти через один, независимо от размерности читаемой переменной.

На контроллере должна быть запущена задача **pb_mst**.

7.3.117 PB_WRITE



Входы:

RUN	BOOLEAN	Запуск функционального блока
ID_DRV	INTEGER	Идентификатор устройства (1..4)
Slave	INTEGER	Адрес подчиненного устройства
AddrD	INTEGER	Начальный адрес данных на подчиненном устройстве
AddrB	INTEGER	Начальный адрес данных в базе (Адрес Modbus – 1)
DATA_TYPE	INTEGER	Тип данных: 0 дискретное (1 байт) 1 дискретное (2 байта) 2 дискретное (3 байта) 3 дискретное (4 байта) 4 2х байтовое целое без знака 5 2х байтовое целое со знаком 6 4х байтовое целое 7 4х байтовое вещественное
Number	INTEGER	Количество данных

Выходы:

Ready	BOOLEAN	Признак завершения операции
Fault	INTEGER	Код завершения операции 0 – выполнено успешно 1 – ошибка выделения памяти (некорректные параметры) 4 – ошибка инициализации задачи связи

Назначение:
Запись группы данных на подчиненное устройство непосредственно из базы UnimodPRO.

Описание:

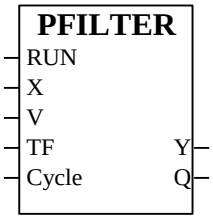
ФБ не выполняет команд записи в подчиненное устройство, а просто записывает данные в разделяемую память драйвера в соответствии с аргументом **DATA_TYPE**.

При записи 2х байтовых целых, 4х байтовых целых и 4х байтовых вещественных, формат чисел автоматически преобразуется из формата UnimodPRO в формат Profibus.

Примечание: Modbus- адреса всех переменных в базе должны идти через один, независимо от размерности записываемой переменной.

На контроллере должна быть запущена задача **pb_mst**.

7.3.118 PFILTER



Входы:

RUN	BOOLEAN	Режим работы
X	REAL	Входной сигнал
V	REAL	Максимальная скорость изменения сигнала (1/сек)
TF	REAL	Максимальная длительность помехи (сек)
Cycle	INTEGER	Период работы блока (мсек)

Выходы:

Y	REAL	Основной выход
Q	BOOLEAN	Признак превышения скорости

Назначение

Функциональный блок PFILTER осуществляет фильтрацию входных аналоговых сигналов от помех с заданными параметрами.

Описание алгоритма

При состоянии входа RUN=TRUE в каждом i-ом цикле выполнения алгоритма определяется реальная скорость изменения входного сигнала X относительно выходного Y:

$$VX_i = \frac{X_i - Y_{i-1}}{Tc},$$

где X_i - значение входного сигнала в текущем цикле; Y_{i-1} - значение выходного сигнала в предыдущем цикле.

Значение Tc приближенно равно входному параметру Cycle и кратно времени цикла контроллера.

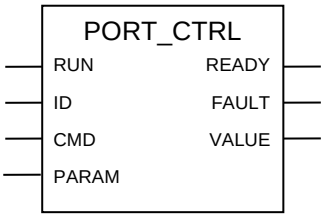
Если скорость изменения входного сигнала не больше V, т.е. $|V_i| \le V$, то входной сигнал передается на выход без изменения, при этом Y=X, Q=FALSE. Если в каком-либо цикле обнаружено, что $|VX_i| \ge V$, то предполагается возможность случайного выброса и выходной сигнал $Y_i=Y_{i-1}$, при этом запускается измерение длительности Tp этой ситуации. Выходной сигнал не изменяется до тех пор, пока одновременно выполняются отношения $|VX_i| \ge V$ и Tp < TF, при этом Q=TRUE. В противном случае, при $|VX_i| \le V$ выходной сигнал Y начинает опять отслеживать входной сигнал, т.е. $Y_i=X_i$, при этом Q=FALSE. Если время Tp истекло, т.е. Tp≥TF, то выходной сигнал также начинает отслеживать входной сигнал, при этом значение Tp обнуляется Tp=0.

Таким образом, изменение входного сигнала со скоростью больше допустимой, рассматривается алгоритмом как случайный выброс, который “вырезается” из входного сигнала, если длительность этого изменения меньше заданного TF. Если длительность такого изменения сигнала больше заданного TF, то это изменение рассматривается алгоритмом как естественное и передается на выход блока с задержкой времени TF.

При состоянии входа RUN=FALSE выход блока Y равен входу X, т.е. фильтрации не происходит. При переходе входного сигнала RUN из состояния FALSE в состояние TRUE VX=0.

Примечание: при значении Cycle=0 период работы блока равен циклу контроллера. При значении V≤0 или TF≤0 входной сигнал X фильтроваться не будет.

7.3.119 PORT_CTRL



Входы:

RUN	Boolean	Запуск функционального блока
ID	Integer	Идентификатор файла порта
CMD	Integer	Код команды
PARAM	Integer	Параметры команды

Выходы:

READY	Boolean	Признак завершения операции
FAULT	Integer	Код завершения операции 0 – выполнено успешно 1 – некорректные параметры 3 – системная ошибка при работе с портом
Value	Integer	Возвращаемое значение

Назначение:
Управление портом.

Описание:
Функциональный блок отправляет служебные команды порту, задаваемому входом **ID**.

Код команды задается входом **CMD**. Возможны следующие значения:

0 - Возвращает число байт, находящихся в буфере на прием и доступных для чтения

1 - Возвращает размер свободного места в буфере на передачу (в байтах)

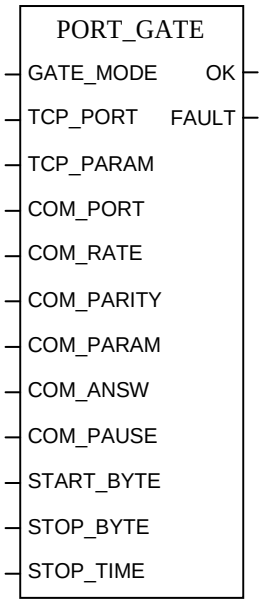
2 - Возвращает статус порта

3 - Очистка входного буфера

4 - Установка линии RTS в состояние, задаваемое через PARAM:
0 - RTS выключено
1 - RTS включено

Вход **PARAM** для функций с кодами 0-3 должен быть сброшен в ноль.

7.3.120 PORT_GATE



Входы:

GATE_MODE	INTEGER	Режим обмена 0 – Трансляция запросов ModbusTCP в ModbusRTU 1 – Трансляция TCP пакетов в COM порт 2 – Трансляция запросов ModbusUDP в ModbusRTU 3 – Трансляция UDP пакетов в COM порт
TCP_PORT	INTEGER	Номер порта TCP/IP
TCP_PARAM	INTEGER	Параметры порта TCP/IP (не используется)
COM_PORT	INTEGER	Номер порта COM
COM_RATE	INTEGER	Скорость обмена порта COM
COM_PARITY	INTEGER	Количество стоповых бит и режим контроля четности порта COM
COM_PARAM	INTEGER	(1-программное управление передатчиком)
COM_ANSW	INTEGER	Интервал времени ожидания ответа (мсек)
COM_PAUSE	INTEGER	Интервал времени перед следующим запросом (мсек)
START_BYTE	INTEGER	Стартовая комбинация байт
STOP_BYTE	INTEGER	Стоповая комбинация байт
STOP_TIME	INTEGER	Интервал времени молчания (между пакетами) (мсек)

Выход:

OK	INTEGER	Признак завершения операции
FAULT	INTEGER	Код завершения операции: 0 – Выполнено успешно; 1 – Ошибка инициализации (некорректные параметры); 2 – Удаленное устройство не отвечает (таймаут истек); 3 – Системная ошибка при работе с портом; 4 – Системная ошибка при обращении к задаче связи 5 – Выполняется запрос 6 – Ошибка принятых данных

Назначение

Открытие портов для транзита TCP\UDP запросов в COM.

Описание:

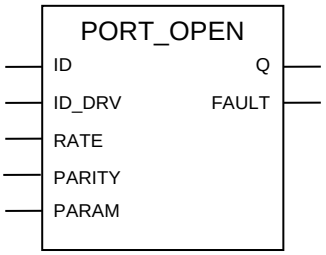
Аргументы **COM_RATE** и **COM_PARITY** используются для конфигурирования последовательной линии. Через аргумент COM_RATE задается скорость обмена:

1200, 1800, 2400, 4800, 7200, 9600, 14400, 19200, 38400, 57600, 115200 бод.

Количество стоповых битов и режим контроля четности задается входом COM_PARITY.
Возможные варианты режима контроля четности представлены в таблице:

Значение	Количество стоповых бит	Контроль четности
0	1	Отключен
1	2	Отключен
2	1	EVEN
3	1	ODD
4	1	SPACE
5	1	MARK

7.3.121 PORT_OPEN



Входы:

ID	Integer	Идентификатор файла порта
ID_DRV	Integer	Номер порта или юнита
RATE	Integer	Скорость обмена
PARITY	Integer	Количество стоповых бит и режим контроля четности
PARAM	Integer	Параметры обмена

Выходы:

Q	Integer	Идентификатор файла порта
FAULT	Integer	Код завершения операции 0 – выполнено успешно 1 – ошибка открытия (некорректные параметры) 3 – системная ошибка при работе с портом

Назначение:
Открытие порта.

Описание:
Функциональный блок открывает последовательный порт, задаваемый входом **ID_DRV**.

При значении входа **ID**, равном нулю, порт открывается с уникальным идентификатором. В противном случае проверяется наличие открытого порта с идентификатором ID. В случае ошибки при открытии файла идентификатор ID равен нулю.

Вход **RATE** может принимать одно из следующих значений:

1200, 1800, 2400, 4800, 7200, 9600, 14400, 19200, 38400, 57600, 115200 бит/с.

Количество бит данных в посылке, стоповых битов и режим контроля четности задается входом **PARITY**.

Таблица 1 – Формат входа PARITY

Байт3	**Байт2**	**Байт1**	**Байт0**	
Резерв, должно быть равно 0			Кол-во бит. Допустимые значения: 5..8; (если не задано, то 8 бит)	См. таблицу 2

Таблица 2 – Режим контроля четности

Значение	**Количество стоповых бит**	**Контроль четности**

0	1	Отключен
1	2	Отключен
2	1	EVEN
3	1	ODD
4	1	SPACE
5	1	MARK

Примеры настройки входа PARITY:
16#00 – кол-во бит 8, стоповых бит 1, контроль четности отключен
16#50 – кол-во бит 5, стоповых бит 1, контроль четности отключен
16#80 – кол-во бит 8, стоповых бит 1, контроль четности отключен
16#73 – кол-во бит 7, стоповых бит 1, контроль четности ODD

Примечание:

Для мастер-модулей M501E/M841E/M902E/M903E/M915E, M991S:

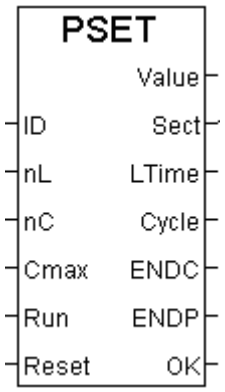
После открытия порта чтение/запись выполняется функциями FL_RD_A / FL_RD_AR и FL_WR_A / FL_WR_AR.

При этом для функции FL_RD_A первый элемент массива содержит количество байт, прочитанных из порта; сами данные располагаются начиная со второго элемента массива. Соответственно, при записи функцией FL_WR_AR в первый элемент массива заносится количество байт для записи в порт.

Для интеллектуального модуля M932C2:

После открытия порта чтение/запись выполняется функциями FL_RD_B, FL_RD_I, FL_RD_R, FL_WR_B, FL_WR_I, FL_WR_R.

7.3.122 PSET



Входы:

ID	#REAL	Имя массива (элементы типа Real)
nL	INTEGER	Количество строк в массиве
nC	INTEGER	Количество столбцов в массиве
CMAX	INTEGER	Число повторений программы (устанавливается при сбросе)
RUN	BOOLEAN	Пуск задатчика
RESET	BOOLEAN	Сброс задатчика

Выходы:

VALUE	REAL	Основной выход задатчика
SECT	INTEGER	Номер текущего участка
TIME	INTEGER	Время, оставшееся до конца текущего участка
CYCLE	INTEGER	Оставшееся число повторений
ENDC	BOOLEAN	Конец текущего повторения программы
ENDP	BOOLEAN	Конец программы
OK	INTEGER	Результат операции

Назначение

Программный задатчик формирует кусочно-линейную функцию времени, состоящую из отрезков. Конечная ордината и продолжительность во времени каждого отрезка задаются в двумерном массиве. Идентификатор массива подается на вход ID. Начальная ордината задается нулевым элементом массива.

Описание алгоритма.

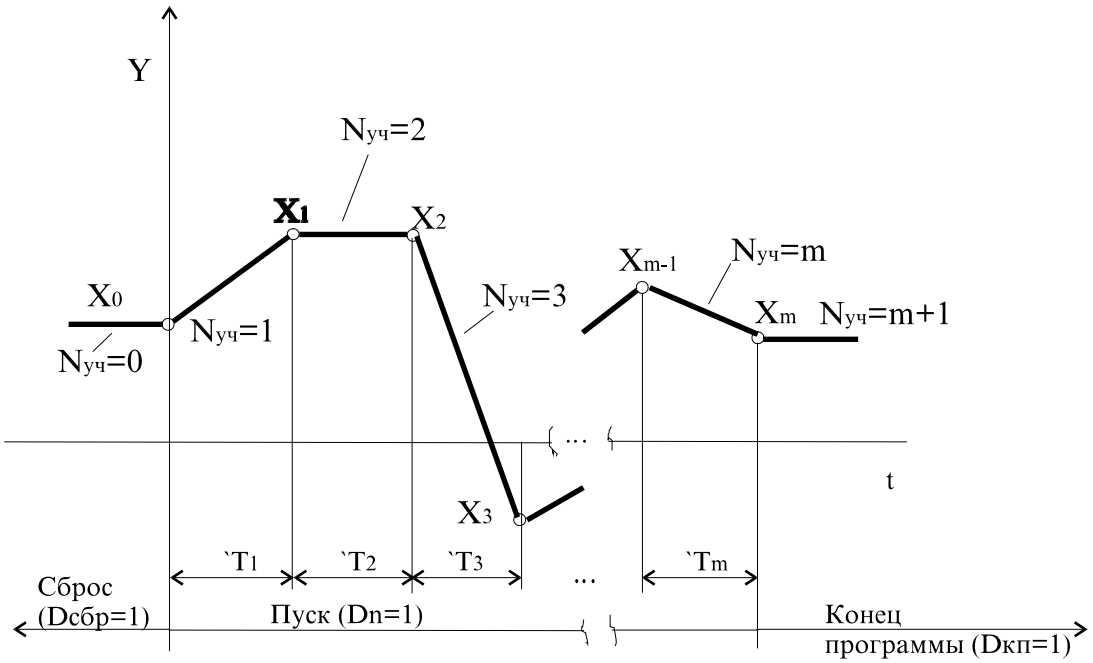
В состоянии сброса значение на выходе VALUE равно начальной ординате. После пуска значение на выходе VALUE начинает изменяться в соответствии с заданной программой.

Вход RESET является приоритетным по отношению к RUN.

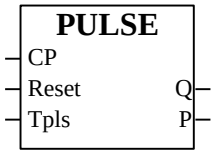
После выполнения последнего участка программа переходит в состояние "конец программы", при этом выходное значение VALUE замораживается. Если задано количество повторов CMAX =1, программа выполняется один раз, после чего переходит в состояние “конец программы”.

При CMAX >1 программа, дойдя до конца, автоматически переходит в начало, оставаясь в состоянии пуска до тех пор, пока не закончится заданное число повторений. После каждого окончания программы выход ENDC (конец повторения) на один цикл переходит в состояние TRUE и вновь возвращается в состояние лог.0. После того, как все повторения будут выполнены, программа переходит в состояние "конец программы". При этом выход ENDP становится равным TRUE и остается в этом состоянии до тех пор, пока программа не будет сброшена.

Сигнал ,формируемый программным задатчиком



7.3.123 PULSE



Входы:

CP	BOOLEAN	Команда "пуск" по фронту
Reset	BOOLEAN	Команда "сброс**"**
Tpls	INTEGER	Длительность импульса (мсек)

Выходы:

Q	BOOLEAN	Основной выход
P	INTEGER	Текущее время импульса (мсек)

Назначение:

Функциональный блок PULSE применяется в тех случаях, когда необходимо сформировать одиночный импульс заданной длительности.

Описание алгоритма:

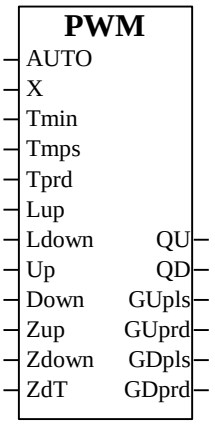
Алгоритм запускается по переднему фронту сигнала на входе CP (пуск), т.е. когда на входе CP дискретный сигнал переходит из состояния лог.0 в состояние лог.1. Перед пуском выходной дискретный сигнал Q отсутствует. После пуска появляется сигнал на выходе Q, причем этот сигнал находится в состоянии лог.1 в течение времени P=Tpls, где Tpls - параметр настройки. По истечении времени Tpls сигнал на выходе вновь переходит в нулевое состояние, после чего алгоритм можно вновь запустить.

На выходе P формируется текущее время, отсчитываемое от момента пуска. После отработки импульса P=0.

Сигнал на входе «Reset» (сброс) в любой момент времени обнуляет оба выхода. При наличии команды "сброс" алгоритм не может быть запущен, а также не может быть повторно запущен командой "пуск" до тех пор, пока не закончится формирование выходного импульса.

Примечание: Для запуска по заднему фронту, сигнал на входе CP инвертируется.

7.3.124 PWM



Входы:

AUTO	BOOLEAN	Режим работы: TRUE -автоматический, FALSE -ручной
X	REAL	Основной вход (скважность импульсов в %)
Tmin	INTEGER	Минимальная длительность импульса (мсек)
Tmps	INTEGER	Минимальная длительность паузы (мсек)
Tprd	INTEGER	Период следования импульсов (мсек)
Lup	INTEGER	Люфт в направлении "больше"(мсек)
Ldown	INTEGER	Люфт в направлении "меньше" (мсек)
Up	BOOLEAN	Команда "больше" в ручном режиме
Down	BOOLEAN	Команда "меньше" в ручном режиме
Zup	BOOLEAN	Сигнал запрета в направлении "Больше"
Zdown	BOOLEAN	Сигнал запрета в направлении "Меньше"
ZdT	BOOLEAN	Запрет накопления минимального импульса

Выходы:

QU	BOOLEAN	Дискретный выход "больше"
QD	BOOLEAN	Дискретный выход "меньше"
GUpls	INTEGER	Длительность импульсов выхода "больше"
GUprd	INTEGER	Период импульсов выхода "больше"
GDpls	INTEGER	Длительность импульсов выхода "меньше"
GDprd	INTEGER	Период импульсов выхода "меньше"

Назначение

Функциональный блок PWM используется для периодического включения и выключения нагрузки в том случае, когда скважность включения должна быть пропорциональна непрерывному управляющему сигналу.

Описание алгоритма.

Блок широтно-импульсной модуляции PWM преобразует значение входного сигнала X в длительность импульса. В автоматическом режиме работы блока (AUTO=TRUE) входной сигнал X преобразуется в последовательность импульсов со скважностью Q, пропорциональной входному сигналу:

$$Q=X/100\%.$$

Период следования импульсов задается на входе Tprd. Реальный период следования импульсов T кратен времени цикла контроллера и отличается от заданного Tprd на время t=0…Tc, где Tc – время одного цикла контроллера:

$$T=Tprd+t.$$

Длительность импульса определяется как произведение периода T и скважности Q:

$$P=T*|Q|,$$

При положительном значении Q импульсы будут формироваться с выхода “больше”, при отрицательном с выхода “меньше”. При абсолютном значении скважности |Q|≥1 выход, соответствующий знаку скважности будет постоянно включен. При значении Q=0 оба выхода находятся в отключенном состоянии.

Если определенная в блоке длительность импульса оказалась меньше Tmin, то длительность устанавливается равной нулю. Причем это значение длительности импульса, который не был выдан в текущем периоде T_i, запоминается и будет добавлено (в соответствии со знаком) к значению длительности импульса в следующем периоде T_{i+1}. При состоянии входа ZdT=TRUE импульс меньше минимального по времени не выдается и далее никак не учитывается, т.е. не добавляется к следующему импульсу.

Если временное расстояние между импульсами меньше заданного параметра Tmps или минимальной длительности импульса Tmin, импульсы слипаются.

Параметры Lup и Ldown задают время дополнительного импульса для выборки люфта исполнительного механизма в направлении соответственно “больше” или “меньше” при изменении направления его движения.

Дискретные входы Zup и Zdown предназначены для блокировки включения исполнительного механизма (ИМ) при достижении крайнего положения. На эти входы подаются либо дискретные сигналы с концевых выключателей ИМ, либо сигналы, сформированные программной логикой. При состоянии Zup=TRUE импульсы с выхода “больше” не формируются, при Zdown=TRUE не формируются импульсы с выхода “меньше”.

При состоянии входа AUTO=FALSE блок работает в ручном режиме, при этом анализируется состояние входов Up и Down. Дискретные входы Up и Down предназначены для ручного управления исполнительным механизмом. При состоянии входа Up=TRUE ИМ перемещается по направлению “больше” (QU=TRUE, GUpls=1000, Guprd=1000), если Down=TRUE по направлению “меньше” (QD=TRUE, GDpls=1000, GDprd=1000). Если Up=Down ИМ не перемещается (все выходы блока равны нулю).

Блок является универсальным в связи с тем, что он может использоваться как для дискретного, так и для импульсного вывода. Дискретные сигналы с выходов блока QU и QD подаются на дискретный вывод, аналоговые сигналы с выходов блока GUpls, GUprd, GDpls и GDprd подаются на импульсный вывод. На дискретных выходах блока Up и Down формируются импульсы, длительность и период следования которых кратны циклу контроллера.

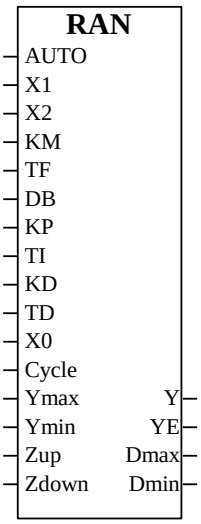
Импульсный вывод работает независимо от цикла работы контроллера, поэтому длительность импульсов с этого вывода не кратна циклу контроллера.

В связи с особенностью реализации импульсного вывода значение длительности импульса (GUpls или GDpls) обновляется один раз в начале каждого периода T. При этом значение на выходе GUprd или GDprd устанавливается равным нулю. Далее до истечения периода T, значение длительности импульса не изменяется, а значение GUprd или GDprd устанавливается равным 100с. Таким образом, реальный период следования импульсов с импульсного вывода будет больше заданного Tprd и кратен циклу контроллера.

Блок PWM обычно работает совместно с импульсным регулятором RIM. Сигнал с выхода импульсного регулятора необходимо подавать на вход X блока PWM. Поскольку длительность импульса определяется один раз за период, блоки RIM и PWM должны работать синхронно, т.е. с одинаковым периодом. Поэтому на вход Cycle блока RIM и на вход Tprd блока PWM необходимо подавать одно и то же значение.

Примечание: блок будет корректно работать только в том случае, если Tmin<Tprd. При переключении на ручной режим работы блока накопленное значение длительности импульса обнуляется вне зависимости от состояния входа ZdT. Если на входе Tprd задать значение Tprd=0, то реальный период следования импульсов T будет равен циклу контроллера. При значении Tprd<0 все выходы блока равны нулю.

7.3.125 RAN



Входы:

AUTO	BOOLEAN	Режим работы: TRUE -автоматический, FALSE -ручной
X1	REAL	Немасштабируемый вход (задание)

X2	REAL	Масштабируемый вход (регулируемый параметр)
KM	REAL	Масштабный коэффициент
TF	REAL	Постоянная времени фильтра (сек)
DB	REAL	Зона нечувствительности
KP	REAL	Коэффициент пропорциональности
TI	REAL	Постоянная времени интегрирования (сек)
KD	REAL	Коэффициент дифференцирования
TD	REAL	Постоянная времени дифференцирования (сек)
X0	REAL	Значение выхода в ручном режиме
Cycle	INTEGER	Период работы блока (мсек)
Ymax	REAL	Ограничение по максимуму
Ymin	REAL	Ограничение по минимуму
Zup	BOOLEAN	Сигнал запрета в направлении "Больше"
Zdown	BOOLEAN	Сигнал запрета в направлении "Меньше"

Выходы:

Y	REAL	Основной выход алгоритма (регулирующее воздействие)
YE	REAL	Сигнал рассогласования
Dmax	BOOLEAN	Признак ограничения по максимуму
Dmin	BOOLEAN	Признак ограничения по минимуму

Назначение

Функциональный блок RAN используется для построения системы ПИД регулирования, имеющей аналоговый выход. Блок, как правило, сочетается с пропорциональным исполнительным механизмом (позиционером), либо используется в качестве ведущего в схеме каскадного регулирования. Помимо формирования ПИД закона в алгоритме вычисляется сигнал рассогласования. Выходной сигнал >граничивается по ма:симуму и минимуму.

Описание алгоритма

Автоматический режим работы регулятора

В автоматическом режиме работы (AUTO=TRUE) функциональный блок RAN формирует регулирующее воздействие по ПИД закону регулирования. Преобразования реализуются в следующей последовательности. Входной сигнал X2 (регулируемый параметр) фильтруется (т.е. проходит через апериодическое звено 1-го порядка) в соответствии с передаточной функцией:

$$W(p)=\frac{1}{TF\cdot p+1}$$,薪где薔TF 딓 поАтоя=ная времени фильтра.

Отфильтрованное значение X2f масштабируется (умножается на коэффициент KM). От значения задания X1 вычитается получившееся после фильтрации и масштабирования значение регулируемого параметра, в результате чего определя5тся значение рассогласования: Xe=X1-KM*X2f. Далее сигнал рассогласования проходит через зону нечувствительности в соотв5тствии со следующим законом:

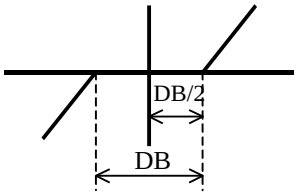
Xz=0, если |Xe|≤DB/2

Xz=Xe-DB/2, если Xe>DB/2

Xz=Xe+DB/2, если Xe<DB/2

где DB – заданная зона нечувствительности.

Прохожд5ние сигнала через зону нечувствительности поясняется следующим рисунком.



Сигнал с выхода зоны нечувств8тельности Xz преобразуется по ПИД з0кону регулирования в соответствии с передаточной функцией:

$$W(p)=KP+\frac{KP}{TI\cdot p}+\frac{KD\cdot TD\cdot p}{TD\cdot p+1},$$

где KP – коэффициент пропорциональности, TI – постоянн0я и=тегрирования, KD – коэффициент дифференцирования, TD – постоянная дифференцирования.

Блок RAN может использоваться в качестве ПД-, ПИ- или П-регулятора. В ПД-регуляторе устанавливается TI=0, при этом интегральная часть обнуляется (KP/TI·p=0). В ПИ-регуляторе устанавливается KD=0, в этом

случае приравнивается к нулю дифференциальная часть регулятора. В П-регуляторе устанавливается TI=0 и KD=0.
Интервал между расчетами регулирующего воздействия не меньше парамеВра Cycle и кратен времени цикла контроллера.
В алгоритме предусмотрено ограничение выходного сигнала Y по нижнему и верхнему пределам. Верхний и нижний пределы задаются входами Ymax и Ymin.
Также предусмотрены дискретные входы Zup (запрет больше) и Zdown (запрет меньше). При Zup=TRUE значения выхода Y и интегральной составляющей не увеличиваются, при Zdown=TRUE не уменьшаются.
Блок имеет 4 выхода. Выход Y - основной выход алгоритма (регулирующее воздействие). На выходе YE формируется сигнал рассогласования, т.е. YE=Xe. Два дискретных выхода Dmax и Dmin фиксируют момент наступления ограничения выходного сигнала Y. Логика формирования выходных дискретных сигналов определяется следующей таблицей (здесь Y1 - сигнал на входе звена ограничения):

Y1	Y	Dmax	Dmin
Ymin<Y1<Ymax	Y=Y1	0	0
Y1≥Ymax	Y=Ymax	1	0
Y≤Ymin	Y=Ymin	0	1

Алгоритм будет правильно работать, только если Ymax>Ymin. Если действует ограничение по максимуму (Dmax=TRUE), значение интегральной части регулятора не увеличивается. Если действует ограничение по минимуму (Dmin=TRUE), значение интеграла не уменьшается.

Ручной режим работы регулятора

При ручном режиме работы (AUTO=FALSE) основной выход блока равен входу X0, т.е. Y=X0. Входной сигнал X2 при ручном режиме работы не фильтруется, поэтому выход YE=Xe=X1-KM·X2. Ограничения и запреты на выход Y действуют также как и при автоматическом режиме работы.
В момент переключения регулятора на автоматический режим работы (переход входного сигнала AUTO из состояния FALSE в состояние TRUE) интегральная часть регулятора приводится к значению IV=X0-KP·Xz, дифференциальная часть регулятора обнуляется (т. е. производная приравнивается к нулю), значение выхода Y не изменяется. Таким образом, переход на автоматический режим выполняется безударно.

***Примечания*:** При значении Cycle≤0 период работы блока равен циклу контроллера. Отрицательные значения параметров KP, TI, KD, TD воспринимаются как нулевые значения. При значении TF≤0 входной сигнал X2 не фильтруется. При значении DB≤0 Xz=Xe. Параметры настройки регулятора (DB, KP, TI, KD, TD) рекомендуется изменять при ручном режиме работы регулятора.

7.3.126 RF_TRIG

Вход:

CLK	BOOLEAN	Основной вход

Выход:

Q	BOOLEAN	Основной выход

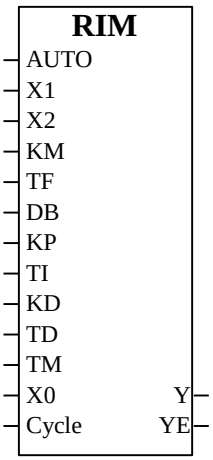
Назначение

Функциональный блок RF_TRIG применяется для выделения переднего фронта дискретного сигнала.

Описание алгоритма

Если на входе блока дискретный сигнал CLK изменяет свое состояние с логического 0 на логическую 1 (передний фронт), то на выходе алгоритма формируется сигнал Q=1 на время одного цикла работы контроллера. Остальное время Q=0.

7.3.127 RIM



Входы:

AUTO	BOOLEAN	Режим работы: TRUE -автоматический, FALSE -ручной
X1	REAL	Немасштабируемый вход (задание)
X2	REAL	Масштабируемый вход (регулируемый параметр)
KM	REAL	Масштабный коэффициент
TF	REAL	Постоянная времени фильтра (сек)
DB	REAL	Зона нечувствительности
KP	REAL	Коэффициент пропорциональности
TI	REAL	Постоянная времени интегрирования (сек)
KD	REAL	Коэффициент дифференцирования
TD	REAL	Постоянная времени дифференцирования (сек)
TM	REAL	Полное время хода ИМ (сек)
X0	REAL	Значение выхода регулятора в ручном режиме
Cycle	INTEGER	Период работы блока (мсек)

Выходы:

Y	REAL	Основной выход (скважность импульсов в %)
YE	REAL	Сигнал рассогласования

Назначение

Функциональный блок RIM используется при построении системы ПИД регулирования, работающей совместно с исполнительным механизмом постоянной скорости. Блок, как правило, применяется в сочетании с алгоритмом широтно-импульсной модуляции (PWM), который преобразует выходной аналоговый сигнал алгоритма RIM в последовательность импульсов для дискретного или импульсного вывода. Помимо формирования закона регулирования в алгоритме вычисляется сигнал рассогласования.

Описание алгоритма

Автоматический режим работы регулятора

В автоматическом режиме работы (AUTO=TRUE) функциональный блок RIM формирует регулирующее воздействие по ПДД^2 закону регулирования. Преобразования входных сигналов в блоке RIM аналогичны преобразованиям в блоке RAN, т.е. входной сигнал X2 (регулируемый параметр) фильтруется и умножается на коэффициент масштабирования KM, получившееся значение вычитается из задания X1, в результате чего определяется сигнал рассогласования Xe, который проходит через зону нечувствительности. Сигнал с выхода зоны нечувствительности Xz преобразуется по ПДД^2 закону в соответствии с передаточной функцией:

$$W(p) = TM \cdot (KP \cdot p + \frac{KP}{TI} + \frac{KD \cdot TD \cdot p^2}{(TD \cdot p + 1)^2}),$$

что позволяет совместно с исполнительным механизмом (ИМ) постоянной скорости имеющим передаточную функцию Wим(p) = 1/(TM·p) приближенно реализовать ПИД преобразование сигнала Xz в соответствии с передаточной функцией:

$$W(p) = KP + \frac{KP}{TI \cdot p} + \frac{KD \cdot TD \cdot p}{(TD \cdot p + 1)^2},$$

где TM – полное время хода ИМ, KP – коэффициент пропорциональности, TI – постоянная интегрирования, KD – коэффициент дифференцирования, TD – постоянная дифференцирования.

Интервал между расчетами регулирующего воздействия не менее параметра Cycle и кратен времени цикла контроллера.

Блок имеет 2 выхода. Выход Y - основной выход алгоритма (регулирующее воздействие). На выходе YE формируется сигнал рассогласования, т.е. YE=Xe.

Ручной режим работы регулятора

При ручном режиме работы (AUTO=FALSE) выход бока Y=X0. Входной сигнал X2 не фильтруется, поэтому YE=Xe=X1-KM·X2.

В момент переключения регулятора на автоматический режим работы (переход входного сигнала AUTO из состояния FALSE в состояние TRUE) первая и вторая производная приравниваются к нулю, выход регулятора равен нулю. Таким образом, переход на автоматический режим выполняется безударно.

Примечание: При значении Cycle≤0 период работы блока равен циклу контроллера. Отрицательные значения параметров KP, KD, TD воспринимаются как нулевые. При значении TI≤0 TI=1. При значении TF≤0 входной сигнал X2 не фильтруется. При значении DB≤0 Xz=Xe. Параметры настройки регулятора (DB, KP, TI, KD, TD) рекомендуется изменять при ручном режиме работы регулятора.

7.3.128 RS

Входы:

SET	BOOLEAN	Если TRUE, устанавливает выход в TRUE
RESET	BOOLEAN	Если TRUE, сбрасывает выход в FALSE (доминанта)

Выход:

Q	BOOLEAN	Основной выход

Назначение

Функциональный блок RS применяется для запоминания дискретных сигналов.

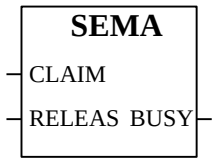
Описание алгоритма

Функциональный блок RS реализует доминанту сброса. Выходной текущий дискретный сигнал Q_i в зависимости от состояния входных дискретных сигналов SET и RESET, и с учетом доминанты сброса устанавливается в следующее состояние:

SET	RESET	Q_{i-1}	Q_i
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	0

Q_{i-1} – состояние выхода в предыдущем цикле;
Q_i – состояние выхода в текущем цикле.

7.3.129 SEMA



Входы:

CLAIM	BOOLEAN	команда “проверить и установить”
RELEASE	BOOLEAN	освободить семафор

Выход:

BUSY	BOOLEAN	состояние семафора

Описание:
Функциональный блок SEMA применяется в качестве семафора
Если CLAIM = TRUE, то выход устанавливается в TRUE.
Если CLAIM = FALSE, а RELEASE = TRUE, то выход устанавливается в FALSE.

7.3.130 SEND_SMS

Входы:

RUN	Boolean	Признак выполнения функционального блока(передача запроса в модем)
AbNum	Message	Номер абонента - получателя сообщения
Message	Message	Текст сообщения передаваемого сообщения

Выходы:

READY	Boolean	Признак завершения операции.
FAULT	Integer	Код ошибки завершения операции.

Назначение

Функциональный блок используется для передачи СМС сообщений через встроенный модем в мастер-модуле M1012E
Запуск передачи происходит по переднему фронту на входе RUN.
После этого READY устанавливается в FALSE, и до завершения передачи вход RUN блоком не опрашивается.

Примечание
Блок рекомендуется использовать совместно с оператором SYSTEM(20,3),
который возвращает состояние передачи СМС:

0: ожидание команды блоком
1: подготовка к передаче команды модему
2: команда передается модему
3: ожидание подтверждения от модема
4: подтверждение от модема получено
5: передача текста в модем
6: ожидание завершения передачи текста
7: подтверждение от модема получено
8: СМС послана
(Здесь устанавливается выход блока "READY" в TRUE, и можно передавать новое СМС)
9: ожидание подтверждения доставки от оператора
10: подтверждения доставки от оператора получено

Блок SEND_SMC и оператор SYSTEM(20,3) предназначен для использования в мастер-модуле M1012E со встроенным GSM модемом.

Корректно передаются сообщения, содержащие только латинские буквы.

Пример на ST:

(* при установке osms_strob в TRUE происходит отправка сообщения*)

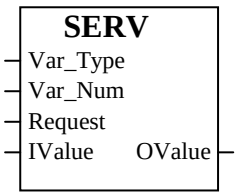
```
osms_status:=system(20,3);

if (osms_strob = true) and (o_sms_run = false) then
  if (osms_status = 0) or (osms_status > 8) then
    o_sms_run:=true;
    o_sms_msg:='TREI test sms';           // текст отправляемого сообщения
    o_sms_num:='+79271234567';               // номер абонента которому отправляется
сообщение
  end_if;
end_if;

o_sms(o_sms_run, o_sms_num, o_sms_msg);          // o_sms - блок типа SEND_SMS
o_sms_ok:=o_sms.ok;
o_sms_fault:=o_sms.fault;

if o_sms_run = true then
   if (o_sms_ok = true) or (o_sms_fault <> 0) then
      o_sms_run:=false;                          //  отправка сообщения завершена
      osms_strob:=false;
   end_if;
end_if;
```

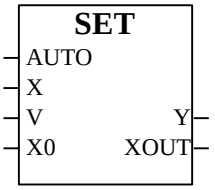
7.3.131 SERV



Назначение:
Блок поверки канала.

Примечание:
Для внутреннего использования отладчиком.
Нельзя использовать в программах.

7.3.132 SET



Входы:

AUTO	BOOLEAN	Режим работы; TRUE -автоматический, FALSE -ручной
X	REAL	Задание
V	REAL	Скорость балансировки (1/сек)
X0	REAL	Значение выхода блока в ручном режиме

Выход:

Y	REAL	Задание для регулятора с динамической балансировкой
XOUT	REAL	Выход без динамической балансировки

Назначение

Функциональный блок SET предназначен для формирования задания для регулятора.

Описание алгоритма

В автоматическом режиме работы блока (AUTO=TRUE) входной сигнал X проходит через звено динамической балансировки. Для этого определяется скорость изменения значения входа X относительно выхода:

$$V1 = \frac{X - Y_{i-1}}{Tc},$$

где Y_{i-1} – значение выхода в предыдущем цикле, Tc – время цикла контроллера.

Если |V1|≤V, то сигнал X проходит на выход без изменений. Если |V1|>V, то выход Y будет изменяться в соответствии со следующей зависимостью:

$$Y_i = Y_{i-1} + sign(V1) \cdot V \cdot Tc,$$

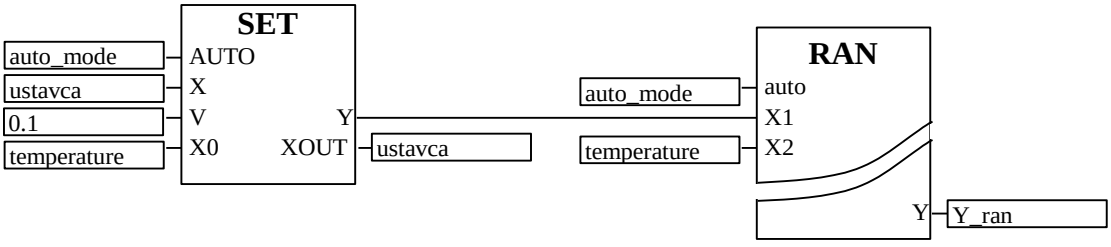
где Y_i – значение выхода в текущем цикле, V – скорость динамической балансировки.

Выход Y будет изменяться с ограниченной скоростью V до тех пор, пока не достигнет значения X.

При отключенной динамической балансировке (V≤0) входной сигнал X проходит на выход Y без изменений. На выход XOUT в автоматическом режиме работы проходит без изменений сигнал с входа X, минуя звено динамической балансировки. В ручном режиме работы (AUTO=FALSE) выходы блока равны входу X0.

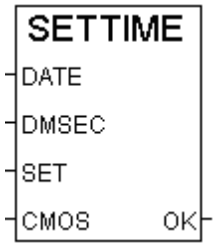
Ниже на рисунке приведен пример соединения блоков SET и RAN. Сигнал с выхода Y блока SET подается на вход X1 (задание) регулятора. На вход X0 блока SET подается тот же сигнал, что и на вход X2 регулятора (регулируемый параметр).

При ручном режиме работы данной схемы ввод нового значения задания (ustavca) игнорируется, т.е. задание отслеживает регулируемый параметр. При переключении на автоматический режим, задание перестает отслеживать регулируемый параметр и равно последнему значению регулируемого параметра в ручном режиме, изменить значение задания можно в следующем цикле работы контроллера. Если выход XOUT не задействовать (т.е. не подключать переменную ustavca к выходу XOUT), значение задания можно изменять при любом режиме работы.



Пример соединения блоков SET и RAN.

7.3.133 SETTIME



Входы:

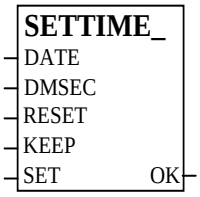
DATE	INTEGER	Текущая дата (день, месяц, год)
DMSEC	INTEGER	Миллисекунда с начала суток
SET	BOOLEAN	Установить (по фронту)
CMOS	BOOLEAN	Признак записи времени в CMOS

Выход:

OK	INTEGER	Результат операции

Назначение
Установка текущего времени

7.3.134 SETTIME_



Входы:

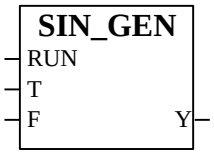
DATE	INTEGER	Текущая дата (день, месяц, год)
DMSEC	INTEGER	Миллисекунда с начала суток
RESET	BOOLEAN	Сбросить поправку времени
KEEP	BOOLEAN	Запомнить текущее системное время (по фронту)
SET	BOOLEAN	Вычислить поправку времени (по фронту)

Выход:

OK	INTEGER	Результат операции

Назначение
Установка текущего времени (специальная) с использованием поправки времени

7.3.135 SIN_GEN



Входы:

RUN	BOOLEAN	Режим работы
T	REAL	Период колебаний (сек)
F	REAL	Фаза колебаний (рад)

Выход:

Y	REAL	Основной выход

Назначение

Функциональный блок SIN_GEN применяется для генерации синусоидальных сигналов при моделировании и настройке контуров регулирования.

Описание алгоритма

Функциональный блок содержит генератор синусоидального сигнала с периодом T и фазой F. Фаза задается в радианах, F=0 – 2π.

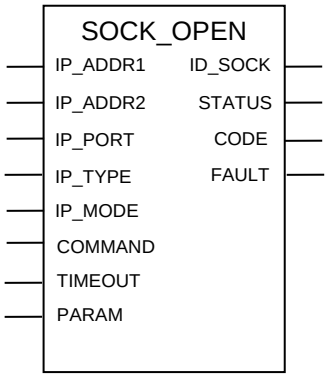
На выходе Y формируется сигнал по закону

$$Y = \sin(\omega \cdot t + \varphi) = \sin(\frac{2 \cdot \pi \cdot t}{T} + F),$$

Текущее время t изменяется дискретно по закону t=n*Tc, где Tc - время цикла контроллера. Период выходного сигнала T приводится к значению, кратному Tc.

Генерация выходного сигнала начинается при установке входного сигнала RUN в состояние TRUE. При состоянии RUN=FALSE генерация синусоидального сигнала прекращается, выходной сигнал принимает значение Y=0.

7.3.136 SOCK_OPEN



Входы:

IP_ADDR1	Integer	IP адрес абонента в основной сети (выход функции TO_IP_ADDR)
IP_ADDR2	Integer	IP адрес абонента в резервной сети (выход функции TO_IP_ADDR)
IP_PORT	Integer	IP порт абонента
IP_TYPE	Integer	Тип сокета: 0-UDP, 1-TCP
IP_MODE	Integer	Режим обмена: 0-Master, 1-Slave
COMMAND	Integer	Команда: 0-Сброс, 1-Открыть
TIMEOUT	Integer	Таймаут ожидания готовности
PARAM	Integer	Параметры (зарезервировано)

Выходы:

ID_SOCK	Integer	Идентификатор сокета
STATUS	Integer	Код статуса сокета 0 – сокет не открыт 1 – сокет открывается 2 – выполняется\контролируется соединение 3 – режим UDP
CODE	Integer	Код статуса соединения (служебный)
FAULT	Integer	Код завершения операции 0 – выполнено успешно 3 – ошибка открытия\соединения 1 – обрыв линии

Назначение:
Открытие сокета IP.

Описание:

Функциональный блок открывает IP сокет при задании IP адреса и IP порта для удаленного абонента.

В случае ошибки при открытии сокета идентификатор ID равен "-1".

Примечание:

Для мастер-модулей M841E/M902E/M915E:

После открытия сокета чтение/запись может выполняться функциями FL_RD_A и FL_WR_A.

При этом первый элемент массива содержит количество байт, прочитанных из порта.

Собственно данные располагаются начиная со второго элемента массива.

Соответственно, при записи, в первый элемент массива заносится количество байт для записи в сокет.

Также могут использоваться и другие функции с префиксом "FL_" для чтения\записи данных.

Для определения количества принятых данных, доступных для считывания, должен использоваться функциональный блок PORT_CTRL.

7.3.137 SR

Входы:

SET	BOOLEAN	Если TRUE, устанавливает выход в TRUE (доминанта)
RESET	BOOLEAN	Если TRUE, сбрасывает выход в FALSE

Выход:

Q	BOOLEAN	Основной выход

Назначение

Функциональный блок SR применяется для запоминания дискретных сигналов.

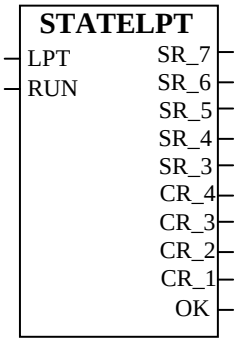
Описание алгоритма

Функциональный блок SR реализует доминанту установки. Выходной текущий дискретный сигнал Q_i в зависимости от состояния входных дискретных сигналов SET и RESET, и с учетом доминанты установки устанавливается в следующее состояние:

SET	RESET	Q_{i-1}	Q_i
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

Q_{i-1} – состояние выхода в предыдущем цикле;
Q_i – состояние выхода в текущем цикле.

7.3.138 STATELPT



Входы:

LPT	INTEGER	Номер порта LPT
RUN	BOOLEAN	Режим работы: TRUE – выполнить, FALSE – не выполнять

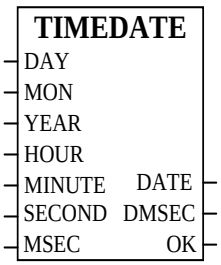
Выход:

SR_7	BOOLEAN	Инверсное состояние линии Busy
SR_6	BOOLEAN	Состояние линии Asc (Acknowledge)
SR_5	BOOLEAN	Состояние линии Paper End
SR_4	BOOLEAN	Состояние линии Select
SR_3	BOOLEAN	Состояние линии Error
CR_4	BOOLEAN	Состояние линии ACKINTEN (Ack Interrupt Enable)
CR_3	BOOLEAN	Состояние линии SelectIn
CR_2	BOOLEAN	Состояние линии Init
CR_1	BOOLEAN	Состояние линии Auto LF
OK	INTEGER	Результат операции (0 – выполнено успешно)

Назначение:
Чтения регистра состояния и регистра управления принтера

Примечание:
Для внутреннего использования отладчиком.
Нельзя использовать в программах.

7.3.139 TIMEDATE



Входы:

DAY	INTEGER	День с начала месяца
MON	INTEGER	Месяц с начала года
YEAR	INTEGER	Год
HOUR	INTEGER	Час с начала суток
MIN	INTEGER	Минута с начала суток
SEC	INTEGER	Секунда с начала минуты
MSEC	INTEGER	Миллисекунда с начала секунды

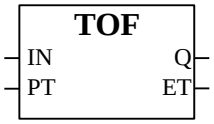
Выходы:

DATE	INTEGER	Дата (день, месяц, год)

DMSEC	INTEGER	Миллисекунда с начала суток
OK	INTEGER	Результат операции (0 - выполнено успешно)

Назначение
Преобразование времени из развернутого в упакованный формат.

7.3.140 TOF



Входы:

IN	BOOLEAN	TRUE - запустить таймер, FALSE - сбросить таймер
PT	INTEGER	Максимальное время отсчета (мсек)

Выходы:

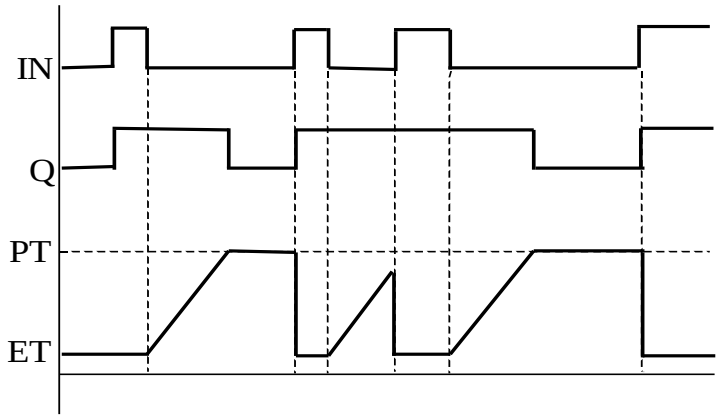
Q	BOOLEAN	TRUE - если время PT не истекло
ET	INTEGER	Текущее время, отсчитываемое таймером

Назначение
Функциональный блок TOF используется для задания временных задержек.

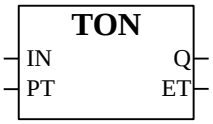
Описание алгоритма
Функциональный блок содержит внутренний таймер, который запускается по заднему фронту входного дискретного сигнала IN. На выходе ET формируется текущее время, отсчитываемое таймером от момента пуска. Если текущее время, отсчитываемое таймером, сравняется со значением входа PT, отсчет времени прекращается. При этом выход Q=0.

Текущее время	Выход Q
ET<PT	Q=1
ET≥PT	Q=0

По переднему фронту сигнала IN отсчет времени прекращается и таймер сбрасывается, выход ET обнуляется, выход Q=1. Работу блока поясняет временная диаграмма.



7.3.141 TON



Входы:

IN	BOOLEAN	TRUE - запустить таймер, FALSE - сбросить таймер
PT	INTEGER	Максимальное время отсчета (мсек)

Выходы:

Q	BOOLEAN	TRUE - если время PT истекло
ET	INTEGER	Текущее время, отсчитываемое таймером

Назначение

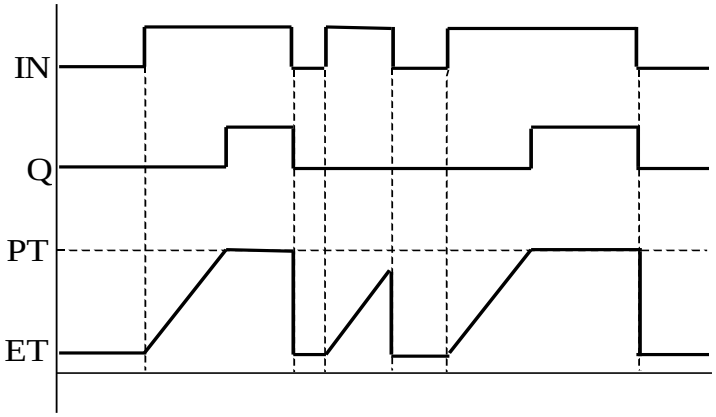
Функциональный блок TON используется для задания временных задержек.

Описание алгоритма

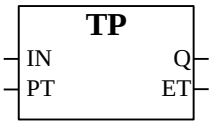
Функциональный блок содержит внутренний таймер, который запускается по переднему фронту входного дискретного сигнала IN. На выходе ET формируется текущее время, отсчитываемое таймером от момента пуска. Если текущее время, отсчитываемое таймером, сравняется со значением входа PT, отсчет времени прекращается. При этом устанавливается в 1 выход Q.

Текущее время	Выход Q
ET<PT	Q=0
ET≥PT	Q=1

По заднему фронту сигнала IN отсчет времени прекращается и таймер сбрасывается, выход ET обнуляется. Работу блока поясняет временная диаграмма.



7.3.142 TP



Входы:

PT	INTEGER	Максимальное время отсчета (мсек)
IN	BOOLEAN	TRUE - запустить таймер, FALSE - сбросить таймер, если время PT истекло

Выходы:

Q	BOOLEAN	TRUE - если таймер считает

ET	INTEGER	Текущее время, отсчитываемое таймером

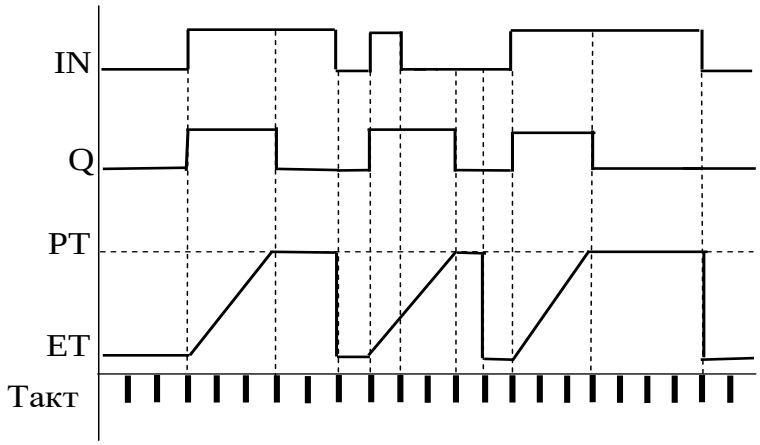
Назначение

Функциональный блок TP используется для задания временных задержек.

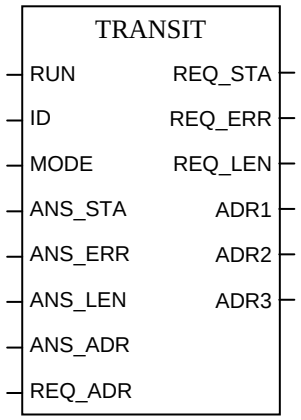
Описание алгоритма

Функциональный блок содержит внутренний таймер, который запускается по переднему фронту входного дискретного сигнала IN. На выходе ET формируется текущее время, отсчитываемое таймером от момента пуска. Во время счета алгоритм не реагирует на изменения сигнала IN. Если текущее время, отсчитываемое таймером, сравняется со значением входа PT, отсчет времени прекращается, таймер останавливается. Сброс таймера происходит по заднему фронту сигнала IN, но только в том случае, если отсчитываемое таймером время достигло значения PT.

Выход Q устанавливается в 1, когда таймер начинает считать. Если таймер остановлен, выход Q равен логическому нулю. Работу блока поясняет временная диаграмма.



7.3.143 TRANSIT



Входы:

RUN	BOOLEAN	Запуск функционального блока
ID	INTEGER	Идентификатор задачи связи
MODE	INTEGER	Протокол (0 - HART, 1 - Modbus)
ANS_STA	INTEGER	Статус ответного пакета
ANS_ERR	INTEGER	Код ошибки ответного пакета
ANS_LEN	INTEGER	Длина ответного пакета (в байтах)
ANS_ADR	INTEGER	Начальный Modbus-адрес данных в базе для ответного пакета
ANS_ADR	INTEGER	Начальный Modbus-адрес данных в базе для запросного пакета

Выход:

REQ_STA	INTEGER	Статус запросного пакета
REQ_ERR	INTEGER	Код ошибки запросного пакета

REQ_LEN	INTEGER	Длина запросного пакета (в байтах)
ADR1	INTEGER	Адрес1
ADR2	INTEGER	Адрес2
ADR3	INTEGER	Адрес3

Назначение

Организация транзитного канала для обмена с произвольными устройствами через мастер-модуль.

Описание

Для организации транзитного канала с использованием функционального блока TRANSIT необходимо следующее:

1) На PC с WindowsXP/7 должен быть запущен соответствующий драйвер виртуального COM порта
2) В исполнительной системе UnimodPRO на мастер модуле должна быть запущена задача "transit"
3) В составе контроллера должен быть соответствующий коммуникационный модуль. Например, для HART протокола – M941A. На коммуникационном модуле должна присутсвовать программа с вызовом функционального блока, соответствующего протоколу, например HART_TRANSIT.

Запрос и ответ помещаются непосредственно в базу UnimodPRO. Для этого при вызове блока указываются начальные Modbus-адреса переменных, содержащих запрос и ответ. Переменные должна располагаться на карте адресов Modbus без пустот, с учетом того, что переменные имеют целый тип (4 байта) и занимают 2 смежных регистра Modbus.

Данные запроса и ответа упаковываются в целые (4 байта) переменные, и их количество должно соответствовать протоколу обмена (режим работы - **MODE**). Например, для HART протокола необходимо 70 целых переменных (255 байт данных + сервисные данные).

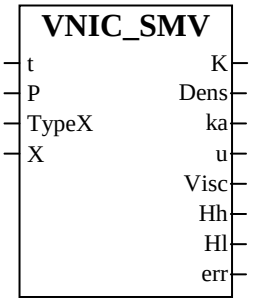
Параметр **ID** используется для логического связывания функционального блока с параметрами виртуального порта, задаваемыми в конфигураторе ***"TREI VCom Configurator"***.

Параметры **Адрес1 - Адрес3** задаются в конфигураторе ***"TREI VCom Configurator"*** и могут быть использованы в технологическом приложении для перенаправления запросов, приходящих через виртуальный порт.

Адрес1 - Адрес3 имеют следующие значения:
Для HART:
- Адрес1 - номер коммуникационного модуля
- Адрес2 - не используется
- Адрес3 - номер канала

Для Modbus:
- Адрес1 - номер COM-порта на мастер-модуле, в который перенаправлять запросы.

7.3.144 VNIC_SMV



Входы:

t	REAL	Температура газа (град.С)
P	REAL	Давление газа (кПа)
TypeX	BOOLEAN	Тип компонентного состава газа: FALSE – в молярных %; TRUE – в объемных %.
X	#REAL	Ссылка на массив с компонентным составом газа (элементы типа Real)

Выходы:

K	REAL	Коэффициент сжимаемости
Dens	REAL	Плотность газа при рабочих условиях (кг / куб.м)
ka	REAL	Показатель адиабаты
u	REAL	Скорость звука в газе (м / с)
Visc	REAL	Динамическая вязкость (мкПа * с)
Hh	REAL	Высшая удельная теплота сгорания (МДж / куб.м)
Hl	REAL	Низшая удельная теплота сгорания (МДж / куб.м)
err	INTEGER	Код ошибки: 0 – нет ошибки; 3 – недопустимый размер массива; 4 – недопустимая ссылка на массив; -1 – недопустимое значение температуры; -2 – недопустимое значение давления; -8 – недопустимое значение доли азота; -16 – недопустимое значение доли углекислого газа; -32 – недопустимое значение доли метана; -64 – недопустимое значение доли этана; -128 – недопустимое значение доли пропана; -256 – недопустимое значение доли бутана; -512 – недопустимое значение доли сероводорода; -1024 – недопустимое значение доли остальных компонентов.

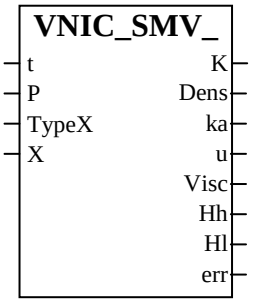
Назначение

Функциональный блок используется для расчета свойств природного газа по методу ВНИЦ СМВ (ГОСТ 30319.2-96).

Вход X содержит концентрации компонентов газа в следующем порядке:

X[1]	Метан
X[2]	Этан
X[3]	Пропан
X[4]	н-Бутан
X[5]	и-Бутан
X[6]	Азот
X[7]	Диоксид углерода
X[8]	Сероводород
X[9]	Ацетилен
X[10]	Этилен
X[11]	Пропилен
X[12]	н-Пентан
X[13]	и-Пентан
X[14]	н-Гексан
X[15]	Бензол
X[16]	н-Гептан
X[17]	Толуол
X[18]	н-Октан
X[19]	Гелий
X[20]	Водород
X[21]	Монооксид углерода
X[22]	Кислород

7.3.145 VNIC_SMV_



Входы:

t	REAL	Температура газа (град.С)
P	REAL	Давление газа (кПа)
TypeX	BOOLEAN	Тип компонентного состава газа: FALSE – в молярных %; TRUE – в объемных %.
X	#REAL	Ссылка на массив с компонентным составом газа (элементы типа Real)

Выходы:

K	REAL	Коэффициент сжимаемости
Dens	REAL	Плотность газа при рабочих условиях (кг / куб.м)
ka	REAL	Показатель адиабаты
u	REAL	Скорость звука в газе (м / с)
Visc	REAL	Динамическая вязкость (мкПа * с)
Hh	REAL	Высшая удельная теплота сгорания (МДж / куб.м)
Hl	REAL	Низшая удельная теплота сгорания (МДж / куб.м)
err	INTEGER	Код ошибки: 0 – нет ошибки; 3 – недопустимый размер массива; 4 – недопустимая ссылка на массив; -1 – недопустимое значение температуры; -2 – недопустимое значение давления; -8 – недопустимое значение доли азота; -16 – недопустимое значение доли углекислого газа; -32 – недопустимое значение доли метана; -64 – недопустимое значение доли этана; -128 – недопустимое значение доли пропана; -256 – недопустимое значение доли бутана; -512 – недопустимое значение доли сероводорода; -1024 – недопустимое значение доли остальных компонентов.

Назначение

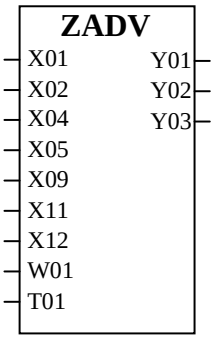
Функциональный блок используется для расчета свойств природного газа по методу ВНИЦ СМВ (ГОСТ 30319.2-96).

Вход X содержит концентрации компонентов газа в следующем порядке:

X[1]	Метан
X[2]	Этан
X[3]	Пропан
X[4]	н-Бутан
X[5]	и-Бутан
X[6]	н-Пентан
X[7]	и-Пентан
X[8]	н-Гексан
X[9]	н-Гептан
X[10]	н-Октан
X[11]	Ацетилен
X[12]	Этилен

X[13]	Пропилен
X[14]	Бензол
X[15]	Толуол
X[16]	Водород
X[17]	Водяной пар
X[18]	Аммиак
X[19]	Метанол
X[20]	Сероводород
X[21]	Метилмеркаптан
X[22]	Диоксид серы
X[23]	Гелий
X[24]	Неон
X[25]	Аргон
X[26]	Монооксид углерода
X[27]	Азот
X[28]	Воздух
X[29]	Кислород
X[30]	Диоксид углерода

7.3.146 ZADV



Входы:

X01	BOOLEAN	КВ "НЕ ОТКРЫТА"
X02	BOOLEAN	КВ "НЕ ЗАКРЫТА"
X04	BOOLEAN	КВ "ХОД НА ЗАКРЫТИЕ"
X05	BOOLEAN	КВ "ХОД НА ОТКРЫТИЕ"
X09	BOOLEAN	Команда "Работа/Ремонт"
X11	BOOLEAN	Команда "ЗАКРЫТЬ" от оператора
X12	BOOLEAN	Команда "ОТКРЫТЬ" от оператора
W01	INTEGER	Слово состояния (Квитирование)
T01	REAL	Полное время хода, с

Выходы:

Y01	BOOLEAN	Команда "ЗАКРЫТЬ"
Y02	BOOLEAN	Команда "ОТКРЫТЬ
Y03	INTEGER	Слово состояния

Примечание

Все дискретные команды – потенциальные. Активный уровень – 1 (TRUE).

Назначение

Функциональный блок используется для управления исполнительными механизмами типа "Задвижка с промежуточным остановом".

На входы X01-X05 подаются сигналы со схемы управления задвижкой.

Вывод механизма в ремонтный режим осуществляется через входной сигнал X09=TRUE. При этом устанавливается запрет выдачи команд управления (открытия/закрытия) в схему управления задвижкой.

На входы X11, X12 подаются потенциальные дискретные команды от оператора (блок управления предназначен для управления задвижкой без самоподхвата).

Алгоритм выявления неисправностей формирует следующие признаки неисправностей:

- нет питания схемы управления (Y03=20)

Признак формируется, если X01=FALSE и X02=FALSE.

- отсутствие подтверждения хода (Y03=10)

Признак формируется, если была выдана команда управления Y01=TRUE, и в течение 1 с не появился сигнал X04=TRUE, или была выдана команда управления Y02=TRUE и в течение 1 с не появился сигнал X05=TRUE.

- превышение времени хода (Y03=12)

Признак формируется, если выдана команда управления Y01=TRUE и в течение времени T01 не появился сигнал X01=TRUE и X02=FALSE или выдана команда управления Y02=TRUE и в течение времени T01 не появился сигнал X02=TRUE и X01=FALSE.

Квитирование признака неисправности осуществляется следующим образом: на вход W01 подается значение, равное Y03 + 1.

Алгоритм выдачи команд управления формирует потенциальные команды с контроллера в электрическую схему управления с учетом признаков неисправности и текущего положения арматуры. Если сформирован один или несколько признаков неисправности, то поданные команды снимаются.

Формируются следующие состояния, в которых может находиться задвижка:

2 – задвижка открыта;
4 – задвижка закрыта;
6 – задвижка открывается;
8 – задвижка закрывается;
10 – нет подтверждения хода;
12 – превышено время хода;
20 – схема разобрана;
22 – задвижка в промежуточном положении;
24 – задвижка выведена в ремонт.

7.4 Функции

В таблице приведены описанные в библиотеке Unimod Pro функции и их реализация на различных типах модулей.

Код	Название	Описание	Мастер-ПК	M841E M921E M902E M903E M915E	M911	M900/M800
	Математические					
21	ABS	Модуль вещественного числа	●	●	●	●
31	EXPT	Экспонента	●	●	●	●
22	POW	Степень	●	●	●	●
23	LOG	Логарифм	●	●	●	●
30	SQRT	Корень квадратный	●	●	●	●
32	TRUNC	Отрезает действительную часть	●	●	●	●
100	ABS_INT	Модуль целого числа		●		
	Тригонометрические					
27	ACOS	Арккосинус	●	●	●	●
28	ASIN	Арксинус	●	●	●	●
29	ATAN	Арктангенс	●	●	●	●
24	COS	Косинус	●	●	●	●
25	SIN	Синус	●	●	●	●
26	TAN	Тангенс	●	●	●	●
	Работа с регистрами					
1	ROL	Вращать влево	●	●	●	●
2	ROR	Вращать вправо	●	●	●	●
3	SHL	Сдвиг влево	●	●	●	●
4	SHR	Сдвиг вправо	●	●	●	●
	Работа с данными					
11	MIN	Минимум	●	●	●	●
12	MAX	Максимум	●	●	●	●
10	LIMIT	Ограничение	●	●	●	●
19	MOD	Модуль	●	●	●	●
13	MUX4	Мультиплексор целочисленный (4 входа)	●	●	●	●
15	MUX8	Мультиплексор целочисленный (8 входов)	●	●	●	●
14	MUXR4	Мультиплексор действительный (4 входа)	●	●	●	●
16	MUXR8	Мультиплексор действительный (8 входов)	●	●	●	●
17	SEL	Двоичный целочисленный селектор	●	●	●	●
18	SELR	Переключатель на 2 вещественных входа	●	●	●	●
20	ODD	Четность	●	●	●	●
33	RAND	Случайная величина	●	●	●	●
81	CHECK_FLOAT	Проверка вещественного значения		●	●	●
115	CHECK_DOUBLE	Проверка вещественного значения двойной точности		●		
	Превращение данных					
38	ASCII	Символ → ASCII код	●	●	●	○
39	CHAR	ASCII код → символ	●	●	●	○
	Работа со строками					
40	MLEN	Взять длину строки	●	●		○
41	DELETE	Удалить подстроку	●	●		○

42	INSERT	Вставить строку	●	●		○
43	FIND	Найти подстроку	●	●		○
44	REPLACE	Заменить подстроку	●	●		○
45	LEFT	Извлечь левую часть	●	●		○
46	MID	Извлечь середину	●	●		○
47	RIGHT	Извлечь правую часть	●	●		○
	Преобразования					
34	TERMOCNV	Преобразование физической величины в температуру	●	●	●	●
35	TERMOINV	Преобразование из температуры в физическую величину	●	●	●	●
36	DATA_STR	Преобразование времени в строку символов	●	●	●	○
84	A_TO_R	Копирование целого числа в вещественное		●		
85	R_TO_A	Копирование вещественного числа в целое		●		
86	FROM_EUHI	Преобразование из формата EUHI в 32 разрядное вещественное		●		
87	TO_EUHI	Преобразование 32 разрядного вещественного числа в формат EUHI		●		
88	I_TO_S	Преобразование целого числа в строку с форматированием		●		
89	R_TO_S	Преобразование вещественного числа в строку с форматированием		●		
96	TIME_TO_UNIX	Преобразование времени из формата Unimod в формат Unix		●		
110	TO_IP_ADDR	Формирование IP адреса из текстового представления		●		
	Логические операции					
5	GETBYTE	Выделение байта в переменной	●	●	●	●
6	SETBYTE	Установка байта в переменной	●	●	●	●
7	GETBIT	Выделение бита в переменной	●	●	●	●
8	SETBIT	Установка бита в переменной	●	●	●	●
91	GETFIELD	Выделение группы бит в переменной		●		
92	SETFIELD	Установка группы бит в переменной		●		
106	GETAR	Выделение байтов из массива		●		
107	SETAR	Установка байтов в массиве		●		
116	GETAR_D	Выделение вещественного двойной точности из массива		●		
117	SETAR_D	Установка вещественного двойной точности в массиве		●		
	Файловые операции					

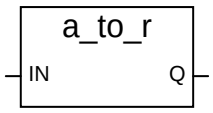
83	FL_FORMAT	Форматирование накопителя			●	●
48	FL_OPEN	Открытие файла	●	●	●	○
49	FL_CLOSE	Закрытие файла	●	●	●	●
50	FL_DEL	Удаление файла	●	●	●	○
51	FL_SPOS	Установка текущей позиции в файле	●	●	●	●
52	FL_GPOS	Получение текущей позиции в файле	●	●	●	●
53	FL_RD_A	Чтение из файла в массив	●	●		○
104	FL_RD_A	Чтение из файла указанного кол-ва байт в массив	●	●		○
54	FL_RD_I	Чтение из файла целой переменной	●	●	●	●
55	FL_RD_R	Чтение из файла вещественной переменной	●	●	●	●
56	FL_RD_M	Чтение из файла сообщения	●	●	●	○
68	FL_RD_B	Чтение из файла байта			●	●
57	FL_WR_A	Запись в файл массива	●	●		○
105	FL_WR_AR	Запись в файл указанного кол-ва байт из массива	●	●		○
58	FL_WR_I	Запись в файл целой переменной	●	●	●	●
59	FL_WR_R	Запись в файл вещественной переменной	●	●	●	●
60	FL_WR_M	Запись в файл сообщения	●	●	●	○
69	FL_WR_B	Запись в файл байта			●	●
66	FL_OPEN_I	Открытие файла			○	●
67	FL_DEL_I	Удаление файла			○	●
61	FL_ERR	Код ошибки операции с файлом	●	●	●	●
82	FL_SCAN	Преобразование файла в массив		●		○
90	FL_COPY	Копирование файла		●		
	Работа с пультом оператора M920L					
70	CLRSCR	Очистка экрана пульта оператора			○	●
71	GOTOXY	Установка позиции курсора			○	●
76	PUT_CHAR	Вывод байта на терминал			○	●
77	PUT_INT	Вывод целого значения на терминал			○	●
78	PUT_REAL	Вывод вещественного значения на терминал			○	●
80	PUT_RSRC	Вывод ресурса на терминал			○	●
79	PUT_MSG	Вывод строки на терминал			○	○
72	GET_CHAR	Чтение кода нажатой клавиши			○	●
73	GET_INT	Ввод целого значения с клавиатуры			○	●
74	GET_REAL	Ввод вещественного значения с клавиатуры			○	●
75	GET_MSG	Ввод строки с клавиатуры			○	○
	Работа с архивами					
98	HDA_LINK_TS	Привязка к переменной HDA времени и даты из другого источника		●		
99	HDA_WRITE_VAR	Принудительная запись переменной HDA в буфер		●		

108	SOE_TO_HDA	Привязка модуля с регистрацией событий к переменной HDA		●		
109	SOE_TO_ARRAY	Привязка модуля с регистрацией событий к массивам		●		
	Разное					
9	DZONE	Зона нечувствительности	●	●	●	●
62	INDEX	Поиск индекса в массиве	●	●		○
63	INTERP	Интерполяция	●	●		○
113	INTERP1	Линейная интерполяция		●		
114	DALIP00	Двойная аналоговая линейная интерполяция		●		
64	SECTION	Кусочно-линейная функция	●	●		○
65	OPERATE	Тестирование и переинициализация мезонина (юнита)	●	●	●	●
95	CRC	Вычисление контрольной суммы		●		
97	BIT_TO_INT	Упаковка 32 булевских значения в целое число		●		
101	AR_COPY	Копирование массива		●		
112	CONFIG	Управление системными настройками		●		

Использованные условные обозначения

Маркировка	Комментарий
●	Функция или функциональный блок реализован на данной платформе
○	Реализация невозможна (ограничение TIC-кода, отсутствие интерфейса или аппаратуры, для управления которыми предназначена данная функция или функциональный блок).
-	Прочерк обозначает, что данная функция или функциональный блок на данной платформе недоступна.
	Пустое поле обозначает, что функция или функциональный блок не реализованы, но реализация возможна.

7.4.1 A_TO_R



Входы:

IN	INTEGER	Целое число

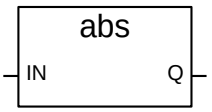
Выход:

Q	REAL	Вещественное число

Назначение

Функция используется для копирования целого числа в вещественное.

7.4.2 ABS



Вход:

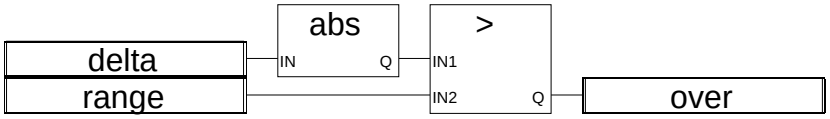
IN	REAL	любая знаковая вещественная величина

Выход:

Q	REAL	модуль вещественной величины

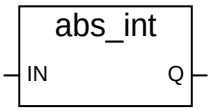
Описание:
Дает абсолютное значение вещественной величины.

(*FBD пример блока “ABS”*)




```
(* ST Эквивалент: *)
over := (ABS (delta) > range);
```

7.4.3 ABS_INT



Вход:

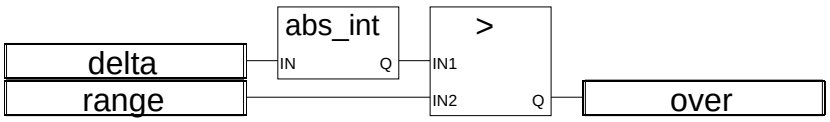
IN	INTEGER	любая знаковая целая величина

Выход:

Q	INTEGER	модуль целой величины

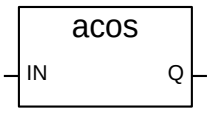
Описание:
Дает абсолютное значение целой величины.

(*FBD пример блока “ABS_INT”*)




```
(* ST Эквивалент: *)
over := (ABS_INT (delta) > range);
```

7.4.4 ACOS



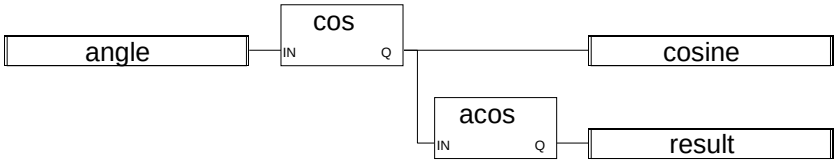
Вход:

IN	REAL	Вещественная величина должна быть в диапазоне [-1.0..+1.0]

Выход:

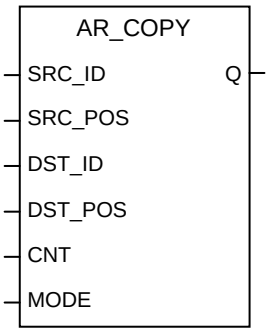
Q	REAL	Арккосинус входа в диапазоне [0.0..Pi]

Описание:
Вычисляет арккосинус вещественной величины.
(*FBD пример блока “COS” и “ACOS”*)




```
(* ST Эквивалент: *)
cosine := COS (angle);
result := ACOS (cosine); (*результат равен углу *)
```

7.4.5 AR_COPY



Вход:

SRC_ID	#ARRAY	Идентификатор массива-источника
SRC_POS	INTEGER	Позиция элемента массива-источника
DST_ID	#ARRAY	Идентификатор массива-приемника
DST_POS	INTEGER	Позиция элемента массива-приемника
CNT	INTEGER	Количество элементов
MODE	INTEGER	Режим: 0 – копирование элементов; 1 – установка CNT-элементов массива-приемника в значение, на которое указывает SRC_POS

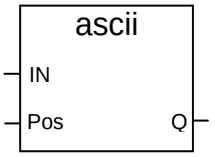
Выход:

Q	INTEGER	Результат операции: 0 – выполнено успешно; 1 – недопустимые входные данные; 2 – выход за границу массива-источника; 3 – выход за границу массива-приемника; 4 – несоответствие типа массивов; 5 – массив-источник не открыт;

6 – массив-приемник не открыт;
7 – несоответствие размеров сообщений.

Описание:
Значения DATE и DMSEC могут быть получены, например, из функции GETTIME.

7.4.6 ASCII



Входы:

IN	MESSAGE	Любая непустая строка
Pos	INTEGER	Позиция выбранного символа в диапазоне [1..len] (len - длина сообщения IN)

Выход:

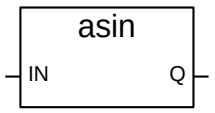
Q	INTEGER	ASCII код выбранного символа (в диапазоне [0..255]) возвращает 0, если Pos вне строки

Описание:
Возвращает ASCII-код символа в строке.

(*FBD пример блока “ASCII”*)


```
(* ST Эквивалент: *)
FirstChr := ASCII (message, 1);
(* FirstChr - это ASCII код первого символа строки *)
```

7.4.7 ASIN



Вход:

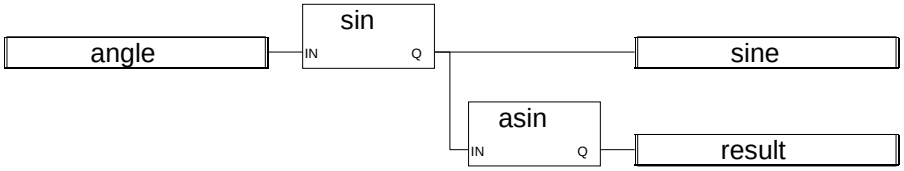
IN	REAL	Вещественная величина должна быть в диапазоне [-1.0..+1.0]

Выход:

Q	REAL	Арксинус входа в диапазоне [-PI/2..+PI/2]

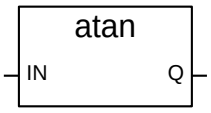
Описание:
Вычисляет арксинус вещественной величины.

(*FBD пример блоков “SIN” и “ASIN”*)




```
(* ST Эквивалент: *)
sine := SIN (angle);
    result := ASIN (sine); (*результат равен углу *)
```

7.4.8 ATAN



Вход:

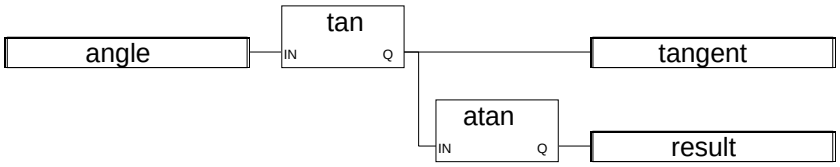
IN	REAL	Вещественная величина

Выход:

Q	REAL	Арктангенс входа в диапазоне [-PI/2..+PI/2]

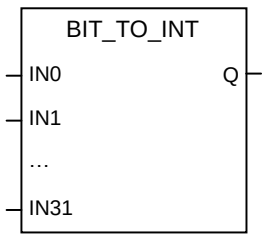
Описание:
Вычисляет арктангенс вещественной величины.

(*FBD пример блока “TAN” и “ATAN”*)




```
(* ST Эквивалент: *)
tangent := TAN (angle);
result := ATAN (tangent); (*результат равен углу *)
```

7.4.9 BIT_TO_INT



Вход:

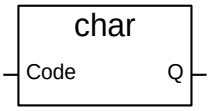
IN0	BOOLEAN	Дискретный вход 0
IN1	BOOLEAN	Дискретный вход 1
IN31	BOOLEAN	Дискретный вход 31

Выход:

Q	INTEGER	Упакованное число

Описание:
Упаковка 32 булевских значения в целое число.

7.4.10 CHAR



Вход:

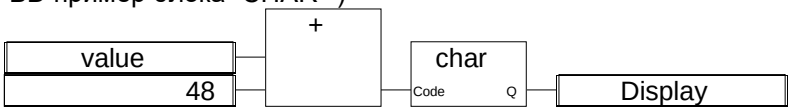
Code	INTEGER	Код в диапазоне [0..255]

Выход:

Q	MESSAGE	Формирует строку из одного символа символ имеет ASCII код заданный на входе (используется ASCII код по модулю 256)

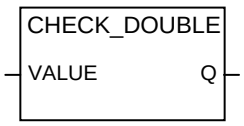
Описание:
Формируется строка из одного символа с заданным ASCII кодом.

(*FBD пример блока “CHAR”*)




```
(* ST Эквивалент: *)
Display := CHAR ( value + 48 );
(* значение в диапазоне [0..9] *)
(* 48 - это ascii код '0' *)
    (* result - строка из одного символа от '0' до '9' *)
```

7.4.11 CHECK_DOUBLE



Вход:

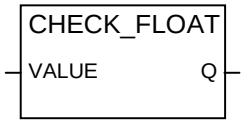
VALUE	DOUBLE	Вещественное число двойной точности

Выход:

Q	INTEGER	Значение:		
		DOUBLE_TYPE_GOOD	0x00	правильное значение
		DOUBLE_TYPE_INF	0x01	+бесконечность или -бесконечность
		DOUBLE_TYPE_NAN	0x02	не правильное чило (NaN/QNaN)
		DOUBLE_TYPE_INDF	0x03	неопределенность

Назначение
Проверка вещественного числа двойной точности

7.4.11 CHECK_FLOAT



Вход:

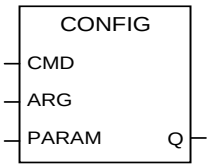
VALUE	REAL	Вещественное число

Выход:

Q	INTEGER	Значение:	
	FLOAT_TYPE_GOOD	0x00	правильное значение
	FLOAT_TYPE_INF	0x01	+бесконечность или -бесконечность
	FLOAT_TYPE_NAN	0x02	не правильное чило (NaN/QNaN)
	FLOAT_TYPE_INDF	0x03	неопределенность

Назначение
Проверка вещественного числа

7.4.45 CONFIG



Входы:

CMD	INTEGER	Команда
ARG	INTEGER	Аргумент
PARAM	INTEGER	Дополнительный параметр

Выход:

Q	Integer	Возвращаемое значение (в случае ошибки: -1)

Назначение
Функция используется для получения и изменения сетевых настроек.
Поддерживаются следующие команды:

1 - Получить IP-адрес порта. Номер порта задается входом ARG (ARG=1..N). Возвращаемое значение - IP-адрес.
2 - Изменить IP-адрес порта в конфигурационном файле. Новый IP-адрес применится после перезапуска модуля. Номер порта задается входом ARG (ARG=1..N), IP-адрес - входом PARAM. В случае успешного завершения возвращаемое значение = 0.
3 - Временно изменить текущий IP-адрес порта (при этом конфигурационный файл не изменяется, после перезапуска модулю вернется прежний IP-адрес). Номер порта задается входом ARG (ARG=1..N), IP-адрес - входом PARAM. В случае успешного завершения возвращаемое значение = 0.
4 - Обнулить маршрут до указанного IP-адреса. Применяется однократно после замены удаленного устройства на аналогичное, с тем же IP-адресом. Вход ARG не используется и должен быть равен 0. IP-адрес удаленного устройства задается входом PARAM.
5 - Получить MAC-адрес порта. Номер порта задается входом ARG (1..N). Вход PARAM может принимать следующие значения:
0 - Получение младшей части
1 - Получение старшей части
6 - Получить идентификатор коммуникационного адаптера (поддерживается только на мастер-модуле M903E). Идентификатор состоит из 3 целых переменных, поэтому вычитывается за 3 запроса. Вход ARG при этом должен принимать значения 0,1,2.

Вход PARAM не используется и должен быть равен 0.

Примечания:
а) Команда 2 изменяет IP-адрес после перезапуска модуля, команда 3 - временно, до перезапуска. Таким образом, для немедленного и постоянного изменения IP-адреса необходимо выполнить два вызова.
б) Для преобразований IP-адреса из цифровое представления в текстовый и обратно используются функции TO_IP_ADDR и FROM_IP_ADDR.
в) Получение MAC-адреса.
Если Ethernet контроллер имеет MAC-адрес FC:83:29:00:01:02,
результат вызова функции будет следующий:
config(5,0) : 16#29000102.
config(5,1) : 16#FC83

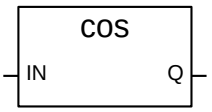
7.4.12 CLRSCR

Назначение: очищает экран пульта оператора M920L и устанавливает позицию курсора в верхний левый угол.

Выход

Q	Integer	Код ошибки 0 - операция выполнена успешно 22 - файл не найден: на модуле не установлен пульт оператора M920L 100 - файл занят: выполняется асинхронный ввод с клавиатуры пульта оператора

7.4.13 COS



Вход:

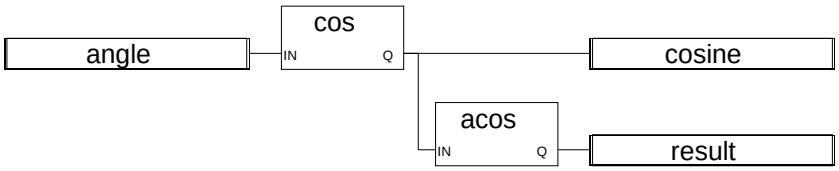
IN	REAL	Вещественная величина

Выход:

Q	REAL	Косинус входа в диапазоне [-1.0..+1.0]

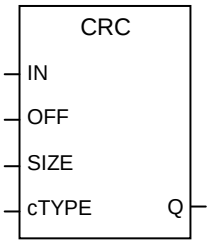
Описание:
Вычисляет косинус вещественной величины.

(*FBD пример блока “COS” и “ACOS”*)




```
(* ST Эквивалент: *)
cosine := COS (angle);
result := ACOS (cosine); (*результат равен углу *)
```

7.4.14 CRC



Вход:

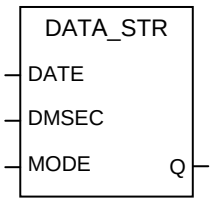
IN	#Integer	Массив данных
OFF	Integer	Смещение стартового элемента (в байтах)
SIZE	Integer	Длина (в байтах)
cTYPE	Integer	Тип контрольной суммы: 0 - простое суммирование байт 1 - crc16

Выход:

Q	Integer	Значение контрольной суммы

Описание:
Вычисление контрольной суммы для массива целых чисел, задаваемого входом IN. Вход OFF задает смещение относительно начала массива, с которого необходимо вычислять контрольную сумму. Длина данных для вычисления задается входом SIZE, алгоритм – входом cTYPE.

7.4.15 DATE_STR



Входы:

DATE	INTEGER	Дата: день, месяц, год (выход функ. блока TIMEDATA)
DMSEC	INTEGER	Миллисекунды с начала суток (выход функ. блока TIMEDATA)
MODE	INTEGER	Режим преобразования: 0 – дата и время 1 – дата 2 – время

Выход:

Q	MESSAGE	Выходная строка символов

Назначение

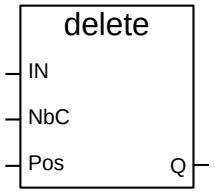
Преобразование времени в строку символов.
Формат строкового представления времени определяется режимом преобразования (вход *MODE*):

- 0 - “mm/dd/yy HH:MM:SS”
- 1 - “mm/dd/yy”
- 2 - “HH:MM:SS”

Каждая составляющая времени представляется в виде двух цифр:

- mm - месяц;
- dd - день;
- yy - год;
- HH - час;
- MM - минута;
- SS - секунда.

7.4.16 DELETE



Входы:

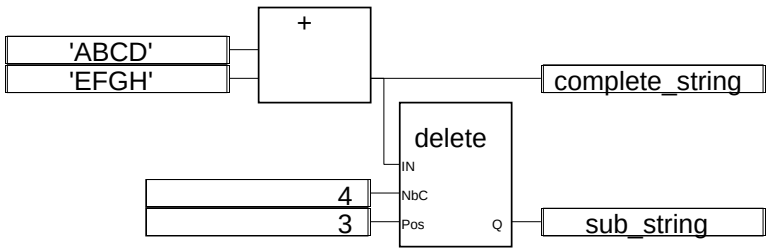
IN	MESSAGE	Любая непустая строка
NbC	INTEGER	Количество символов, которые нужно удалить
Pos	INTEGER	Позиция первого символа для удаления (позиция первого символа строки - 1)

Выход:

Q	MESSAGE	Измененная строка: пустая строка если Pos < 1 первоначальная строка если Pos > длина IN первоначальная строка если NbC <= 0

Описание:
Удаляет часть строки.

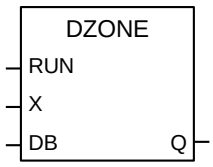
(*FBD пример блока “DELETE”*)



(* ST Эквивалент: *)

```
complete_string := 'ABCD' + 'EFGH'; (* полная строка - это 'ABCDEFGH' *)
sub_string := DELETE (complete_string, 4, 3); (* sub_string is 'ABGH' *)
```

7.4.17 DZONE



Входы:

RUN	BOOLEAN	Режим работы: TRUE-нормальный, FALSE-отключенный
X	REAL	Основной вход
DB	REAL	Зона нечувствительности

Выход:

Q	REAL	Основной выход

Назначение

Функция DZONE используется при создании схем регулирования, чтобы оградить исполнительные органы от лишних операций при малом отличии регулируемой величины от значения задания.

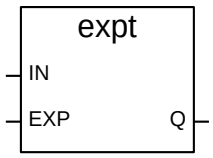
Описание алгоритма

При состоянии входа RUN=TRUE алгоритм работает в соответствии со следующим законом:

$$Q=0, \text{ если } |X| \le DB/2,$$
$$Q = X - DB/2, \text{ если } X > DB/2,$$
$$Q = X + DB/2, \text{ если } X < DB/2,$$

где DB – заданная зона нечувствительности, X – основной вход алгоритма, *Q* – основной выход алгоритма Т.е. при абсолютном значении входа $|X| \le DB/2$ выход алгоритма равен нулю, в остальных случаях из значения X вычитается значение DB/2 в соответствии со знаком. Значение входа DB<0 алгоритм воспринимает как DB=0. При состоянии входа RUN=FALSE выход алгоритма равен входу *Q*=X.

7.4.18 EXPT



Входы:

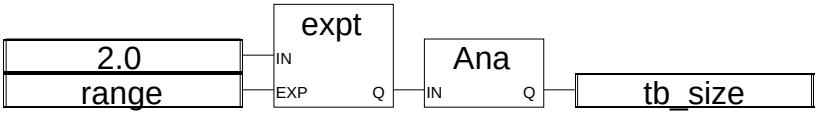
IN	REAL	Знаковая вещественная база
EXP	INTEGER	Знаковая целая степень

Выход:

Q	REAL	Результат функции равен IN в степени EXP

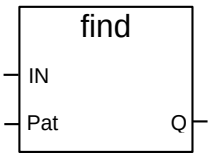
Описание:
Дает вещественной результат операции: (база экспонента) ‘база’ - первый аргумент, экспонента - второй.

(*FBD пример блока “EXPT”*)



(* ST Эквивалент: *)
tb_size := ANA (EXPT (2.0, range));

7.4.19 FIND



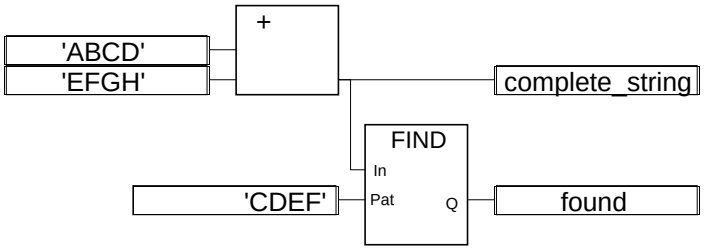
Входы:

IN	MESSAGE	Любая строка
Pat	MESSAGE	Любая непустая строка

Выход:

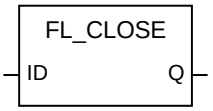
Q	INTEGER	Находит позицию первого символа вхождения подстроки Pat в строке IN =0 если подстрока Pat не найдена

Описание:
Находит подстроку в строке. Возвращает положение подстроки в строке. Функция отличает заглавные буквы от прописных.
(*FBD пример блока “FIND”*)



(* ST Эквивалент: *)
complete_string := 'ABCD' + 'EFGH'; (* полная строк - это 'ABCDEFGH' *)
found := FIND (complete_string, 'CDEF'); (* найдено 3 *)

7.4.20 FL_CLOSE



Вход:

ID	INTEGER	Идентификатор файла.

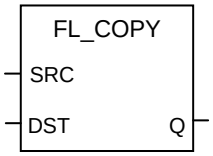
Выход:

Q	INTEGER	Результат операции: 0 – выполнено успешно 9 – ошибка при закрытии файла 12 – файл не открыт

Назначение

Функция FL_CLOSE используется для закрытия файла, открытого функцией FL_OPEN.

7.4.21 FL_COPY



Вход:

SRC	MESSAGE	Полное имя исходного файла
DST	MESSAGE	Полное имя файла назначения

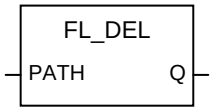
Выход:

Q	INTEGER	Результат операции: 0 – выполнено успешно 1 - ошибка при работе с памятью 2 - несоответствие атрибутов объекта 3 - недопустимые входные данные 5 - ошибка при копировании данных 22 - файл не найден

Назначение

Функция FL_COPY используется для копирования файла.

7.4.22 FL_DEL



Вход:

PATH	MESSAGE	Полное имя файла

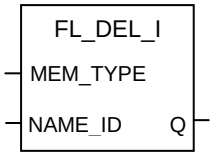
Выход:

Q	INTEGER	Результат операции: 0 – выполнено успешно <>0 – ошибка

Назначение

Удаление файла. Функция FL_DEL используется для открытия файла только на мастер-модуле, а на интеллектуальном модуле для этих целей используется функция FL_ DEL_I.

7.4.23 FL_DEL_I



Входы:

MEM_TYPE	INTEGER	Тип используемой памяти (FRAM/SRAM/FLASH и т.п.)
NAME_ID	INTEGER	Имя файла в системе

Выход:

Q	INTEGER	Код ошибки

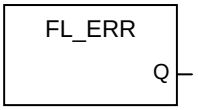
Назначение

Функция FL_DEL_I переназначена для удаления файла или ресурса. Тип памяти и файловой системы, в которой расположен файл, задается параметром MEM_TYPE, а имя файла – параметром NAME_ID.

Удаляемый файл не должен быть открыт для чтения или записи. Проба удаления файла, открытого для записи или чтения, возвращает код ошибки “2 - несоответствие атрибутов объекта”.

Функция FL_ DEL _I используется для открытия файла или ресурса только на интеллектуальном модуле.

7.4.24 FL_ERR



Выход:

Q	INTEGER	Код ошибки

Назначение

Получение кода ошибки последней операции с массивом/файлом

Для мастер-модулей ***M841E/M902E/M921E/M915E*** выход Q может принимать следующие значения:

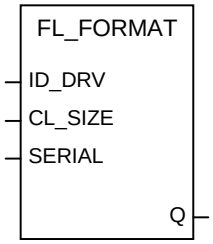
1 - операция не поддерживается
2 - путь к файлу некорректен, или попытка открыть несуществующий файл при установленном флаге RDONLY
4 - операция прервана системным сигналом
5 - ошибка чтения/записи на устройство
6 - устройство не найдено
9 - задан неверный идентификатор файла
13 - доступ запрещен
16 - ошибка открытия файла
20 - путь к директории задан некорректно
21 - указано имя директории
23 - открыто слишком много файлов
24 - открыто слишком много файловых дескрипторов
27 - размер файла превышает допустимое значение
28 - недостаточно свободного места на диске
30 - файл доступен только для чтения
78 - слишком длинное имя файла
79 - размер файла превышает допустимое значение
89 - операция для указанной файловой системы не поддерживается
302 - ошибка файловой системы

Для остальных модулей выход Q принимает следующие значения:

1 - ошибка при работе с памятью
2 - несоответствие аттрибутов объекта

3 - недопустимые входные данные
4 - объект уже существует
5 - ошибка при копировании данных
6 - ошибка при открытии файла
7 - ошибка при установке текущей позиции в файле
8 - ошибка при чтении текущей позиции в файле
9 - ошибка при закрытии файла
10 - позиция чтения опережает позицию записи в кольцевом файле
11 - плохое имя файла (превышает макс.длинну или равно нулю)
12 - файл не открыт
14 - массив не открыт
15 - ошибка заполнения массива из файла (несоответствие размеров)
16 - достигнут конец файла
20 - Функция не реализована. Вызываемая функция не поддерживается
21 - Неизвестный тип объекта. При открытии файла указано неверное значение типа памяти
22 - файл не найден
23 - ошибка преобразования данных
24 - аппаратная ошибка
25 - устройство не отформатировано
99 - асинхронная операция прервана пользователем
100 - асинхронная операция еще не выполнена

7.4.25 FL_FORMAT



Входы:

ID_DRV	INTEGER	Идентификатор устройства хранения
CL_SIZE	INTEGER	Размер кластера в байтах
SERIAL	INTEGER	Серийный номер

Выход:

Q	INTEGER	Количество отформатированных кластеров

Назначение

Функция FL_FORMAT предназначена для форматирования устройства хранения. На M911 идентификатор устройства может принимать значения: 1 - SRAM или 2 – FLASH; 0 - стандартная файловая система (мастер модуль на платформе PC104), 2 - файловая система в памяти FRAM (интеллектуальный модуль), 3 - файловая система во внешней памяти FLASH (интеллектуальный модуль).

Размер кластера в байтах различный для разных устройств. Для SRAM может быть любое значение из диапазона от 4096 (0x1000) до 262144 (0x40000), для FLASH правильное значение размера кластера можно определить по следующей формуле: CL_SIZE = 64К x Cnt, где Cnt может принимать значения от 1 до 28.

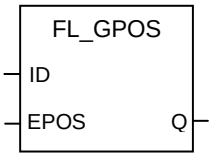
От размера кластера зависит количество записей в FAT и соответственно количество сохраняемых файлов. Так, например, если размер кластера для SRAM равен 64К, то всего будет 256/64=4 записи в FAT SRAM (устройство “/S/”). Попытка записи пятого файла на это устройство вызовет ошибку 1 - «Недостаточно памяти».

Серийный номер SERIAL - любое 32-х разрядное целое число, которое запоминается в отдельном поле FAT. На параметры форматирования серийный номер не влияет.

Если количество отформатированных кластеров равно нулю, произошла ошибка форматирования. Прочитать код ошибки можно с помощью функции FL_ERR.

Более подробная информацию по работе с файловой системой содержится в документе: Unimod Pro. Файловый менеджер. п.п. 1.1 «Файловая система контроллеров».

7.4.26 FL_GPOS



Входы:

ID	INTEGER	Идентификатор файла
EPOS	INTEGER	Значение позиции в случае ошибки

Выход:

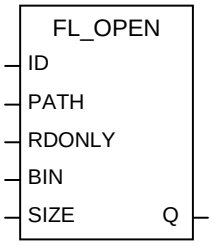
Q	INTEGER	Текущая позиция в файле

Назначение

Функция FL_GPOS используется для получения текущей позиции в файле, который был открыт функцией FL_OPEN.

На вход EPOS подается значение, которое будет на выходе функции при обнаружении ошибки. Код ошибки уточняется вызовом функции FL_ERR.

7.4.27 FL_OPEN



Входы:

ID	INTEGER	Идентификатор файла (проверяется наличие открытого файла)
PATH	MESSAGE	Полное имя файла
RDONLY	BOOLEAN	Файл открывается только для чтения
BIN	BOOLEAN	Файл имеет двоичный формат
SIZE	INTEGER	Размер файла

Выход:

Q	INTEGER	Идентификатор файла

Назначение

Функция FL_OPEN используется для открытия файла. При значении входа ID равном нулю – открывается файл с уникальным идентификатором. В противном случае проверяется наличие открытого файла с идентификатором ID. При задании полного имени файла используются символы слэш “ / ”.

После открытия файла текущая позиция устанавливается в начало файла. Позиция может быть явно установлена через вызов функции FL_SPOS. В случае, если размер файла не задан (-1 на входе *SIZE*), существующий файл открывается для чтения и записи. В противном случае файл будет усечен до заданного размера. Если файл не существует – он будет создан.

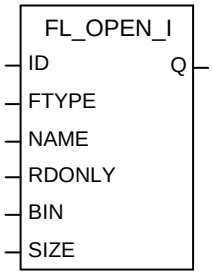
В случае ошибки при открытии файла – идентификатор ID равен нулю. Код ошибки уточняется вызовом функции FL_ERR.

Функция FL_OPEN используется для открытия файла только на мастер-модуле, а на интеллектуальном модуле для этих целей используется функция FL_OPEN_I.

Пример:
Для мастер-модуля M841E/M902E/M921E: "/unimod/sram/file.txt"
Для мастер-модуля M915E: "/fs/etfs/unimod/file.txt"

Более подробная информацию по работе с файловой системой содержится в документе: Unimod Pro. Файловый менеджер. п.п. 1.1 «Файловая система контроллеров».

7.4.28 FL_OPEN_I



Входы:

ID	INTEGER	Идентификатор файла
FTYPE	INTEGER	Тип используемой памяти (FRAM/SRAM/FLASH и т.п.)
NAME	INTEGER	Имя файла в системе
RDONLY	BOOLEAN	Открыть только для чтения
BIN	BOOLEAN	Открыть в двоичном режиме
SIZE	INTEGER	Установить максимальный размер файла, байт

Выход:

Q	INTEGER	Идентификатор файла

Описание функции:

Функция FL_OPEN_I используется для открытия файла или ресурса.

При значении входа ID равном нулю – открывается файл с уникальным идентификатором. На интеллектуальных модулях M953C и M832C уникальные идентификаторы файлов выбираются псевдослучайно в диапазоне от 128 до 255.

При ненулевом входе ID проверяется наличие открытого файла с таким идентификатором. Если такой файл уже открыт, функция FL_OPEN_I не производит проверку параметров и независимо от того, соответствуют ли имя файла, тип памяти и т.п., возвращает идентификатор ID и устанавливает код ошибки “4 – объект уже существует”.

На интеллектуальных модулях максимальное количество одновременно открытых файлов ограничено размером доступной памяти и равно 5-ти:

_BUFFERS	0x05	Максимальное количество открытых файлов

При превышении этого числа функция FL_OPEN_I возвращает ноль и устанавливает код ошибки “1 – ошибка при работе с памятью”.

Для того чтобы открыть ресурс технологической программы, функции FL_OPEN_I необходимо задать соответствующий тип памяти. На сегодняшний момент определены следующие типы памяти:

_RESOURCE	0x0001	Функция FL_OPEN_I открывает ресурс
_FRAMFAT	0x0002	Резерв, файловая система в памяти FRAM
_FLASHFAT	0x0003	Резерв, файловая система во внешней памяти FLASH

Каждый ресурс технологической программы имеет свой собственный уникальный номер, который передаётся функции FL_OPEN_I как имя файла NAME. Опционально, в старших 16 разрядах имени файла может быть указан

тип запрашиваемого ресурса (см. рисунок). Если тип ресурса с заданным именем не совпадёт с запрошенным значением, функция FL_OPEN_I возвращает ноль и устанавливает код ошибки “22 - объект с заданным именем не найден”.

Исполнительная система Unimod Pro поддерживает следующие типы ресурсов:

RT_NONE	0x0000	Неизвестный или любой тип ресурса
RT_ASCIISTRING	0x0001	Строка, массив знаков ASCII
RT_BYTEARRAY	0x0002	Массив байт
RT_INTARRAY	0x0003	Массив целых констант
RT_FLOATARRAY	0x0004	Массив вещественных констант
RT_BINFILE	0x0005	Двоичный файл
RT_TEXTFILE	0x0006	Текстовый файл

Открытие ресурсов технологической программы возможно исключительно в режиме “только для чтения”. При попытке открытия ресурса в режиме для записи (т.е. RDONLY = false), функция FL_OPEN_I возвращает ошибку “2 - несоответствие атрибутов объекта”.

Файл ресурса открывается в двоичном режиме независимо от значения аргумента BIN. Аргумент SIZE не используется и во время работы с ресурсами должен быть равен нулю.

Функция FL_OPEN_I устанавливает следующие коды ошибок:

0 Операция выполнена успешно.
1 Нехватка памяти для открытия файла. Превышено количество открытых файлов.
2 Несоответствие атрибутов объекта. Попытка открытия ресурса для записи.
4 Объект уже существует. Файл с данным идентификатором уже открыт.
21 Указано неверное значение типа памяти.
22 Объект не найден. Не удалось найти файл или ресурс с заданным именем.

Примеры использования:

В приведенном примере технологическая программа открывает массив байт с номером 90 (5Ah), производит проверку на наличие ошибок и читает первые четыре байта ресурса как целое значение.

```
 Handle := FL_OPEN_I (0, 1, 16#2005A, true, false, 0);
if Handle <> 0 then
        Value := FL_RD_I (Handle, -1);
        if (Value <> -1) or (FL_ERR () = 0) then
                (* анализ прочитанного значения *)
        end_if;
        Error := FL_CLOSE (Handle);
end_if;
```

Имя ресурса 16#2005A можно условно разделить на две части. Младшие 16 разрядов задают номер ресурса. Старшие 16 разрядов содержат тип запрашиваемого ресурса: 2 для массива байт.

Для открытия ресурса не обязательно указывать его тип, достаточно оставить старшие 16 разрядов имени файла равными нулю, или иначе – задать тип ресурса RT_NONE. Например:

```
Handle := FL_OPEN_I (0, 1, 90, true, false, 0);
```

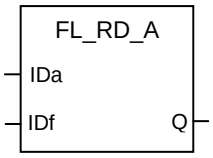
Программа открывает ресурс любого (или неопределенного) типа с номером 90.

В некоторых случаях бывает полезно задавать фиксированные идентификаторы файлов. Для этого служит аргумент ID. При нулевом значении ID идентификатор файла выделяется автоматически.

Функция FL_OPEN_I используется для открытия файла или ресурса только на интеллектуальном модуле.

Более подробная информацию по работе с файловой системой содержится в документе: Unimod Pro. Файловый менеджер. п.п. 1.1 «Файловая система контроллеров».

7.4.29 FL_RD_A



Входы:

IDa	#Пустая ссылка	Идентификатор массива
IDf	INTEGER	Идентификатор файла

Выход:

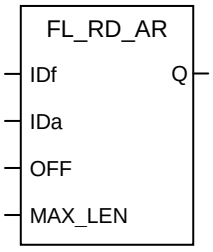
Q	INTEGER	Результат операции 0 - выполнено успешно 3 - недопустимые входные данные 5 - ошибка при копировании данных 7 - ошибка при установке текущей позиции в файле 8 - ошибка при чтении текущей позиции в файле 12 - файл не открыт 14 - массив не открыт

Назначение

Заполнение массива данными из файла.

Чтение из файла производится с текущей позиции для чтения. По завершению операции текущая позиция для чтения увеличивается на размер прочитанного блока данных.

7.4.30 FL_RD_AR



Входы:

IDf	INTEGER	Идентификатор файла
IDa	#Пустая ссылка	Идентификатор массива
OFF	INTEGER	Смещение (в байтах) в массиве
MAX_LEN	INTEGER	Максимальная длина (в байтах) данных на чтение

Выход:

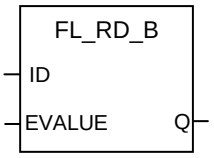
Q	INTEGER	Результат операции >0 - длина прочитанных данных 0 - нет данных на чтение -1 - недопустимые входные данные -2 - ошибка доступа к файлу -3 - ошибка доступа к массиву -4 - ошибка при заполнении массива из файла

Назначение

Заполнение массива данными из файла.

Чтение из файла производится с текущей позиции для чтения. По завершении операции текущая позиция для чтения увеличивается на размер прочитанного блока данных.

7.4.31 FL_RD_B



Входы:

ID	INTEGER	Идентификатор файла
EVALUE	INTEGER	Выходное значение в случае ошибки

Выход:

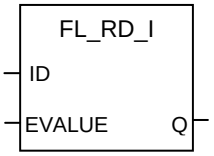
Q	INTEGER	Результат операции 0 - выполнено успешно 3 - недопустимые входные данные 5 - ошибка при копировании данных 7 - ошибка при установке текущей позиции в файле 8 - ошибка при чтении текущей позиции в файле 12 - файл не открыт

Назначение

Чтение из текущей позиции в файле байта данных.

По окончании чтения текущая позиция увеличивается на размер считываемого элемента. Значение EVALUE транслируется на выход функции в случае ошибки. Код ошибки уточняется вызовом функции FL_ERR.

7.4.32 FL_RD_I



Входы:

ID	INTEGER	Идентификатор файла
EVALUE	INTEGER	Значение переменной в случае ошибки

Выход:

Q	INTEGER	Значение переменной

Назначение

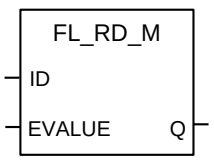
Функция FL_RD_I используется для чтения из файла переменной целого типа.

Файл должен быть предварительно открыт функцией FL_OPEN.

На вход EVALUE подается значение, которое будет на выходе функции при обнаружении ошибки. Код ошибки уточняется вызовом функции FL_ERR.

Чтение из файла производится с текущей позиции для чтения. По завершению операции текущая позиция для чтения увеличивается на размер прочитанного блока данных.

7.4.33 FL_RD_M



Входы:

ID	INTEGER	Идентификатор файла
EVALUE	MESSAGE	Значение переменной в случае ошибки

Выход:

Q	MESSAGE	Значение переменной

Назначение

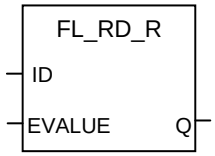
Функция FL_RD_M используется для чтения из файла строковой переменной.

Файл должен быть предварительно открыт функцией FL_OPEN.

На вход EVALUE подается значение, которое будет на выходе функции при обнаружении ошибки. Код ошибки уточняется вызовом функции FL_ERR.

Чтение из файла производится с текущей позиции для чтения. По завершению операции текущая позиция для чтения увеличивается на размер прочитанного блока данных.

7.4.34 FL_RD_R



Входы:

ID	INTEGER	Идентификатор файла
EVALUE	REAL	Значение переменной в случае ошибки

Выход:

Q	REAL	Значение переменной

Назначение

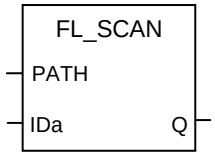
Функция FL_RD_R используется для чтения из файла переменной вещественного типа.

Файл должен быть предварительно открыт функцией FL_OPEN.

На вход EVALUE подается значение, которое будет на выходе функции при обнаружении ошибки. Код ошибки уточняется вызовом функции FL_ERR.

Чтение из файла производится с текущей позиции для чтения. По завершению операции текущая позиция для чтения увеличивается на размер прочитанного блока данных.

7.4.35 FL_SCAN



Входы:

PATH	MESSAGE	Полное имя файла
IDa	#Пустая ссылка	Ссылка на массив (Real, Integer)

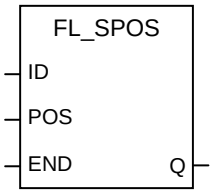
Выход:

Q	INTEGER	Результат операции: 0 – выполнено успешно; 1 – недопустимые входные данные

Назначение

Заполнение массива данными из текстового файла

7.4.36 FL_SPOS



Входы:

ID	INTEGER	Идентификатор файла
POS	INTEGER	Текущая позиция в файле
END	BOOLEAN	Признак установки текущей позиции в конец файла

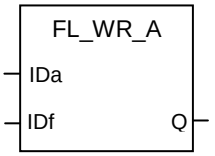
Выход:

Q	INTEGER	Результат операции: 0 – выполнено успешно 7 – ошибка при установке текущей позиции в файле 12 – файл не открыт

Назначение

Функция FL_SPOS используется для установки текущей позиции в файле, который был открыт функцией FL_OPEN.

7.4.37 FL_WR_A



Входы:

IDa	#Пустая ссылка	Имя массива
IDf	INTEGER	Идентификатор файла

Выход:

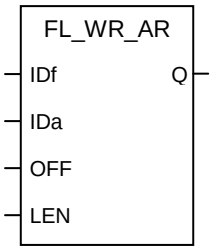
Q	INTEGER	Результат операции 0 - выполнено успешно 3 - недопустимые входные данные 5 - ошибка при копировании данных 7 - ошибка при установке текущей позиции в файле 8 - ошибка при чтении текущей позиции в файле 12 - файл не открыт 14 - массив не открыт

Назначение

Функция FL_WR_A используется для записи в файл массива переменных.

Файл должен быть предварительно открыт функцией FL_OPEN.
Запись в файл производится с текущей позиции для записи. По завершению операции текущая позиция для записи увеличивается на размер записанного блока данных.

7.4.38 FL_WR_AR



Входы:

IDf	INTEGER	Идентификатор файла
IDa	#Пустая ссылка	Идентификатор массива
OFF	INTEGER	Смещение (в байтах) в массиве
LEN	INTEGER	Длина (в байтах) данных на запись

Выход:

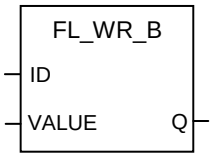
Q	INTEGER	Результат операции >0 - длина записанных данных -1 - недопустимые входные данные -2 - ошибка доступа к файлу -3 - ошибка доступа к массиву -4 - ошибка при записи в файл

Назначение

Функция FL_WR_AR используется для записи в файл массива байт.

Файл должен быть предварительно открыт функцией FL_OPEN.
Запись в файл производится с текущей позиции для записи. По завершении операции текущая позиция для записи увеличивается на размер записанного блока данных.

7.4.39 FL_WR_B



Входы:

ID	INTEGER	Идентификатор файла
VALUE	INTEGER	Значение элемента

Выход:

Q	INTEGER	Результат операции 0 - выполнено успешно 3 - недопустимые входные данные 5 - ошибка при копировании данных 7 - ошибка при установке текущей позиции в файле 8 - ошибка при чтении текущей позиции в файле 12 - файл не открыт

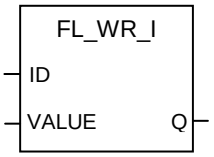
Назначение

Функция FL_WR_B используется для записи в файл одного байта.

Файл должен быть предварительно открыт функцией FL_OPEN. Запись возможна в файлы находящиеся на устройстве “/S/” (SRAM) и открытые в режиме «чтение/запись». Возможна произвольная запись в заданную позицию в файле. Запись в файл производится с текущей позиции для записи. По завершению операции текущая позиция для записи увеличивается на размер записанного блока данных. Если текущая позиция превышает фактический размер файла, то размер файла увеличивается.

Максимальный размер файла зависит от размера кластера, заданного при форматировании (см. FL_FORMAT). Значение элементов файла, в позиции которых запись не производилась, неопределенное.

7.4.40 FL_WR_I



Входы:

ID	INTEGER	Идентификатор файла
VALUE	INTEGER	Значение переменной

Выход:

Q	INTEGER	Результат операции: 0 - выполнено успешно 3 - недопустимые входные данные 5 - ошибка при копировании данных 7 - ошибка при установке текущей позиции в файле 8 - ошибка при чтении текущей позиции в файле 12 - файл не открыт

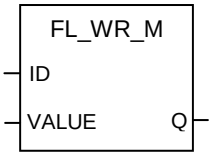
Назначение

Функция FL_WR_I используется для записи в файл переменной целого типа.

Файл должен быть предварительно открыт функцией FL_OPEN.

Запись в файл производится с текущей позиции для записи. По завершению операции текущая позиция для записи увеличивается на размер записанного блока данных.

7.4.41 FL_WR_M



Входы:

ID	INTEGER	Идентификатор файла
VALUE	MESSAGE	Значение переменной

Выход:

Q	INTEGER	Результат операции: 0 - выполнено успешно 3 - недопустимые входные данные 5 - ошибка при копировании данных 7 - ошибка при установке текущей позиции в файле 8 - ошибка при чтении текущей позиции в файле 12 - файл не открыт

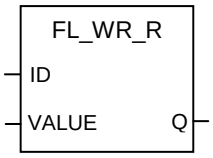
Назначение

Функция FL_WR_M используется для записи в файл строковой переменной.

Файл должен быть предварительно открыт функцией FL_OPEN.

Запись в файл производится с текущей позиции для записи. По завершению операции текущая позиция для записи увеличивается на размер записанного блока данных.

7.4.42 FL_WR_R



Входы:

ID	INTEGER	Идентификатор файла
VALUE	REAL	Значение переменной

Выход:

Q	INTEGER	Результат операции: 0 - выполнено успешно 3 - недопустимые входные данные 5 - ошибка при копировании данных 7 - ошибка при установке текущей позиции в файле 8 - ошибка при чтении текущей позиции в файле 12 - файл не открыт

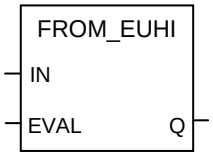
Назначение

Функция FL_WR_R используется для записи в файл переменной вещественного типа.

Файл должен быть предварительно открыт функцией FL_OPEN.

Запись в файл производится с текущей позиции для записи. По завершению операции текущая позиция для записи увеличивается на размер записанного блока данных.

7.4.43 FROM_EUHI



Входы:

IN	INTEGER	Число в формате EUHI
EVAL	REAL	Значение на выходе в случае ошибки

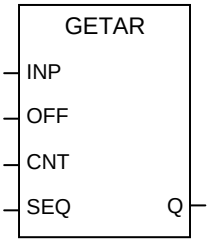
Выход:

Q	REAL	32 разрядное вещественное число

Назначение

Функция используется для преобразования числа из формата EUHI (16 разрядное вещественное) в 32 разрядное вещественное число.

7.4.44 GETAR



Входы:

INP	#ANY	Идентификатор массива
OFF	INTEGER	Смещение (в байтах) в массиве
CNT	INTEGER	Количество байтов для извлечения
SEQ	INTEGER	Порядок байтов:

0 по умолчанию
1 "0-1" - 2х байтовое
2 "1-0" - 2х байтовое, замена байтов
3 "0-1-2" - 3х байтовое
4 "2-1-0" - 3х байтовое, замена байтов
5 "0-1-2-3" - 4х байтовое
6 "1-0-3-2" - 4х байтовое, замена байтов
7 "2-3-0-1" - 4х байтовое, замена слов
8 "3-2-1-0" - 4х байтовое, замена байтов, замена слов

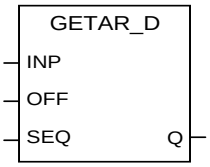
Выход:

Q	INTEGER	Полученное значение (в случае ошибки - 0)

Назначение

Функция GETAR используется для извлечения группы байтов из массива. Содержимое массива не изменяется.

7.4.45 GETAR_D



Входы:

INP	#ANY	Идентификатор массива
OFF	INTEGER	Смещение (в байтах) в массиве
SEQ	INTEGER	Порядок байтов:

0 “0-1-2-3-4-5-6-7” – копирование без замен
1 "1-0-3-2-5-4-7-6" – замена байтов
2 "2-3-0-1-6-7-4-5" – замена слов
3 "3-2-1-0-7-6-5-4" – замена байтов, замена слов
4 “4-5-6-7-0-1-2-3” – замена двойных слов
5 "5-4-7-6-1-0-3-2" – замена двойных слов, замена байтов
6 "6-7-4-5-2-3-0-1" – замена двойных слов, замена слов
7 "7-6-5-4-3-2-1-0" – замена двойных слов, замена байтов, замена слов

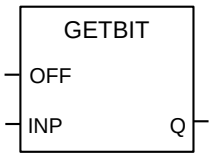
Выход:

Q	DOUBLE	Полученное значение (в случае ошибки - 0)

Назначение

Функция GETAR_D используется для извлечения вещественного значения двойной точности из массива. Содержимое массива не изменяется.

7.4.45 GETBIT



Входы:

OFF	INTEGER	Смещение на бит в переменной [0÷31]
INP	INTEGER	Значение переменной

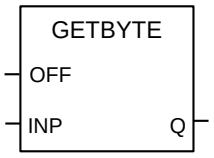
Выход:

Q	BOOLEAN	Значение бита

Назначение

Функция GETBIT используется для выделения бита в переменной целого типа. Значение входной переменной не изменяется.

7.4.46 GETBYTE



Входы:

OFF	INTEGER	Смещение на байт в переменной [0÷3]
INP	INTEGER	Значение переменной

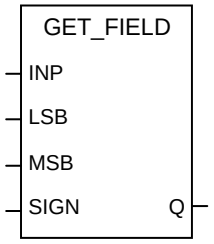
Выход:

Q	INTEGER	Значение выделенного байта

Назначение

Функция GETBYTE используется для выделения байта в переменной целого типа. Значение входной переменной не изменяется.

7.4.47 GETFIELD



Входы:

INP	INTEGER	Значение переменной
LSB	INTEGER	Младший бит в переменной [0-31]
MSB	INTEGER	Старший бит в переменной [0-31]
SIGN	BOOLEAN	Флаг знака (0 – выделяемая величина беззнаковая, 1 – знаковая)

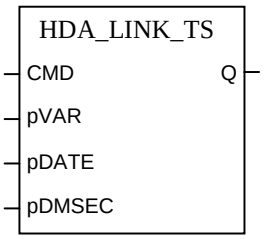
Выход:

Q	INTEGER	Значение

Назначение

Функция GETFIELD используется для выделения группы бит в переменной целого типа. Значение входной переменной не изменяется.

7.4.48 HDA_LINK_TS



Вход:

CMD	BOOLEAN	TRUE – выполнить привязку, FALSE – отменить
pVAR	#VAR	Переменная HDA
pDATE	#VAR	Дата в формате Unimod
pDMSEC	#VAR	Время в формате Unimod

Выход:

Q	INTEGER	Результат выполнения операции

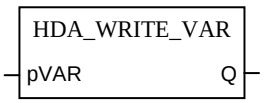
Описание:

Логическое связывание переменной pVAR и метки времени pDATE-pDMSEC.
После выполнения функции переменная pVAR будет заноситься в архив HDA с метками времени, взятыми не из внутреннего таймера контроллера, а из переменных pDATE и pDMSEC.

На данный момент pDATE и pDMSEC должны быть целыми переменными.

Значения DATE и DMSEC могут быть получены, например, из функции GETTIME.

7.4.49 HDA_WRITE_VAR



Вход:

pVAR	#VAR	Переменная HDA

Выход:

Q	INTEGER	Результат выполнения операции

Описание:

Принудительная запись переменной в буфер (переменная должна иметь атрибут "Архивация" в словаре).
При этом параметры “Величина фильтрации” и “Время между сохранениями” игнорируются.

7.4.50 GET_CHAR

Назначение: Функция предназначена для чтения кода нажатой клавиши из буфера клавиатуры

Выход

Q	Integer	Код нажатой клавиши или -1

Коды ошибок:

0 - операция выполнена успешно
2 - несоответствие атрибутов объекта, ошибка записи в файл

16 - достигнут конец файла, буфер клавиатуры пульта оператора пуст
22 - файл не найден, на модуле не установлен пульт оператора M920L
100 - файл занят: выполняется асинхронный ввод с клавиатуры пульта оператора

7.4.51 GET_INT

Назначение: Функция предназначена для чтения значения целого значения из ресурса или файла

Входы:

ID	Integer	Идентификатор файла или консоли.
EValue	Integer	Результат вызова в случае ошибки.

Выход:

Q	Integer	Прочитанное значение или EValue.

Коды ошибок:

0 - операция успешно завершена
2 - несоответствие атрибутов объекта, ошибка чтения
12 - файл не открыт, аргумент ID не является идентификатором файла
16 - достигнут конец файла, недостаточно данных для чтения
23 - ошибка преобразования строки в целое значение
99 - операция прервана: асинхронный ввод прерван пользователем
100 - файл занят: выполняется асинхронный ввод с клавиатуры пульта оператора

7.4.52 GET_MSG

Назначение: Функция предназначена для чтения строки из ресурса или файла.

Входы:

ID	Integer	Идентификатор файла или консоли
EValue	Message	Результат вызова в случае ошибки

Выход:

Q	Integer	Прочитанное значение или EValue

Коды ошибок:

0 - операция успешно завершена
2 - несоответствие атрибутов объекта, ошибка чтения
12 - файл не открыт, аргумент ID не является идентификатором файла
16 - достигнут конец файла, недостаточно данных для чтения
23 - ошибка преобразования строки в целое значение
99 - операция прервана, асинхронный ввод прерван пользователем
100 - файл занят: выполняется асинхронный ввод с клавиатуры пульта оператора

7.4.53 GET_REAL

Назначение: Функция предназначена для чтения значения вещественного типа из файла.

Входы:

ID	Integer	Идентификатор файла или консоли
EValue	Real	Результат вызова в случае ошибки

Выход:

Q	Real	Прочитанное значение или EValue

Коды ошибок:

0 - операция успешно завершена
2 - несоответствие атрибутов объекта, ошибка чтения
12 - файл не открыт, аргумент ID не является идентификатором файла
16 - достигнут конец файла, недостаточно данных для чтения
23 - ошибка преобразования строки в целое значение
99 - операция прервана: асинхронный ввод прерван пользователем
100 - файл занят: выполняется асинхронный ввод с клавиатуры пульта оператора

7.4.54 GOTOXY

Назначение: Задает позицию курсора пульта оператора M920L.

Входы:

X	Integer	Номер знака в линии
Y	Integer	Номер линии

Выход:

Q	Integer	Код ошибки

Описание:
Функция GOTOXY производит установку курсора пульта оператора M920L в произвольную позицию. Новая позиция курсора задаётся аргументами X и Y. Если в списке юнитов интеллектуального модуля не числится юнит U920L, функция GOTOXY устанавливает и возвращает код ошибки "22 - файл не найден".

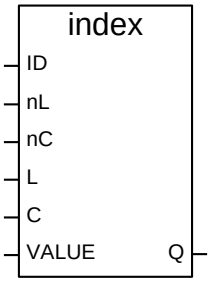
Операция позиционирования курсора кодируется и передаётся в пульт оператора как последовательность трёх байт:

1Bh XX YY
где XX - координата X курсора, YY - координата Y курсора.

Полное писание набора команд можно найти в технической документации пульта оператора M920L.
Коды ошибок:

0 - операция выполнена успешно
22 - файл не найден, на модуле не установлен пульт оператора M920L
100 - файл занят: выполняется асинхронный ввод с клавиатуры пульта оператора

7.4.55 INDEX



Входы:

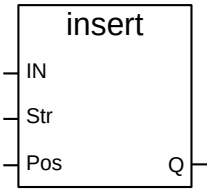
ID	#REAL	Имя массива (элементы типа Real)
nL	INTEGER	Количество строк в массиве
nC	INTEGER	Количество столбцов в массиве
L	INTEGER	Номер строки поиска (-1 - поиск по строкам в столбце)
C	INTEGER	Номер столбца поиска (-1 - поиск по столбцам в строке)
VALUE	REAL	Значение для поиска

Выход:

Q	Integer	Значение индекса (-1 - индекс не найден)

Назначение
Функция INDEX выполняет поиск индекса в действительном массиве. Для поиска индекса используются переменная, подаваемая на вход VALUE. Если на вход C подано ненулевое значение, поиск выполняется в нулевом столбце массива-таблицы. Условие поиска: $(X >= X_{i-1}) \& (X < X_i)$

7.4.56 INSERT



Входы:

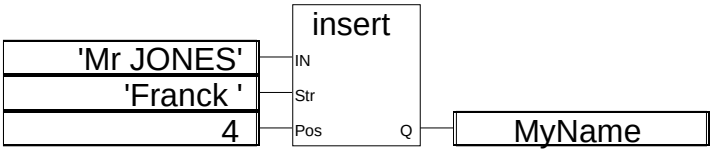
IN	MESSAGE	Начальная строка
Str	MESSAGE	Строка, которую нужно вставить
Pos	INTEGER	Позиция вставки: вставка делается перед указанной позицией (первая позиция - 1)

Выход:

Q	MESSAGE	Измененная строка: пустая строка, если Pos < 1 первоначальная строка, если Pos > длина IN соединение строк если Pos > длина IN

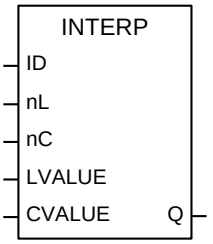
Описание:
Вставляет подстроку в строку, начиная с указанной позиции

(*FBD пример блока “INSERT”*)




```
(* ST Эквивалент: *)
MyName := INSERT ('Mr JONES', 'Frank ', 4);
(* MyName - это 'Mr Frank JONES' *)
```

7.4.57 INTERP



Входы:

ID	#REAL	Имя массива (элементы типа Real)
nL	INTEGER	Количество строк в массиве

nC	INTEGER	Количество столбцов в массиве
LVALUE	REAL	Входное значение для поиска индекса по строкам
CVALUE	REAL	Входное значение для поиска индекса по столбцам

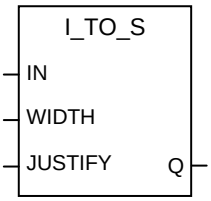
Выход:

Q	REAL	Выходное значение интерполяции

Назначение

Функция INTERP вычисляет интерполированное значение в таблице по 4-м точкам. Для поиска индексов по строкам и столбцам используются переменные, подаваемые на входы LVALUE и CVALUE. Если массивы-векторы IDL не заданы (индексы равны нулю), то поиск индексов осуществляется в нулевом столбце и нулевой строке таблицы, заданной индексом ID.

7.4.58 I_TO_S



Входы:

IN	INTEGER	Целое число
WIGHT	INTEGER	Ширина поля для форматирования
JUSTIFY	INTEGER	Тип выравнивания (0 - по левому краю, 1 - по центру, 2 - по правому краю)

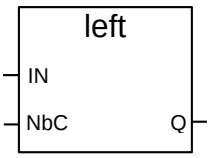
Выход:

Q	MESSAGE	Строка

Назначение

Функция используется для преобразования вещественного числа в строку с форматированием.

7.4.59 LEFT



Входы:

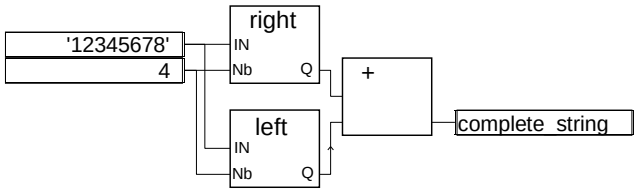
IN	MESSAGE	Любая непустая строка
NbC	INTEGER	Количество символов, которые нужно изъять (не может быть больше строки IN)

Выход:

Q	MESSAGE	Левая часть строки IN (ее длина - NbC) Пустая строка если NbC < 0 Вся строка IN если NbC > длины IN

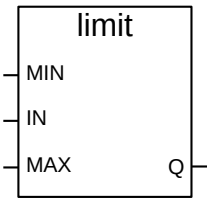
Описание:

Чтение левой часть строки. Количество символов задано.
(*FBD пример блоков “LEFT” и “RIGHT”*)




```
(* ST Эквивалент: *)
complete_string := RIGHT ('12345678', 4) + LEFT ('12345678', 4);
(* полная строка - это '56781234' значение выходящее из RIGHT - это '5678'
значение выходящее из LEFT  - это '1234'*)
```

7.4.60 LIMIT



Входы:

MIN	INTEGER	Минимальная допустимая величина
IN	INTEGER	Любая знаковая целая величина
MAX	INTEGER	Максимальная допустимая величина

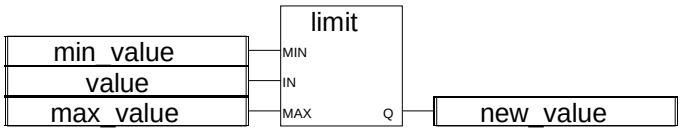
Выход:

Q	INTEGER	Выходная величина ограниченная заданным диапазоном

Описание:

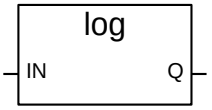
Удерживает входную величину в указанном диапазоне (между максимумом и минимумом). Входная величина становится равной максимуму, если она больше максимума или становится равной минимуму, если она меньше минимума.

(*FBD пример блока “LIMIT”*)




```
(* ST Эквивалент: *)
new_value := LIMIT (min_value, value, max_value);
(* ограничивает значение в диапазоне [min_value..max_value] *)
```

7.4.61 LOG



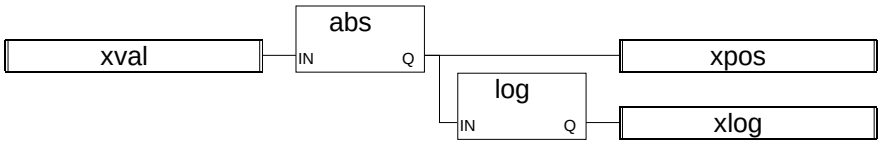
Вход:

IN	REAL	Вещественная величина, должна быть больше 0

Выход:

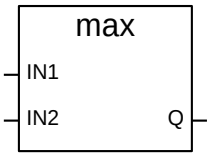
Q	REAL	Логарифм (основание 10) входной величины

Описание:
Вычисляет логарифм (основание 10) вещественной величины.
(*FBD пример блока “LOG”*)




```
(* ST Эквивалент: *)
xpos := ABS (xval);
xlog := LOG (xpos);
```

7.4.62 MAX



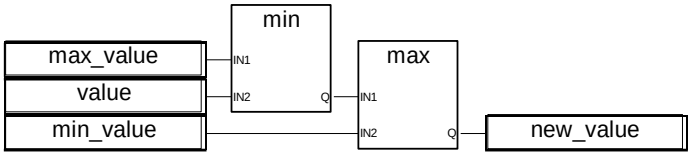
Входы:

IN1	INTEGER	Любая знаковая целая величина
IN2	INTEGER	Любая знаковая целая величина

Выход:

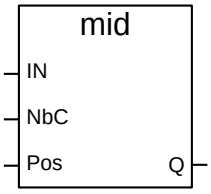
Q	INTEGER	Максимум из двух входных значений

Описание:
Определяет максимальное значение из двух целых.
(*FBD пример блоков “MIN” и “MAX”*)




```
(* ST Эквивалент: *)
new_value := MAX (MIN (max_value, value), min_value);
(*ограничивает значение в диапазоне [min_value..max_value] *)
```

7.4.63 MID



Входы:

IN	MESSAGE	Любая непустая строка
NbC	INTEGER	Количество символов, которые нужно изъять, не может быть больше длины строки IN

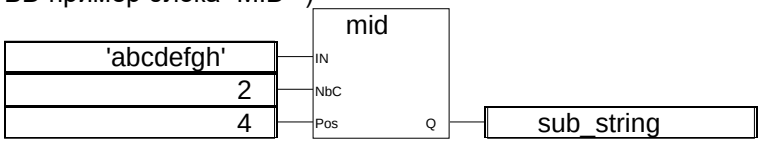
Pos	INTEGER	Позиция подстроки Pos, должна указывать на первый символ подстроки (первая правильная позиция 1)

Выход:

Q	MESSAGE	Средняя часть строки IN (ее длина = NbC) пустая строка если параметры неправильные

Описание:
Чтение части строки по количеству символов и позиции первого символа.

(*FBD пример блока “MID”*)




```
(* ST Эквивалент: *)
sub_string := MID ('abcdefgh', 2, 4);
(* подстрока - 'de' *)
```

7.4.64 MIN

Входы:

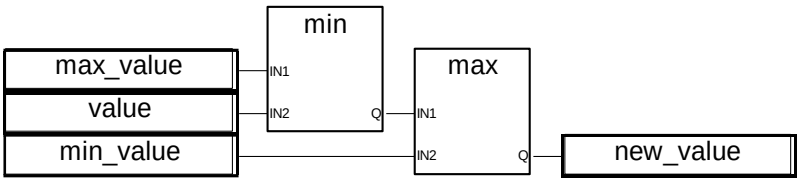
IN1	INTEGER	Любая знаковая целая величина
IN2	INTEGER	Любая знаковая целая величина

Выход:

Q	INTEGER	Минимум из двух входных значений

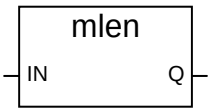
Описание:
Определяет минимальное значение из двух целых величин.

(*FBD пример блоков “MIN” и “MAX”*)




```
(* ST Эквивалент: *)
new_value := MAX (MIN (max_value, value), min_value);
(*ограничивает значение в диапазоне [min_value..max_value] *)
```

7.4.65 MLEN



Вход:

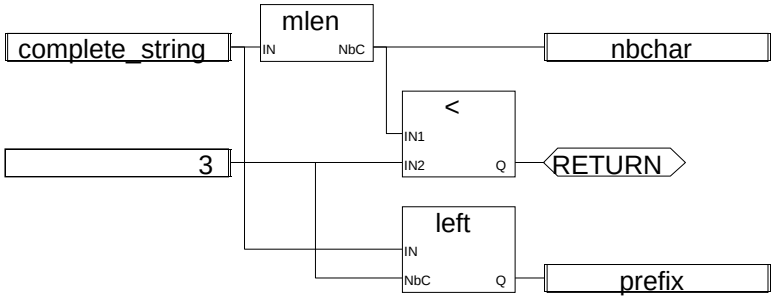
IN	MESSAGE	Любая непустая строка

Выход:

Q	INTEGER	Количество символов в строке IN

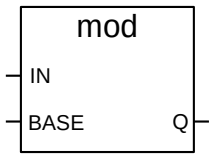
Описание:
Вычисляет длину строки.

(*FBD пример блока “MLEN”*)



(* ST Эквивалент: *)
nbchar := MLEN (complete_string);
If (nbchar < 3) Then Return; End_if;
prefix := LEFT (complete_string, 3);
(*эта программа извлекает 3 символа из левой части строки, кладет результат в строковую переменную prefix и никаких действий не выполняется, если длина строки менее чем 3 символа*)

7.4.66 MOD



Входы:

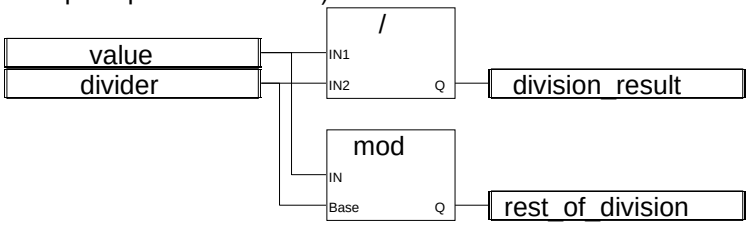
IN	INTEGER	Любая знаковая целая величина
Base	INTEGER	Любая знаковая целая величина больше нуля

Выход:

Q	INTEGER	Остаток от деления (IN / Base) возвращает -1, если Base <= 0

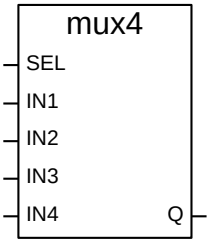
Описание:
Вычисляет остаток от деления.

(*FBD пример блока “MOD”*)



(* ST Эквивалент: *)
division_result := (value / divider); (* целое деление *)
rest_of_division := MOD (value, divider); (* остаток от деления *)

7.4.67 MUX4



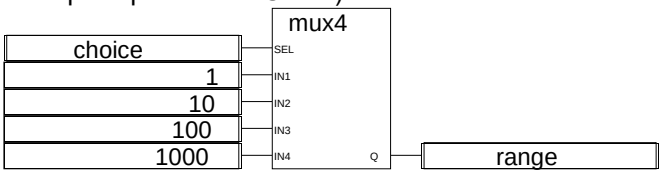
Входы:

SEL	INTEGER	Целый селектор (должен быть в диапазоне [0..3])
IN1..IN4	INTEGER	Целая аналоговая величина

Выход:

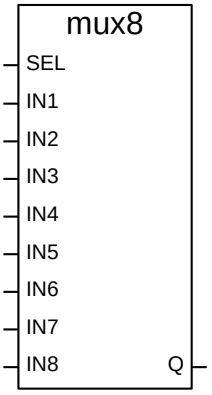
Q	INTEGER	= IN1 если SEL = 0 = IN2 если SEL = 1 = IN3 если SEL = 2 = IN4 если SEL = 3 = 0 для других значений селектора

Описание:
Мультиплексор 4 входов: выбирает одно из четырех целых чисел.
(*FBD пример блока “MUX4”*)




```
(* ST Эквивалент: *)
range := MUX4 (choice, 1, 10, 100, 1000);
```

7.4.68 MUX8



Входы:

SEL	INTEGER	Целый селектор (должен быть в диапазоне [0..7])
IN1..IN8	INTEGER	Целая аналоговая величина

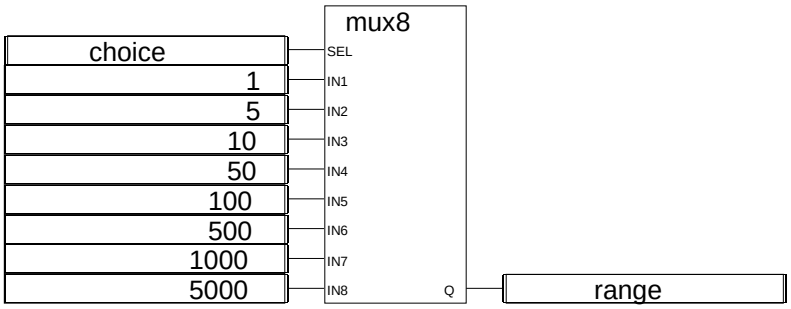
Выход:

Q	INTEGER	= IN1 если SEL = 0 = IN2 если SEL = 1 … = IN8 если SEL = 7 = 0 для других значений селектора

Описание:

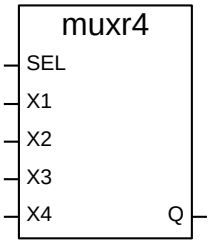
Мультиплексор 8 входов: выбирает одно из восьми целых чисел.

(*FBD пример блока “MUX8”*)



(* ST Эквивалент: *)
range := MUX8 (choice, 1, 5, 10, 50, 100, 500, 1000, 5000);
(* выбирает из 8 предопределенных значений, например, если выбрана 3, значение будет 50 *)

7.4.69 MUXR4



Входы:

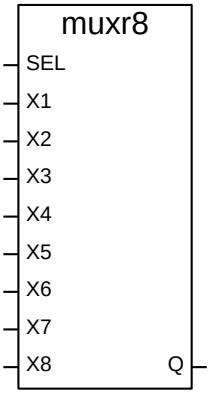
SEL	INTEGER	Целый селектор (должен быть в диапазоне [0..3])
X1..X4	REAL	Действительная аналоговая величина

Выход:

Q	REAL	= X1 если SEL = 0 = X2 если SEL = 1 = X3 если SEL = 2 = X4 если SEL = 3 = 0 для других значений селектора

Описание:
Мультиплексор 4 входов: выбирает одно из четырех действительных чисел.
(* ST Эквивалент: *)
range := MUXR4 (choice, 1.0, 10.0, 100.0, 1000.0);

7.4.70 MUXR8



Входы:

SEL	INTEGER	Целый селектор (должен быть в диапазоне [0..7])
X1..X8	REAL	Целая аналоговая величина

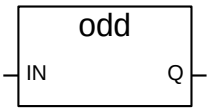
Выход:

Q	REAL	= X1 если SEL = 0 = X2 если SEL = 1 … = X8 если SEL = 7 = 0 для других значений селектора

Описание:
Мультиплексор 8 входов: выбирает одно из восьми действительных чисел.

(* ST Эквивалент: *)
range := MUXR8 (choice, 1.0, 5.0, 10.0, 50.0, 100.0, 500.0, 1000.0, 5000.0);
(* выбирает из 8 предопределенных значений, например, если выбрана 3, значение будет 50.0 *)

7.4.71 ODD



Вход:

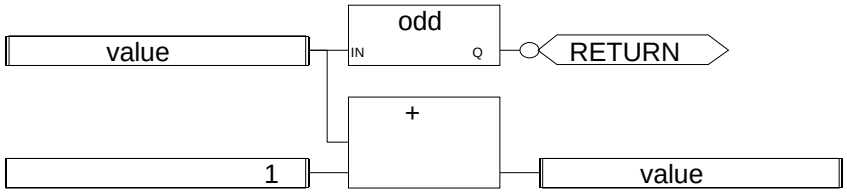
IN	INTEGER	Любая целая знаковая аналоговая величина

Выход:

Q	BOOLEAN	=TRUE, если входная величина нечетная =FALSE, если входная величина четная

Описание:
Проверяет целую величину на четность: результат – признак четный или нечетный.

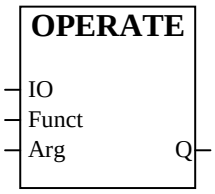
(*FBD пример блока “ODD”*)



(* ST Эквивалент: *)

```
If Not (ODD (value)) Then Return; End_if;
value := value + 1;
(*делает значение четным*)
```

7.4.72 OPERATE, OPERATE_F



Входы:

IO	#Ввод/вывод	Переменная ввода/вывода
Funct	INTEGER	Код запроса
Arg	INTEGER	Аргумент

Выход:

Q	INTEGER	Результат вызова

Назначение:
Тестирование и инициализация юнита/мезонина

ВНИМАНИЕ! ФБ OPERATE используется для совместимости со старыми версиями. Рекомендуется использовать OPERATE_F.

Вызов функции ***operate*** используется для специальных запросов к физическим каналам, которые связаны с переменными ввода/вывода. Функция возвращает значение целого типа и имеет следующий формат: **operate(переменная В/В, code, arg)**, где:

- переменная В/В - переменная ввода/вывода, переменная модульной структуры;
- code - код запроса;
- arg - аргумент, для выполнения запроса.

Примечание.

Применение не указанных в настоящем руководстве запросов не допускается.

В результате вызова OPERATE с неизвестным кодом команды или некорректным номером юнита, функция возвращает нулевое значение. OPERATE возвращает ноль, если юнит с указанным номером не установлен или не поддерживает данную функцию.

Аргумент “Arg” должен быть сброшен в ноль, если не используется командой явно.

Для Мастер-М911Е поддерживается только код **16#0002 (2),** выполняющий проверку достоверности чтения диагностических переменных и переменных ввода/вывода.

Для Мастер-М841Е в результате вызова OPERATE с неизвестным кодом команды, функция возвращает “-1”.

16#0001 (1): Запрос на переинициализацию юнита

Команда производит переинициализацию юнита. Обновление входных переменных для юнитов ввода задерживается на время, необходимое для инициализации юнита. Функция возвращает единицу, если команда выполнена успешно, ноль в противном случае.

Флаг ошибки работы юнита, если был установлен перед вызовом данной функции, сбрасывается только в случае (и после) успешной переинициализации юнита.

16#0002 (2): Чтение состояния юнита / операции ввода/вывода

Функция возвращает код завершения последней операции ввода/вывода по физическому каналу, к которому привязана переменная, указанная в качестве первого аргумента. Анализируя возвращаемый код, можно определить степень достоверности информации, находящейся в переменной ввода/вывода.

Диагностируемые ошибки имеют следующие коды:
Для мастер-модуля **M911E** и для модулей **M800**/**M900**:

- 0 – ошибок не обнаружено;
- 1 – команда не поддерживается модулем (несоответствие версии модуля);
- 2 – неверный формат запроса (несоответствие версии модуля);
- 3 – выполнение запроса недопустимо в текущем режиме работы модуля;
- 4 – выполнение запроса невозможно, модуль остановлен переключателем RUN/STOP;
- 5 – превышение допустимого размера приложения, загрузка невозможна;
- 15 – таймаут при ожидании ответа;
- 255 – выполнение запроса невозможно, модуль занят;
- 300 – перегрузка цепей юнита;
- 400 – короткое замыкание во внешней цепи;
- 500 – обрыв внешней цепи;
- 600 – отсутствие напряжения на внешнем источнике питания;
- 700 – аппаратная ошибка в работе канала;
- 800 – ошибка метрологических констант;
- +1000 – значение не достоверно;
- +2000 – отладочный режим, переменная заблокирована отладчиком;

Для мастер-модулей **M841E/M902E/M921E/M915E/M903E**:

a) Диагностика обмена с модулями ввода/вывода:

- 0 – ошибок не обнаружено;
- +2 – ошибка переполнения UART;
- +4 – ошибка на линии;
- +8 – ошибка адреса (ответ идет с отличным от запроса адресом);
- +16 – ошибочное количество байт данных;
- +32 – повторная стартовая комбинация;
- +64 – ошибка контрольной суммы;
- +128 – таймаут при ожидании очередного байта на приеме;
- +256 – нет параметров (перезапуск модуля)
- +1000 – значение не достоверно;
- +2000 – отладочный режим, переменная заблокирована отладчиком;

b) Диагностика межконтроллерного обмена:

- 0 – ошибок не обнаружено;

Ошибки пакета:

- 101 – превышение размера фрейма;
- 102 – ошибочный формат пакета;
- 103 – ошибка контрольной суммы заголовка;
- 104 – ошибка контрольной суммы пакета;

Ошибки данных:

- 201 – объем данных локального и удаленного узла не совпадает;
- 202 – ошибочная длина данных;
- 203 – ошибочный идентификатор сеанса;
- 204 – ошибочный порядок пакетов;
- 205 – превышение количества пакетов;
- 206 – превышение объема данных;
- 207 – ошибка контрольной суммы;
- 208 – состав переменных локального и удаленного узла не совпадает
- 209 – ошибка внутренних идентификаторов;
- 210 – ошибка режимов работы;

- 1000 – значение не достоверно;

Примечание (относится только к диагностике обмена с модулями M500).
Кроме указанных кодов, для мастер-модулей ***M915E/M903E*** при обмене с модулями ***M500*** функция анализирует поканальную диагностику. Т.е. если в качестве первого аргумента указать переменную, содержащую значение канала, то даже в случае успешного чтения/записи значения функция operate_f может вернуть ненулевое значение, если в поканальной диагностике для данного канала содержится ненулевое значение:

- 256+Err, где Err – значение поканальной диагностики для данного канала.

16#0003 (3): Установка недостоверного значения

Функция устанавливает значение, равное arg, принимаемое переменной, указанной в качестве первого аргумента, при возникновении ошибки выполнения операции ввода/вывода. Используется для булевских и аналоговых переменных. В случае успешного выполнения функция возвращает установленное значение.

16#0004 (4): Отмена установки недостоверного значения

Аргумент “Arg” не используется. Функция отменяет действие вызова функции OPERATE с кодом 3 для переменной, указанной в качестве первого аргумента. При возникновении ошибки выполнения операции ввода/вывода значение переменной остается предыдущим. В результате вызова OPERATE в “Q” прописывается единица.

16#0005 (5): Чтение информации из расширенного состояния модуля

Результат вызова функции зависит от значения аргумента:	
01h(1)	Режим работы
02h(2)	Глобальные флаги ошибок
03h(3)	Критические ошибки приложения
04h(4)	Код сброса
05h(5)	Версия ПО
06h(6)	Тип модуля
07h(7)	Аппаратная конфигурация
08h(8)	Динамические ошибки приложения
09h(9)	Флаги ошибок окружения
0Ah(10)	Аппаратные ошибки работы модулей
0Bh(11)	Ошибки конфигурации модулей
0Ch(12)	Аппаратные ошибки в работе юнитов
0Dh(13)	Ошибки чтения метрологических констант
0Eh(14)	Ошибки внешних цепей юнитов
0Fh(15)	Флаги ошибок связи

16#0006 (6): Чтение/запись модульных переменных с режимом “по запросу”

Функция выполняет действие в соответствии со значением аргумента “Arg”:

1 – Выполнить чтение переменной;
2 – Выполнить запись переменной.

Если присутствует ошибка во входных параметрах (например, попытка чтения/записи переменной с несоответствующим режимом), результат вызова функции будет равен нулю. В случае успешного выполнения функция возвращает единицу.

16#000A (10): Чтение типа юнита (регистры CCR0…CCR1).

Функция возвращает код типа юнита. В случае, когда юнит не установлен или если номер юнита выходит за номинальный диапазон, результат вызова функции будет равен нулю.

16#000B (11): Чтение состояния физических выводов юнита.

Физический интерфейс юнита состоит из четырёх двунаправленных линий D0..D3 и двух входных диагностических линий S0 и S1. Данная команда выполняет чтение непосредственно физических выводов юнита и возвращает маску их состояния. Результат вызова функции:

MSB 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 S1 S0 D3 D2 D1 D0 **LSB**

Если адресуемый юнит не имеет физического интерфейса (например, виртуальные юниты термометра термокомпенсации, юнит состояния и диагностики модуля и т.п.), результат вызова функции будет равен нулю. При возникновении аппаратной ошибки во время доступа к регистрам юнита, результат устанавливается в -1.

16#000C (12): Управление индикатором юнита.

Эта команда разрешает или запрещает управление, а также устанавливает состояние индикатора юнита. Верхние 24 бита аргумента “Arg” не используются и могут принимать любые значения. Младшие 8 бит описаны подробнее:

MSB X X X X X X X X X X X X X X X X X X X X X X X X M1 M0 DF EN S3 S2 S1 S0 **LSB**

Биты S3..S0: Собственно новое состояние светодиодной матрицы индикатора.
Бит EN: Включить/отключить (1/0) управление индикатором.
Бит DF: Разрешить отображение ошибки в работе юнита по умолчанию. Если бит DF установлен, в случае возникновения аппаратной ошибки юнита, контроль над индикатором принимает интеллектуальный модуль и сигнализирует ошибку миганием светодиодов юнита синхронно с красным светодиодом состояния модуля.
Биты M1, M0: Управление режимами мигания светодиодов. 00 – мигание выключено; 01 – мигают включенные светодиоды, а выключенные остаются выключенными; 10 – мигают выключенные светодиоды, включенные остаются включенными; 11 – попеременно зажигаются то включенные, то включенные светодиоды индикатора.
Биты, обозначенные как X: не используются, могут принимать любые значения.

В случае использования приложением модуля энергонезависимой памяти, состояние индикатора, установленное данной функцией OPERATE, не сохраняется.

16#0011 (17): Запрос на переинициализацию юнита с аргументом.

Команда производит переинициализацию юнита. Значение аргумента “Arg” зависит от типа установленного юнита (по умолчанию - ноль). Обновление входных переменных для юнитов ввода задерживается на время, необходимое для инициализации юнита. Функция возвращает единицу, если команда выполнена успешно, ноль в противном случае.
Флаг ошибки работы юнита, если был установлен перед вызовом данной функции OPERATE, сбрасывается только после успешной переинициализации юнита.

16#0015 (21): Блокировка опроса модуля ввода/вывода.

В зависимости от значения аргумента “Arg” функция выполняет следующие действия:

1 – блокировать опрос модуля в/в;
0 – возобновить опрос модуля в/в.

На транзитные запросы блокировка не распространяется. Также, вызов функции с аргументом, равным 1, сбрасывает невыполненные запросы на чтение/запись по всем переменным данного модуля с режимом чтения/записи «по запросу».

В случае успешного выполнения функция возвращает установленное значение.

16#001A (26): Чтение регистров состояния юнита (CCR2…CCR3).

Аргумент “Arg” не используется. Результатом вызова функции является битовая маска состояния выбранного юнита. Значение неописанных здесь битов не всегда равно нулю и зависит от типа юнита. Список общих флагов, находящихся в регистре состояния юнита:

Бит 4: Юнит находится в режиме инициализации. Устанавливается при возникновении аппаратной ошибки или запросом с кодом команды 01h через функцию OPERATE.
Бит 5: Отсутствие напряжения на внешнем источнике питания. Фиксируется при отсутствии напряжения на диагностических выводах юнита.
Бит 6: Обрыв внешней цепи.
Бит 7: Обнаружена аппаратная ошибка работы юнита.
Бит 8: Ошибка метрологических констант канала. Несоответствие существующих констант выбранному типу юнита.
Бит 9: Переменная, подключенная к каналу, заблокирована.
Бит 10: Короткое замыкание внешней цепи.

Полный список флагов зависит от реализации драйвера юнита, а также описание битов регистра состояния и находится в спецификации конкретного юнита.

16#001B (27): Запись специальных регистров юнита (CSR0…CSR3).

16#002A (42): Чтение входных регистров юнита (CIR0…CIR3).

16#002B (43): Запись выходных регистров юнита (COR0…COR3).

16#003A (58): Чтение входных регистров юнита (CIR4…CIR7).

16#003B (59): Запись выходных регистров юнита (COR4…COR7).

16#004A (74): Чтение специальных регистров юнита (CSR0…CSR3).

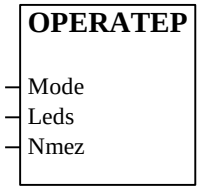
Реализация данной функции, как и возвращаемое вызовом OPERATE значение зависит от типа установленного юнита. Если не сказано иначе, OPERATE возвращает ноль.

16#0064 (100): Конфигурация интерфейса ввода (подробное описание в документе «Unimod Pro. Пульт оператора M920L»).

16#0065 (101): Конфигурация стандартного вывода (подробное описание в документе «Unimod Pro. Пульт оператора M920L»).

Подробная информация о реализации команд записи в регистры должна находиться в спецификации юнита. По умолчанию вызов функции OPERATE возвращает ноль.

7.4.73 OPERATEP



Входы:

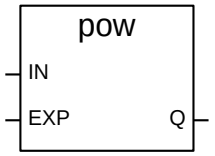
Mode	INTEGER	Режим работы мезонина
Leds	INTEGER	Индикация
NMez	INTEGER	Номер мезонина

Назначение:
Устанавливает режим работы и индикации пожарного юнита/мезонина

Mode - Управление режимом работы
- 16#00 - перейти в нормальный режим работы
- 16#10 - выдавать всегда положительную полярность (поверка)
- 16#20 - дать отрицательную полярность на 3 секунды (сброс датчиков)
- 16#40 - изменить состояние светодиодов

Leds - Индикация светодиодами
- 1 - внимание "сработал 1 датчик"
- 2 - "пожар"
- 4 - дежурный режим "шлейф в порядке"
- 8 - обрыв или КЗ в линии
- 0 - ошибка юнита/мезонина

7.4.74 POW



Входы:

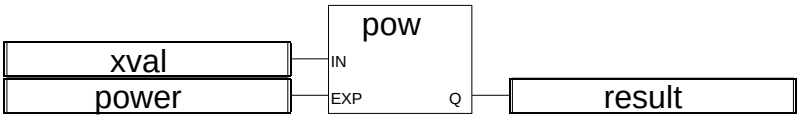
IN	REAL	Знаковая вещественная база
EXP	REAL	Знаковая вещественная степень

Выход:

Q	REAL	Результат функции равен IN в степени EXP

Описание:
Дает вещественной результат операции: (база $^{\text{экспонента}}$) ‘база’ - первый аргумент, экспонента - второй.

(*FBD пример блока “POW”*)




```
(* ST Эквивалент: *)
result := POW (xval, power);
```

7.4.75 PUT_CHAR

Назначение: Функция передаёт байт в пульт оператора M920L.

Входы:

Value	Integer	Значение

Выход:

Q	Integer	Код ошибки

Коды ошибок:

0 - операция выполнена успешно
22 - файл не найден, на модуле не установлен пульта оператора M920L
100 - файл занят, выполняется асинхронный ввод с клавиатуры пульта оператора

7.4.76 PUT_INT

Назначение: Функция производит преобразование целого значения в строку с последующей записью в файл.

Входы:

ID	Integer	Идентификатор файла или консоли
Value	Integer	Выводимое значение

Выход:

Q	Integer	Код ошибки

Коды ошибок:

- 0 - операция выполнена успешно
- 2 - несоответствие атрибутов объекта, ошибка записи в файл
- 12 - файл не открыт, аргумент ID не является идентификатором файла
- 16 - достигнут конец файла, недостаточно места для записи
- 100 - файл занят: выполняется асинхронный ввод с клавиатуры пульта оператора

7.4.77 PUT_MSG

Назначение: Функция производит запись строковой переменной в файл.

Входы:

ID	Integer	Идентификатор файла или консоли
Value	Message	Выводимое значение

Выход:

Q	Integer	Код ошибки

Коды ошибок:

- 0 - операция выполнена успешно
- 2 - несоответствие атрибутов объекта, ошибка записи в файл
- 12 - файл не открыт, аргумент ID не является идентификатором файла
- 16 - достигнут конец файла, недостаточно места для записи
- 100 - файл занят: выполняется асинхронный ввод с клавиатуры пульта оператора

7.4.78 PUT_REAL

Назначение: Функция производит преобразование целого значения в строку с последующей записью в файл.

Входы:

ID	Integer	Идентификатор файла или консоли
Value	Real	Выводимое значение

Выход:

Q	Integer	Код ошибки

Коды ошибок:

- 0 - операция выполнена успешно
- 2 - несоответствие атрибутов объекта, ошибка записи в файл
- 12 - файл не открыт, аргумент ID не является идентификатором файла
- 16 - достигнут конец файла, недостаточно места для записи
- 100 - файл занят: выполняется асинхронный ввод с клавиатуры пульта оператора

7.4.79 PUT_RSRC

Назначение: Функция предназначена для записи данных ресурса в поток ввода/вывода пульта оператора или файл, предварительно открытый функцией FL_OPEN_I.

Входы:

ID	Integer	Идентификатор файла или консоли
Resource	Integer	Идентификатор ресурса

Выход:

Q	Integer	Код ошибки

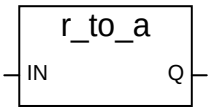
Описание:
Уникальный номер (имя) ресурса передаётся в младшем слове RsrcId.
Опционально, в старших 16 разрядах RsrcId может быть указан тип запрашиваемого ресурса.
Если тип ресурса с заданным именем не совпадёт с запрошенным значением, функция PUT_RSRC возвращает
ошибку "22 - файл не найден".
Вызванная с идентификатором файла CONIO, функция PUT_RSRC служит для асинхронной передачи данных из
ресурса в M920L. При этом не накладывается никаких ограничений ни на тип, ни длину ресурса. Вывод данных ресурса
происходит с текущей позиции курсора.
Вывод ресурсов на пульт оператора может быть использован для:

1) Вывода строк и графики на экран.
2) Передачи специальных команд.
3) Загрузки символов пользователя.

Коды ошибок:

0 - операция выполнена успешно
2 - несоответствие атрибутов объекта, ошибка записи в файл
12 - файл не открыт, аргумент ID не является идентификатором файла
16 - достигнут конец файла, недостаточно места для записи
100 - файл занят: выполняется асинхронный ввод с клавиатуры пульта оператора

7.4.80 R_TO_A



Входы:

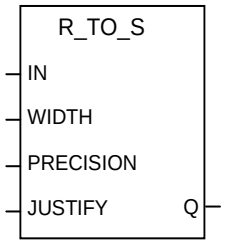
IN	REAL	Вещественное число

Выход:

Q	INTEGER	Целое число

Назначение
Функция используется для копирования вещественного числа в целое.

7.4.81 R_TO_S



Входы:

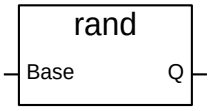
IN	REAL	Вещественное число
WIGHT	INTEGER	Ширина поля для форматирования
PRECISION	INTEGER	Количество знаков после запятой
JUSTIFY	INTEGER	Тип выравнивания (0 - по левому краю, 1 - по центру, 2 - по правому краю)

Выход:

Q	MESSAGE	Строка

Назначение
Функция используется для преобразования вещественного числа в строку с форматированием.

7.4.82 RAND



Вход:

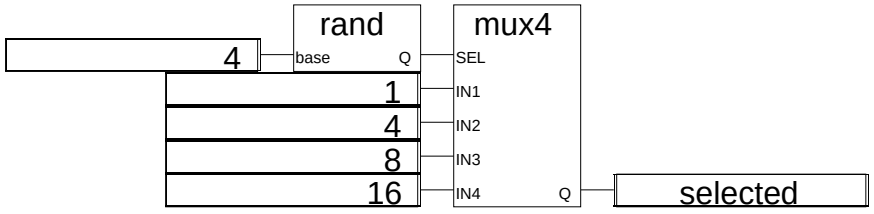
Base	INTEGER	Определяет допустимое множество чисел

Выход:

Q	INTEGER	Случайная величина в диапазоне [0…(base–1)]

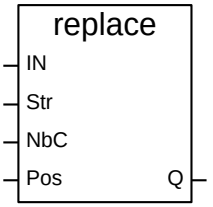
Описание:
Дает случайную целую величину в заданном диапазоне.

(*FBD пример блока “RAND”*)




```
(* ST Эквивалент: *)
selected := MUX4 ( RAND (4), 1, 4, 8, 16 );
(*
случайный выбор одного из 4 предопределенных значений
RAND выдает значение в интервале [0..3],
так 'selected', выходящий из MUX4, получит случайное значение
1 если 0 выходит из RAND,
или 4 если 1 выходит из RAND,
или 8 если 2 выходит из RAND,
или 16 если 3 выходит из RAND*)
```

7.4.83 REPLACE



Входы:

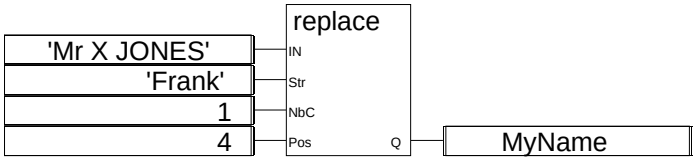
IN	MESSAGE	Любая строка
Str	MESSAGE	Строка, которую нужно вставить
NbC	INTEGER	Количество символов, которые нужно удалить
Pos	INTEGER	Позиция первого измененного символа (первая правильная позиция 1)

Выход:

Q	MESSAGE	Измененная строка: - NbC символов удаляются, начиная с позиции Pos - строка Str вводится, начиная с позиции Pos

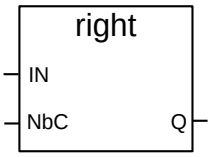
Возвращает пустую строку, если Pos <= 0.
Возвращает соединение строк (IN+Str), если Pos больше, чем длина строки IN
Возвращает начальную строку IN если NbC <= 0

Описание:
Заменяет часть строки новым набором символов.
(*FBD пример блока “REPLACE”*)




```
(* ST Эквивалент: *)
MyName := REPLACE ('Mr X JONES, 'Frank', 1, 4);
(* MyName - это 'Mr Frank JONES' *)
```

7.4.84 RIGHT



Входы:

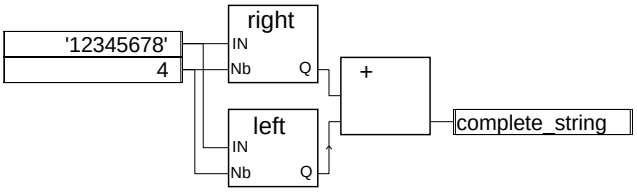
IN	MESSAGE	Любая непустая строка
NbC	INTEGER	Количество символов, которые нужно изъять (не может быть больше строки IN)

Выход:

Q	MESSAGE	Правая часть строки IN (ее длина - NbC) пустая строка, если NbC < 0 строка IN если NbC > длины IN

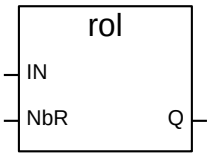
Описание:
Читает правую часть строки. Количество символов задано.

(*FBD пример блоков “LEFT” и “RIGHT”*)



(* ST Эквивалент: *)
complete_string := RIGHT ('12345678', 4) + LEFT ('12345678', 4);
(* полная строка - это '56781234'
значение выходящее из RIGHT - это '5678'
значение выходящее из LEFT - это '1234'*)

7.4.85 ROL



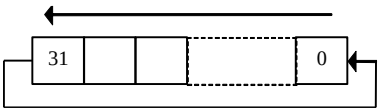
Входы:

IN	INTEGER	Любая целая аналоговая величина
NbR	INTEGER	Количество вращаемых бит (в диапазоне [1..31])

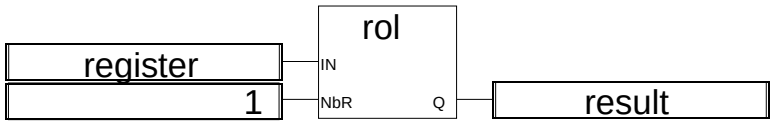
Выход:

Q	INTEGER	Величина с передвинутыми влево битами если NbR <= 0, то нет никакого эффекта

Описание:
Вращает биты влево. Вращаются 32 бита



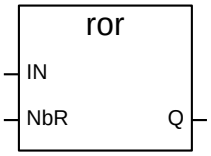
(*FBD пример блока “ROL”*)



(* ST Эквивалент: *)
result := ROL (register, 1);

(* register = 2#0100_1101_0011_0101*)
(* result = 2#1001_1010_0110_1010*)

7.4.86 ROR



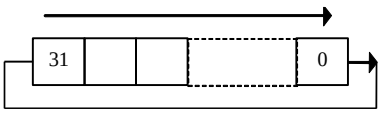
Входы:

IN	INTEGER	Любая целая аналоговая величина
NbR	INTEGER	Количество вращаемых бит (в диапазоне [1..31])

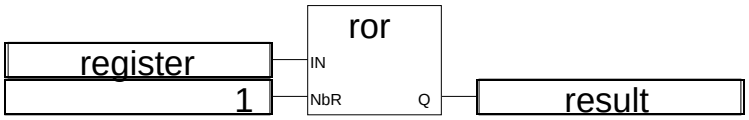
Выход:

Q	INTEGER	Величина с передвинутыми вправо битами если NbR <= 0, то нет никакого эффекта

Описание:
Вращает биты вправо. Вращаются 32 бита:



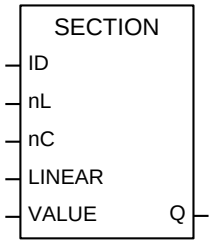
(*FBD пример блока “ROR”*)




```
(* ST Эквивалент: *)
result := ROR (register, 1);

(* register = 2#0100_1101_0011_0101 *)
(* result   = 2#1010_0110_1001_1010 *)
```

7.4.87 SECTION



Входы:

ID	#REAL	Имя массива (элементы типа Real)
nL	INTEGER	Количество строк в массиве
nC	INTEGER	Количество столбцов в массиве
LINEAR	BOOLEAN	Признак кусочно-линейной функции
VALUE	REAL	Входное значение

Выход:

Q	REAL	Выходное значение

Назначение

Функция SECTION вычисляет линеаризованное значение функции Q=f(X) методом кусочно-линейной аппроксимации с использованием узловых точек. Координаты концов отрезков определяются парами Xi, Qi (абсцисса и ордината конца отрезка), которые задаются в массиве.

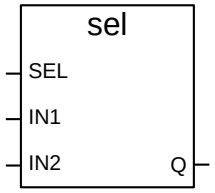
Линеаризованное значение Q вычисляется по следующей формуле:

$$Q = Q_{i-1} + ((Q_i - Q_{i-1}) / (X_i - X_{i-1})) * (X - X_{i-1})$$

где X_i, Q_i – значения в узлах аппроксимации.

При нулевом значении на входе LINEAR формируется кусочно-ступенчатая функция.

7.4.88 SEL



Входы:

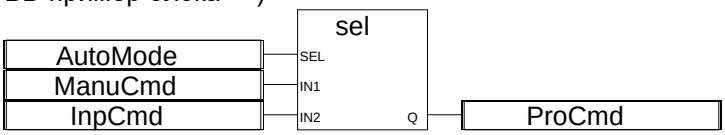
SELECT	BOOLEAN	Определяет выбранную величину
IN1, IN2	INTEGER	Любая целая аналоговая величина

Выход:

Q	INTEGER	= IN1, если SEL = FALSE
		= IN2, если SEL = TRUE

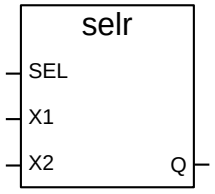
Описание:
Двоичный селектор: выбирает значение из двух целых чисел.

(*FBD пример блока “”*)



(* ST Эквивалент: *)
ProCmd := SEL (AutoMode, ManuCmd, InpCmd);
(* выбор команды *)

7.4.89 SELR



Входы:

SEL	BOOLEAN	Определяет выбранную величину
X1, X2	REAL	Любая целая аналоговая величина

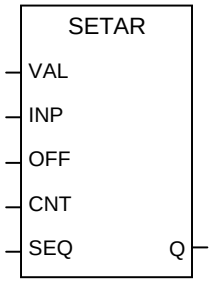
Выход:

Q	REAL	= X1, если SEL = FALSE
		= X2, если SEL = TRUE

Описание:
Двоичный селектор: выбирает значение из двух целых чисел.

(* ST Эквивалент: *)
ProCmd := SELR (AutoMode, ManuCmd, InpCmd);
(* выбор команды *)

7.4.90 SETAR



Входы:

VAL	INTEGER	Устанавливаемое значение
INP	#ANY	Идентификатор массива
OFF	INTEGER	Смещение (в байтах) в массиве
CNT	INTEGER	Количество байтов для установки
SEQ	INTEGER	Порядок байтов:

0 по умолчанию
1 "0-1" - 2х байтовое
2 "1-0" - 2х байтовое, замена байтов
3 "0-1-2" - 3х байтовое
4 "2-1-0" - 3х байтовое, замена байтов
5 "0-1-2-3" - 4х байтовое
6 "1-0-3-2" - 4х байтовое, замена байтов
7 "2-3-0-1" - 4х байтовое, замена слов
8 "3-2-1-0" - 4х байтовое, замена байтов, замена слов

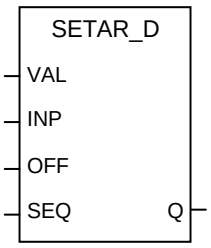
Выход:

Q	INTEGER	Устанавливаемое значение (в случае ошибки - 0)

Назначение

Функция SETAR используется для установки группы байтов в массиве. Содержимое остальных байтов массива не изменяется.

7.4.44 SETAR_D



Входы:

VAL	INTEGER	Устанавливаемое значение
INP	#ANY	Идентификатор массива
OFF	INTEGER	Смещение (в байтах) в массиве
CNT	INTEGER	Количество байтов для извлечения
SEQ	INTEGER	Порядок байтов:

0 “0-1-2-3-4-5-6-7” – копирование без замен
1 "1-0-3-2-5-4-7-6" – замена байтов
2 "2-3-0-1-6-7-4-5" – замена слов
3 "3-2-1-0-7-6-5-4" – замена байтов, замена слов
4 “4-5-6-7-0-1-2-3” – замена двойных слов
5 "5-4-7-6-1-0-3-2" – замена двойных слов, замена байтов
6 "6-7-4-5-2-3-0-1" – замена двойных слов, замена слов
7 "7-6-5-4-3-2-1-0" – замена двойных слов, замена байтов, замена слов

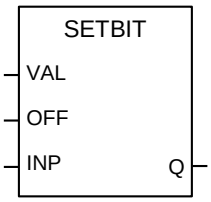
Выход:

Q	INTEGER	Устанавливаемое значение (в случае ошибки - 0)

Назначение

Функция SETAR_D используется для установки вещественного значения двойной точности в массиве. Содержимое остальных байтов массива не изменяется.

7.4.91 SETBIT



Входы:

VAL	BOOLEAN	Значение бита
OFF	INTEGER	Смещение на бит в переменной [0÷31]
INP	INTEGER	Значение переменной до операции

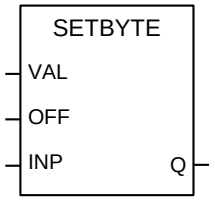
Выход:

Q	INTEGER	Значение переменной после операции

Назначение

Функция SETBIT используется для установки бита в переменной целого типа. Значение остальных бит переменной не изменяется.

7.4.92 SETBYTE



Входы:

VAL	INTEGER	Значение байта
OFF	INTEGER	Смещение на байт в переменной [0÷3]
INP	INTEGER	Значение переменной до операции

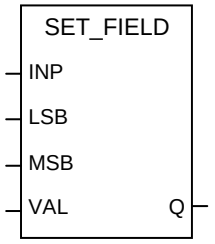
Выход:

Q	INTEGER	Значение переменной после операции

Назначение

Функция SETBYTE используется для установки байта в переменной целого типа. Значение остальных байтов переменной не изменяется.

7.4.93 SETFIELD



Входы:

INP	INTEGER	Значение переменной до операции
LSB	INTEGER	Младший бит в переменной [0-31]
MSB	INTEGER	Старший бит в переменной [0-31]
VAL	INTEGER	Устанавливаемое значение

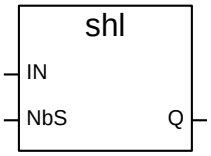
Выход:

Q	INTEGER	Значение переменной после операции

Назначение

Функция SETFIELD используется для установки группы бит в переменной целого типа. Значение остальных бит переменной не изменяется.

7.4.94 SHL



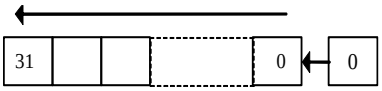
Входы:

IN	INTEGER	Любая целая аналоговая величина
NbS	INTEGER	Количество сдвигаемых бит (в диапазоне [1..31])

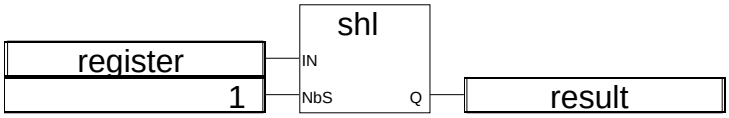
Выход:

Q	INTEGER	Сдвинутая влево величина если NbR <= 0, то нет никакого эффекта при сдвиге младший бит заменяет нулем

Описание:
Сдвигает биты влево. Сдвигаются 32 бита.



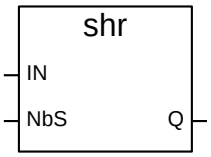
(*FBD пример блока “SHL”*)




```
(* ST Эквивалент: *)
result := SHL (register,1);

(* register = 2#0100_1101_0011_0101 *)
(* result   = 2#1001_1010_0110_1010 *)
```

7.4.95 SHR



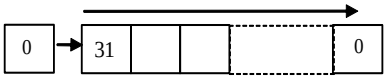
Входы:

IN	INTEGER	Любая целая аналоговая величина
NbS	INTEGER	Количество сдвигаемых бит (в диапазоне [1..31])

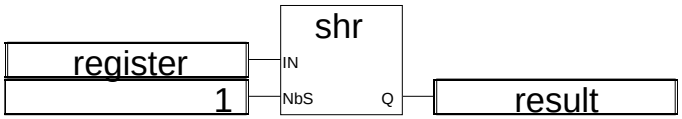
Выход:

Q	INTEGER	Сдвинутая вправо величина если NbR <= 0, то нет никакого эффекта при сдвиге старший бит заменяет нулем

Описание:
Сдвигает биты вправо. Сдвигаются 32 бита.



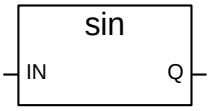
(*FBD пример блока “SHR”*)




```
(* ST Эквивалент: *)
result := SHR (register,1);

(* register = 2#1100_1101_0011_0101 *)
(* result    = 2#1110_0110_1001_1010 *)
```

7.4.96 SIN



Вход:

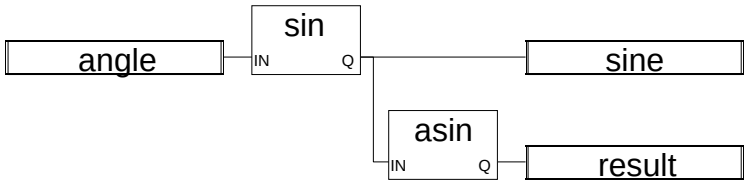
IN	REAL	Вещественная величина

Выход:

Q	REAL	Синус входа в диапазоне [-1.0..+1.0]

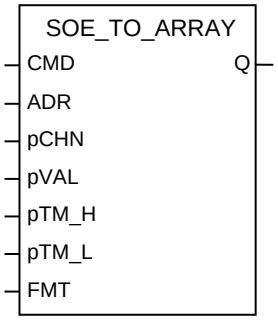
Описание:
Вычисляет синус вещественной величины.

(*FBD пример блока “SIN” и “ASIN”*)




```
(* ST Эквивалент: *)
sine := SIN (angle);
result := ASIN (sine); (*результат равен углу *)
```

7.4.97 SOE_TO_ARRAY



Вход:

CMD	BOOLEAN	TRUE - Выполнить привязку, FALSE - Отменить
ADR	INTEGER	Адрес модуля
pCHN	#INTEGER	Массив с номерами каналов (1..N)
pVAL	#ANY	Ссылка на массив для хранения значений
pTM_H	#INTEGER	Ссылка на массив для старшей части времени
pTM_L	#INTEGER	Ссылка на массив для младшей части времени
FMT	INTEGER	Формат меток времени: 0 - Unimod 1 - Unix

Выход:

Q	INTEGER	Результат выполнения операции 0 - ошибок нет 1 - неизвестный модуль 2 - неверно заданы входные параметры

Назначение:
Привязка модуля с регистрацией событий к массивам

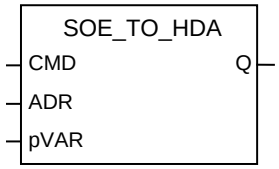
Описание:
Функция используется при необходимости хранить архивы HDA не только локально, но и передавать, например по межконтроллерному обмену или при резервировании.

Функция выполняет логическое связывание модуля с регистрацией событий (например M557DR) и массивов для хранения прочитанных данных.
После выполнения функции массив pVAL будет заполняться значениями, полученными с модуля с адресом ADR. Массив pCHN будет заполняться номерами каналов, по которым получены события. Массивы pTM1 и pTM2 будут заполняться метками времени, формат меток времени задается аргументом FMT зависимости от аргумента FMT:

При FMT равном 0 (формат Unimod):
pTM_H - дата в формате Unimod
pTM_L - время в формате Unimod
При FMT равном 1 (формат Unix):
pTM_H - кол-во секунд от начала 1970 года
pTM_L - кол-во миллисекунд от начала секунды

Примечание
Регистрация событий на модуле начинается только после синхронизации времени модуля с временем мастера (см. system(1,0))

7.4.98 SOE_TO_HDA



Вход:

CMD	BOOLEAN	TRUE - Выполнить привязку, FALSE - Отменить
ADR	INTEGER	Адрес модуля
pVAR	#ANY	Массив переменных HDA

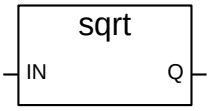
Выход:

Q	INTEGER	Результат выполнения операции (0 - ошибок нет): 1 - неизвестный модуль 2 - неверный тип или размер массива pVAR 8 - системная ошибка

Назначение:
Привязка модуля с регистрацией событий к переменной HDA

Описание:
Логическое связывание переменных массива pVAR и модуля с регистрацией событий (например M557DR). Кол-во элементов в массиве pVAR должно соответствовать кол-ву каналов модуля ввода-вывода.
После выполнения функции переменные pVAR будут заполнять архив HDA событиями, полученными с модуля с адресом ADR. Для переменных pVAR в словаре должна быть включена архивация.
Примечание
Регистрация событий на модуле начинается только после синхронизации времени модуля с временем мастера (см. system(1,0))

7.4.99 SQRT



Вход:

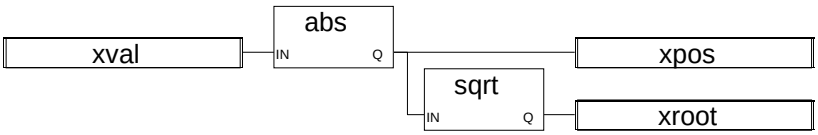
IN	REAL	Вещественная величина, должна быть больше 0

Выход:

Q	REAL	Квадратный корень входной величины

Описание:
Вычисляет квадратный корень действительной величины.

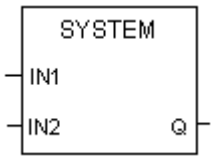
(*FBD пример блока “SQRT”*)




```
(* ST Эквивалент: *)
xpos := ABS (xval);
xroot := SQRT (xpos);
```

7.4.100 SYSTEM

Внимание! Применение не указанных в настоящем руководстве запросов не допускается!



Входы:

IN1	INTEGER	код запроса
IN2	INTEGER	аргумент, для выполнения запроса

Выход:

Q	INTEGER	Ответ на запрос

Назначение
Системная функция, используется для выполнения специальных запросов к исполнительной системе

Вызов функции SYSTEM с неизвестным кодом запроса или некорректным аргументом возвращает **ноль**.

16#0001 (1): Установка времени на модуле (модулях)

	Мастер-ПК	M841E M921E M902E M915E M903E	M911	M900/M800
аргумент = 0 Широковещательная установка времени аргумент = X Установка времени на модуле с адресом <X>	•	•	•	

16#0002 (2): Чтение таймаута обмена с мастер-модулем.

	Мастер-ПК	M841E M921E M902E M915E	M911	M900/M800
Аргумент не используется и должен быть равен нулю. Функция возвращает значение интервала времени в миллисекундах, истёкшее от момента получения последнего корректного пакета от мастер-модуля. Команда применима для контроля наличия связи между модулем и мастером или для анализа активности шины ST-BUS. Расчет интервала производится с дискретностью цикла приложения.				•
Чтение времени от последнего запроса с верхнего уровня: arg =1 Время от последнего запроса полученного через ECAT (сек); arg =2 Время от последнего запроса полученного через MK-UART (сек).			•	

16#0003 (3): Чтение списка динамических ошибок выполнения приложения.

		Мастер-ПК	M841E M921E M902E M915E M903E	M911	M900/M800
Аргумент не используется и должен быть равен нулю. Детализация динамических ошибок выполнения приложения:					
Бит 0	Целочисленное деление на ноль	•	•	•	•
Бит 1	Переполнение в результате преобразовании “вещественного” типа в “целый”	•	•	•	•
Бит 2	FPU: Деление на ноль	•	•	•	•
Бит 3	FPU: Неверный формат	•	•	•	
Бит 4	FPU: Переполнение результата вычисления в плюс бесконечность	•	•	•	•
Бит 5	FPU: Переполнение результата вычисления в минус бесконечность	•	•	•	•
Бит 6	FPU: Ошибка доминанты				•
Бит 7	FPU: Зарезервировано на будущее				•
Бит 8	TIME: Переполнение активной таймерной переменной				•
Бит 9	TIME: Переполнение переменной в следствии математических операций				•
Бит 10	TIME: Запись неверного значения через словарь обмена				•
Примечание: Пакет ответа на расширенный опрос состояния не содержит старший байт динамических ошибок выполнения приложения (биты 8..10 включительно). Т.е. флаги ошибок таймерных переменных доступны исключительно со стороны технологической программы модуля, но при желании могут быть экспортированы на верхний уровень через переменную в словаре обмена модуля					

16#0004 (4): Чтение кода запуска модуля

		Мастер-ПК	M841E M921E M902E M915E M903E	M911	M900/M800
arg =0 Чтение кода запуска модуля arg =1 Признак перезапуска модуля Обозначены следующие значения кода сброса/запуска приложения:					
00h	Нормальный режим работы модуля.	•	•	•	•
01h	Нормальное включение питания.		•	•	
02h	Остановка модуля тумблером RUN/STOP		•	•	
03h	Остановка приложения внешним запросом		•	•	
04h	Перезапуск приложения по таймеру Watchdog	•		•	•
05h	Остановка работы из-за внутренней ошибки приложения.	•	•	•	•
10h	Кратковременный провал питания.		•	•	
11h	Переполнение аппаратного таймера Watchdog.		•	•	
12h	Отключение питания во время режима SLEEP.			•	
13h	Переполнение стека микроконтроллера (аппаратное).			•	
14h	Критическая ошибка выполнения программы	•		•	•
15h	Отсутствие системного сигнала CLK				
1Fh	Не удалось установить источник аппаратного сброса микроконтроллера.			•	
20h	Некорректное выполнение ветвления в коде программы.		•	•	
21h	Сработало неизвестное прерывание			•	

22h	Недопустимые изменения в регистрах микроконтроллера.			•	
23h	Вызов исключения без установленного обработчика.			•	
28h	Полный перезапуск модуля, так как перезагрузка матрицы не удалась.			•	
29h	Перезагрузка матрицы: некорректный ответ по шине PT-BUS.			•	
2Ah	Потеря конфигурации матрицы Acex с последующей перезагрузкой.			•	

16#0005 (5): Чтение флагов ошибок питания модуля.

		Мастер-ПК	M841E M921E M902E M915E	M911	M900/M800
Результат вызова данной функции зависит от значения аргумента:					
00h	Функция возвращает флаги ошибок напряжений питания модуля:				
Бит 0	Разряд аккумулятора подпитки часов реального времени				
Бит 1	Критическое падение напряжения питания 2,4 V				
Бит 2	Критическое падение напряжения питания 3,3 V			•	
Бит 3	Критическое падение напряжения питания 5 V				
Бит 4	Критическое падение напряжения питания 24 V			•	
01h	Минимальное значение напряжения питания 3,3 V, милливольт				
02h	Максимальное значение напряжения питания 3,3 V, милливольт				
	Измеренное значение напряжения питания 3,3 V, милливольт			•	
03h	Минимальное значение напряжения питания 5 V, милливольт				
04h	Максимальное значение напряжения питания 5 V, милливольт				
	Измеренное значение напряжения питания 24 V, милливольт			•	
05h	Текущее значение напряжения питания 3,3 V, милливольт				
06h	Текущее значение напряжения питания 5 V, милливольт				
Примечание: Физическое значение напряжения питания (U) возвращается как целое число (RES) и рассчитывается по формуле: RES := **real_to_int** (U * 1000.0); Для получения значения напряжения VOL в вещественном формате необходимо произвести обратное преобразование: VOLout := **int_to_real** (system (5, VOL)) / 1000.0; На модуле M953C отсутствует поддержка измерения напряжений питания 2,4 и 5 V и функция, вызванная с аргументом 01h или 03h, возвращает -1					

16#0006 (6): Чтение значения температуры окружающей среды.

		Мастер-ПК	M841E M921E M902E	M911	M900/M800
Возвращаемая величина – температура в градусах Цельсия, зависит от значения аргумента:					
00h	Чтение флагов температуры окружающей среды Возвращаемая величина имеет следующую структуру			•	
Бит 5	Температура ниже нормы			•	
Бит 6	Температура выше нормы			•	
01h	Температура процессора		•		
	Температура окружающей среды			•	
02h	Нижний предел температуры			•	

03h	Верхний предел температуры			•	
Примечание: Значение температуры процессора имеет вещественный формат, и для корректного отображения это значение необходимо преобразовать функцией *A_TO_R()*.			•		
Примечание: Измерение температуры на модуле производится датчиком температуры, его наличие зависит от исполнения (аппаратной конфигурации) модуля. Значение температуры окружающей среды возвращается в градусах Цельсия без действительной части. Для получения боле точного значения следует использовать переменную, привязанную к юниту датчика температуры.					•

16#0007 (7): Аппаратные ошибки работы модулей.

		Мастер-ПК	M841E M921E M902E M915E	M911	M900/M800
Аргумент не используется и должен быть равен нулю. Детализация аппаратных ошибок модулей:					
Бит 0	Ошибка работы сетевого адаптера WIZNET				
Бит 1	Ошибка работы дополнительного контроллера PIC				
Бит 2	Ошибка работы внешней памяти FLASH			•	
Бит 3	Ошибка работы энергонезависимой памяти SRAM/FRAM			•	
Бит 4	Ошибка чтения или установки времени в RTC			•	
Бит 5	Произошла перезагрузка матрицы ACEX			•	
Бит 6	Резерв				
Бит 7	Резерв				
Бит 8	Ошибка работы шины PT-BUS на модуле расширения 1			•	
Бит 9	Ошибка работы шины PT-BUS на модуле расширения 2			•	
Бит 10	Ошибка работы шины PT-BUS на модуле расширения 3			•	
Бит 11	Ошибка работы шины PT-BUS на модуле расширения 4			•	
Бит 12	Ошибка работы памяти ППЗУ на модуле расширения 1			•	
Бит 13	Ошибка работы памяти ППЗУ на модуле расширения 2			•	
Бит 14	Ошибка работы памяти ППЗУ на модуле расширения 3			•	
Бит 15	Ошибка работы памяти ППЗУ на модуле расширения 4			•	
Бит 16	Ошибка работы термометра на модуле расширения 1			•	
Бит 17	Ошибка работы термометра на модуле расширения 2			•	
Бит 18	Ошибка работы термометра на модуле расширения 3			•	
Бит 19	Ошибка работы термометра на модуле расширения 4			•	
Бит 20	Резерв				
Бит 21	Резерв				
Бит 22	Резерв				
Бит 23	Резерв				

16#0008 (8): Ошибки конфигурации модулей

		Мастер-ПК	M841E M921E M902E M915E	M911	M900/M800
Аргумент не используется и должен быть равен нулю. Детализация ошибок конфигурации модулей:					
Бит 0	Несовпадение CRC массива конфигурации				
Бит 1	Неверный MAC адрес (0000h или 0FFFFh)				
Бит 2	Резерв				
Бит 3	Резерв				
Бит 4	Несовпадение CRC массива конфигурации модуля расширения 1			•	

Бит 5	Несовпадение CRC массива конфигурации модуля расширения 2			•	
Бит 6	Несовпадение CRC массива конфигурации модуля расширения 3			•	
Бит 7	Несовпадение CRC массива конфигурации модуля расширения 4			•	

16#0009 (9): Аппаратные ошибки работы юнитов.

	Мастер-ПК	M841E M921E M902E M915E	M911	M900/M800
Аргумент не используется и должен быть равен нулю. Оператор SYSTEM возвращает 32 флага, по одному на каждый юнит			•	•

16#000A (10): Чтение ошибок метрологических констант.

	Мастер-ПК	M841E M921E M902E M915E	M911	M900/M800
Аргумент не используется и должен быть равен нулю. Оператор SYSTEM возвращает 32 флага, по одному на каждый юнит			•	•

16#000B (11): Чтение ошибок внешних цепей юнитов.

	Мастер-ПК	M841E M921E M902E M915E	M911	M900/M800
Аргумент не используется и должен быть равен нулю. Оператор SYSTEM возвращает 32 флага, по одному на каждый юнит			•	•

16#000C (12): Чтение массива ошибок связи.

		Мастер-ПК	M841E M921E M902E M915E	M911	M900/M800
Аргумент не используется и должен быть равен нулю. Детализация ошибок связи:					
Бит 0	Переполнение буфера приёмника UART			•	
Бит 1	Обнаружена ошибка фрейма при приёме байта			•	
Бит 2	Превышение длины принимаемого пакета			•	
Бит 3	Несовпадение контрольной суммы пакета			•	
Бит 4	Ошибка при приёме данных пакета			•	
Бит 5	Активность линии ST-BUS во время обработки пакета			•	
Бит 6	Резерв;				
Бит 7	Резерв;				
Бит 8	Резерв;				
Бит 9	Резерв;				
Бит 10	Резерв;				
Бит 11	Резерв;				
Бит 12	Резерв;				
Бит 13	Резерв;				
Бит 14	Резерв;				

Бит 15	Резерв;				
Бит 16	RS232/RS485/BLUETOOTH: Аппаратные ошибки				
Бит 17	RS232/RS485/BLUETOOTH: Несовпадение контр. суммы пакета				
Бит 18	RS232/RS485/BLUETOOTH: Неверный формат пакета				
Бит 19	RS232/RS485/BLUETOOTH: Таймаут связи				
Бит 20	Резерв;				
Бит 21	Резерв;				
Бит 22	Резерв;				
Бит 23	Резерв;				
Бит 24	WIZNET: Аппаратные ошибки			•	
Бит 25	WIZNET: Несовпадение контрольной суммы пакета			•	
Бит 26	WIZNET: Неверный формат пакета			•	
Бит 27	WIZNET: Таймаут связи			•	

16#000E (14): Включить звуковой сигнал.

		Мастер-ПК	M841E M921E M902E M915E	M911	M900/M800
Результат вызова оператора SYSTEM зависит от значения аргумента:					
00h	Выключить звуковой сигнал				
01h	Включить звуковой сигнал				

16#000F (15): Чтение состояния переключателей модуля.

		Мастер-ПК	M841E M921E M902E M915E	M911	M900/M800
Результат вызова оператора SYSTEM зависит от значения аргумента:					
00h	Состояние переключателя скорости ST-BUS RATE.			•	
01h	Состояние переключателя адреса модуля ADR-L.			•	
02h	Состояние переключателя адреса модуля ADR-H.			•	
03h	Состояние джемпера запрета записи памяти FLASH. Вызов SYSTEM возвращает **единицу**, если джемпер установлен (запрет записи), или **ноль**, если джемпер снят. В случае возникновения ошибки доступа к памяти (аппаратная неисправность памяти FLASH или её отсутствие в данной конфигурации модуля), оператор возвращает **-1**. **Внимание:** данная команда зарезервирована исключительно для использования в производственных проверках модуля. Выполнение команды производит пробную запись в регистр состояния памяти FLASH, уменьшая тем самым продолжительность жизни микросхемы.			•	

16#0011 (17): Чтение статистики работы (**мастер-модуля**).

		Мастер-ПК	M841E M921E M902E M915E M903E	M911	M900/M800
Результат вызова оператора SYSTEM зависит от значения аргумента:					
00h(0)	Общее время наработки микроконтроллера, в сек.		•	•	
01h(1)	Время наработки в режиме выполнение приложения, в сек.		•	•	
02h(2)	Количество сбросов по выключению питания			•	
03h(3)	Время последнего сброса по выключению питания			•	
04h(4)	Дата последнего сброса по выключению питания			•	
05h(5)	Количество сбросов по отсутствию SYSCLK			•	
06h(6)	Время последнего сброса по отсутствию SYSCLK			•	
07h(7)	Дата последнего сброса по отсутствию SYSCLK			•	
08h(8)	Количество сбросов по WDT			•	
09h(9)	Время последнего сброса по WDT			•	
0Ah(10)	Дата последнего сброса по WDT			•	
0Bh(11)	Количество сбросов по неустановленной причине			•	
0Ch(12)	Время последнего сброса по неустановленной причине			•	
0Dh(13)	Дата последнего сброса по неустановленной причине			•	
0Eh(14)	Количество сбросов по провалу питания			•	
0Fh(15)	Тип последнего сброса по провалу питания. Возвращает: 1 - напряжение 3,3 V ниже нормы 2 - напряжение 3,3 V выше нормы 3 - напряжение 5,0 V ниже нормы 4 - напряжение 5,0 V выше нормы 5 - напряжение 2,5 V отсутствует			•	
10h(16)	Время последнего сброса по провалу питания			•	
11h(17)	Дата последнего сброса по провалу питания			•	
12h(18)	Количество сбросов по сбою в ACEX			•	
13h(19)	Время последнего сброса по сбою в ACEX			•	
14h(20)	Дата последнего сброса по сбою в ACEX			•	
15h(21)	Количество сбросов по смене конфигурации			•	
16h(22)	Время последнего сброса по смене конфигурации			•	
17h(23)	Дата последнего сброса по смене конфигурации			•	
18h(24)	Количество выходов за пределы рабочей температуры			•	
19h(25)	Тип последнего выхода за пределы рабочей температуры Возвращает: 1- меньше минимально допустимой 2- больше максимально допустимой			•	
1Ah(26)	Время последнего выхода за пределы рабочей температуры			•	
1Bh(27)	Дата последнего выхода за пределы рабочей температуры			•	
1Ch(28)	Время последнего возврата в нормальный температурный режим			•	
1Dh(29)	Дата последнего возврата в нормальный температурный режим			•	

16#0011 (17): Чтение массива статистики (**интеллектуального модуля**)

№ элемента	Описание	Блок
1	Общее время наработки модуля, секунд.	Заголовок
2	Время наработки модуля в режиме выполнения технологической программы, секунд.	
3	Общее количество сбросов модуля.	
4	Количество обнаруженных ошибок каналов ввода/вывода.	

5	1-й элемент регистратора событий	FIFO
6	2-й элемент регистратора событий	
…	…	
36	32-й элемент регистратора событий	

В каждый из 32-х элементов регистратора событий входит код события (код сброса модуля) и опционально код ошибки технологической программы. Регистрируются также дата и время события. Следует отметить, что если в аппаратной конфигурации модуля отсутствуют часы реального времени (RTC), поле даты события примет нулевое значение, а в поле ‘время’ будет прописан порядковый номер сброса модуля.

№ элемента	Описание
1	Код сброса модуля или код события.
2	Код ошибки технологической программы.
3	Время происхождения или порядковый номер события.
4	Дата происхождения события или ноль.

16#0012 (18): Чтение параметров времени исполнения приложения

		Мастер-ПК	M841E M921E M902E M915E M903E	M911	M900/M800
Результат вызова оператора SYSTEM зависит от значения аргумента:					
00h(0)	Длительность минимального цикла		•	•	
01h(1)	Длительность максимального цикла		•	•	
02h(2)	Длительность предыдущего цикла		•	•	
03h(3)	Длительность цикла в режиме отладки (задание)		•	•	
04h(4)	Длительность минимального цикла технологического приложения		•		
05h(5)	Длительность максимального цикла технологического приложения		•		
06h(6)	Длительность предыдущего цикла технологического приложения		•		
07h(7)	Длительность предыдущего цикла задачи межконтроллерного обмена		•		

Для кода 7 номер задачи задается вторым байтом. Например:
system(18, 16#107) - получить время выполнения для задачи 1.

16#0013 (19): Чтение параметров технологического приложения

		Мастер-ПК	M841E M921E M902E M915E M903E	M911	M900/M800
Результат вызова оператора SYSTEM зависит от значения аргумента:					
01h(1)	Размер приложения		•		
02h(2)	Контрольная сумма приложения		•		
03h(3)	Версия Unimod Pro		•		
04h(4)	Дата (кол-во секунд с 1970г) (мл. часть)		•		
05h(5)	Дата (кол-во секунд с 1970г) (ст. часть)		•		
06h(6)	Идентификатор сборки приложения		•		
07h(7)	Фиксированный цикл приложения		•		
08h(8)	Режим исполнения приложения		•		

16#0014 (20): Чтение ошибок встроенного коммуникационного адаптера (ВКА) (мастер-модуль **M911E**).

Аргумент	Байт 3	Байт 2	Байт 1	Байт 0
0			Тип протокола	Версия ПО
1	Ошибки МК	Ошибки ВКА	Ошибки ПП	Состояние
2 (с обнулением)	Ошибки МК	Ошибки ВКА	Ошибки ПП	Состояние

Ошибки МК – ошибки на системной шине, определяемые со стороны микроконтроллера (МК)
- 0-й бит тайм-аут записи
- 1-й бит тайм-аут чтения
- 2-й бит неправильный формат пакета
- 3-й бит несовпадение контрольной суммы

Ошибки ВКА – ошибки на системной шине определяемые со стороны ВКА
- 0-й бит – переполнение приемного буфера
- 1-й бит – неправильное направление передачи данных
- 2-й бит – неизвестная команда
- 3-й бит – ошибки массива конфигурации
- 4-й бит – недопустимая команда
- 5-й бит – ошибка контрольной суммы
- 6-й бит – логическая ошибка пакета

Ошибки ПП - ошибки на мультимастерной (RS485) шине при приеме/передаче (ПП) данных
- 0-й бит – аппаратные ошибки
- 1-й бит – ошибки CRC
- 2-й бит – ошибки формата
- 3-й бит – ошибки таймаута на ответ

16#0015 (21): Установка таймаута связи с верхним уровнем через MK-UART (сек) (мастер-модуль **M911E**).
16#0016 (22): Установка таймаута связи с верхним уровнем через ECAT (сек) (мастер-модуль **M911E**).

Только для M911E (для других мастер-модулей описание приведено ниже)

Аргумент – значение таймаута 0-65535. Нулевое значение отключает счетчик таймаута.
По умолчанию значение таймаута для MK-UART равно нулю, для ECAT – 60 сек.
При отсутствии запросов через MK-UART/ECAT в течение заданного таймаута производится запись в лог-файл.
Также устанавливаются флаг «Таймаут связи на MK-UART»/«Таймаут связи на ECAT» в байте «ошибки связи» (читается через запрос расширенного состояния).

16#0015 (21): Конфигурация и статистика обмена по шине ST-BUS1 (мастер-модули **M841E/M902E/M921E**)
16#0016 (22): Конфигурация и статистика обмена по шине ST-BUS2 (мастер-модули **M841E/M902E/M921E**)

Только для M841E/M902E/M921E (для других мастер-модулей описание приведено ниже)

В зависимости от значения аргумента “Arg” позволяет получать кол-во отправленных/принятых пакетов или для определенной шины ST-BUS, или для определенного модуля.

Аргумент *"Arg"* имеет следующий формат:

Байт	3	2	1	0
Назначение	Резерв, должен быть равен 0	Номер модуля	Резерв, должен быть равен 0	Код запроса

Кол-во отправленных/принятых пакетов для указанной шины ST-BUS

Если байт *"Номер модуля"* равен 0, то SYSTEM позволяет читать статистику для всей шины ST-BUS, в зависимости от кода запроса.

Код запроса	
00h(0)	Чтение текущего режима Возвращает: 1 - режим тестирования 2 - прием пакетов только по первой линии 3 - прием пакетов только по второй линии 4 - прием по обеим линиям
01h(1)	Режим тестирования
02h(2)	Принимать пакеты только по первой линии
03h(3)	Принимать пакеты только по второй линии
04h(4)	Прием по обеим линиям
05h(5)	Сброс в исходное состояние (режим тестирования, обнуление счетчиков)
06h(6)	Количество отправленных пакетов
07h(7)	Количество успешно принятых ответных пакетов по линии 1
08h(8)	Количество успешно принятых ответных пакетов по линии 2
09h(9)	Количество ошибок по линии 1
0Ah(10)	Количество ошибок по линии 2

Коды запроса 0…4 имеют значение, если включен режим тестирования линий каждой шины ST-BUS (M) (см. документ *«Unimod Pro. Руководство пользователя»*, раздел – *Программа диагностики UMDiag*, пункт *Редактор конфигурации*). Если режим тестирования отключен, то счетчики успешно принятых пакетов и ошибок разделения по линиям иметь не будут.

Примеры вызовов:

Ret_Val:=SYSTEM(21, 0) – чтение текущего режима работы шины ST-BUS1;

Ret_Val:=SYSTEM(22, 0) – чтение текущего режима работы шины ST-BUS2;

Ret_Val:=SYSTEM(21, 5) – сброс всех счетчиков для шины ST-BUS1;

Ret_Val:=SYSTEM(22, 5) – сброс всех счетчиков для шины ST-BUS2;

Ret_Val:=SYSTEM(21, 10) – кол-во ошибок по линии 2 шины ST-BUS1;

Ret_Val:=SYSTEM(22, 10) – кол-во ошибок по линии 2 шины ST-BUS2;

Кол-во отправленных/принятых пакетов для указанного модуля ввода/вывода

Ненулевое значение байта *"Номер модуля"* позволяет читать статистику обмена с конкретным модулем в/в. Коды 0…4 при этом игнорируются.

Примеры вызовов (модуль подключен к шине ST-BUS1):

Ret_Val:=SYSTEM(21, 16#50007) – кол-во успешно принятых ответных пакетов от модуля 5 по линии 1

Ret_Val:=SYSTEM(21, 16#50008) – кол-во успешно принятых ответных пакетов от модуля 5 по линии 2

16#0015 (21): Статистика обмена по шинам ST-BUS (мастер-модули **M401E/M501E/M903E**/**M915E**)

Только для M401E/M501E/M903E/M915E (для других мастер-модулей описание приведено выше)

В зависимости от значения аргумента “Arg” позволяет получать кол-во отправленных/принятых пакетов или для определенной шины ST-BUS, или для определенного модуля.

Аргумент *"Arg"* имеет следующий формат:

Байт	3	2	1	0
Назначение	Номер линии BUSXXX	Номер модуля	Номер шины ST-BUS (1..N)	Код запроса

Код запроса – обязательный аргумент, может принимать следующие значения:

00h(0)	Сброс всех счетчиков
01h(1)	Количество отправленных пакетов по линии 1
02h(2)	Количество отправленных пакетов по линии 2
03h(3)	Количество успешно принятых ответных пакетов по линии 1

04h(4)	Количество успешно принятых ответных пакетов по линии 2
05h(5)	Количество ошибок по линии 1
06h(6)	Количество ошибок по линии 2

Кроме “Кода запроса” должен быть заполнен один из байтов 1..3

Байт 1. Кол-во отправленных/принятых пакетов для указанной шины ST-BUS

Чтение статистики для всей шины ST-BUS (в зависимости от кода запроса). Байт 1 при этом задает логический номер шины:

Тип мастер-модуля	**Значение байта 1**	**Описание**
M903E	1	Порт ST-BUS
M915E	1	Порт ST-BUS
	2	Юнит 1 (UCOM)
	3	Юнит 2 (UCOM)
	4	Юнит 3 (UCOM)

Примеры вызовов:

```
Ret_Val:=SYSTEM(21, 16#100); // сброс всех счетчиков для шины ST-BUS1
Ret_Val:=SYSTEM(21, 16#200); // сброс всех счетчиков для шины ST-BUS2
Ret_Val:=SYSTEM(21, 16#106); // кол-во ошибок по линии 2 шины ST-BUS1
Ret_Val:=SYSTEM(21, 16#206); // кол-во ошибок по линии 2 шины ST-BUS2
```

Байт 2. Кол-во отправленных/принятых пакетов для указанного модуля ввода/вывода

Ненулевое значение байта *"Номер модуля"* позволяет читать статистику обмена с конкретным модулем в/в. Поле *"Номер шины ST-BUS"* при этом игнорируется.

Примеры вызовов:

```
Ret_Val:=SYSTEM(21, 16#50003); // кол-во успешно принятых ответных пакетов от модуля 5 по линии 1
Ret_Val:=SYSTEM(21, 16#50004); // кол-во успешно принятых ответных пакетов от модуля 5 по линии 2
```

Байт 3. Кол-во отправленных/принятых пакетов для шины BUSXXX (настраиваются в web-конфигураторе)

Примеры вызовов:

```
Ret_Val:=SYSTEM(21, 16#1000000); // сброс всех счетчиков для шины BUS001
Ret_Val:=SYSTEM(21, 16#2000000); // сброс всех счетчиков для шины BUS002
Ret_Val:=SYSTEM(21, 16#1060000); // кол-во ошибок по линии 2 шины BUS001
Ret_Val:=SYSTEM(21, 16#2060000); // кол-во ошибок по линии 2 шины BUS002
```

16#0017 (23): Чтение флага “Логическая ошибка приложения”.

	Мастер-ПК	M841E M921E M902E M915E M903E	M911	M900/M800
Аргумент не используется и должен быть равен нулю. Функция возвращает значение флага (см. system с кодом 24).		•		

16#0018 (24): Установка/сброс флага “Логическая ошибка приложения”.

		Мастер-ПК	M841E M921E M902E M915E M903E	M911	M900/M800
Результат вызова оператора SYSTEM дублирует значение аргумента:					
00h	Сбросить флаг		•		
01h	Установить флаг		•		

Применяется при 100% резервировании контроллеров (подробнее см. пп.6.2, 7.2. “Резервирование контроллеров”). При установке единичного значения флага “основной” мастер-модуль выполняет попытку перехода в резерв аналогично аппаратным ошибкам (отключение модуля, и т.д.).

16#0019 (25): Снять запрет перехода контроллера в резерв по конкретным типам ошибок.

		Мастер-ПК	M841E M921E M902E M915E M903E	M911	M900/M800
Результат вызова оператора SYSTEM дублирует значение аргумента:					
01h(1)	Разрешить переход в резерв по "Ошибке внешних цепей юнитов"		•		
02h(2)	Разрешить переход в резерв, если за цикл обмена по одной из линий от модулей не было получено ни одного ответного пакета		•		
03h(3) – 09h(9)	Резерв				
0Ah(10)	Разрешить переход в резерв Backup – контроллеру		•		

Применяется при резервировании контроллеров (подробнее см. пп.6.2, 7.2. “Резервирование контроллеров”). Код 1 имеет значение при 100% резервировании, код 2 – при резервировании процессорной части (при 100% резервировании переход выполняется автоматически).
Примечание:
При использовании кода 2, во избежание частого переключения статусов при одновременном обрыве линий у обоих контроллеров, введена задержка в 3 секунды между получением контроллером статуса “основного” и переходом в резерв по данной ошибке.

16#001A (26): Запрет перехода контроллера в резерв по конкретным типам ошибок.

		Мастер-ПК	M841E M921E M902E M915E M903E	M911	M900/M800
Результат вызова оператора SYSTEM дублирует значение аргумента:					
01h	Запрет перехода в резерв по "Ошибке внешних цепей юнитов"		•		
02h(2)	Запретить переход в резерв, если за цикл обмена по одной из линий от модулей не было получено ни одного ответного пакета		•		
03h(3) – 09h(9)	Резерв				
0Ah(10)	Запрет перехода в резерв Backup – контроллеру		•		

Применяется при резервировании контроллеров (подробнее см. пп.6.2, 7.2. “Резервирование контроллеров”). Код 1 имеет значение при 100% резервировании, код 2 – при резервировании процессорной части (при 100% резервировании переход выполняется автоматически).

16#001B (27): Чтение состояния блокировок, установленных системным вызовом system(26,…).

		Мастер-ПК	M841E M921E M902E M915E M903E	M911	M900/M800
Результат вызова оператора SYSTEM зависит от значения аргумента:					
01h	Чтение состояния блокировки по "Ошибке внешних цепей юнитов"		•		
02h(2)	Чтение состояния блокировки на переход в резерв, если за цикл обмена по одной из линий от модулей не было получено ни одного ответного пакета		•		
03h(3) – 09h(9)	Резерв				
0Ah(10)	Чтение состояния блокировки перехода в резерв Backup – контроллеру		•		

Применяется при резервировании контроллеров (подробнее см. пп.6.2, 7.2. “Резервирование контроллеров”).

16#001C (28): Чтение уставки часового пояса.

	Мастер-ПК	M841E M921E M902E M915E M903E	M911	M900/M800
Аргумент не используется и должен быть равен нулю.		•		

Чтение уставки, задаваемой системным вызовом system(29,…).

16#001D (29): Установка часового пояса.

	Мастер-ПК	M841E M921E M902E M915E M903E	M911	M900/M800
Возвращаемое значение дублирует значение аргумента ***arg***.		•		

Данное значение прибавляется к метки времени, полученной с NTP-сервера, и вычитается при упаковке времени в Unix-time (см. *“Руководство по программированию”*, функция *TIME_TO_UNIX*), т.к. в обоих случаях применяется формат UTC+0.

16#001E (30): Чтение ошибок задач связи

		Мастер-ПК	M841E M921E M902E M915E	M911	M900/M800
Аргумент - номер задачи связи от 0 до 99. Возвращаемая величина – код ошибки:					
1	Таймаут на прием				
2	Таймаут на передачу				
3	Несовпадение контрольной суммы пакета				
4	Несовпадение контрольной суммы заголовка				
5	Несовпадение контрольной суммы фрейма				
6	Неверный формат пакета				

16#001F (31): Чтение состояния дискретных (импульсных) входов.

Для мастер-модулей **M501E/M903E/M915E:**

Формат аргумента:

Бит	32		…		16	15		…		0
Значение	номер юнита (1..4)					номер канала (1..4)				

Возвращаемое значение – 0 или 1, в зависимости от состояния входа.
Для мастер-модуля M903E следует задавать номер юнита=0, номер канала=1.

Для мастер-модулей **M1011E/M1011E2:**

Результат вызова оператора зависит от значения аргумента:

Аргумент (Dec)	Результат
1	состояние входа канала 1 (0-выкл, 1 – вкл.)
2	состояние входа канала 2 (0-выкл, 1 – вкл.)
8	кол-во импульсов по каналу 1
9	кол-во импульсов по каналу 2
16	сброс счетчика импульсов по каналу 1
17	сброс счетчика импульсов по каналу 2

16#0020 (32): Установка дискретных выходов (мастер-модули **M501E/M903E/M915E**).

Формат аргумента:

Бит	32		…		16	15		…		0
Значение	номер юнита (1..4)					номер канала (1..4)				

Для мастер-модуля M903E следует задавать номер юнита=0, номер канала=1.

16#0021 (33): Сброс дискретных выходов (мастер-модуль **M501E/M903E/M915E**).

Формат аргумента:

Бит	32		…		16	15		…		0
Значение	номер юнита (1..4)					номер канала (1..4)				

Для мастер-модуля M903E следует задавать номер юнита=0, номер канала=1.

16#0022 (34): Чтение диагностики по обмену с модулем

		Мастер-ПК	M841E M921E M902E M915E M903E	M911	M900/M800
Аргумент – номер модуля. Возвращаемая величина – статус предыдущего цикла обмена:					
0	Завершено без ошибок		•		
1	Завершено с ошибками		•		

Возвращаемое значение устанавливается в единицу, если в предыдущем цикле обмена с модулем возникли ошибки, т.е. хотя бы по одной переменной присутствует недостоверность. В случае асинхронного обмена по линии данный вызов имеет значение только когда завершен цикл обмена (см. вызов SYSTEM(40,3)), во время асинхронного цикла вызов будет возвращать единицу.

16#0023 (35): Чтение/запись блоков данных с модулей в/в

Действие системного вызова аналогично вызову функции operate с кодом 6. Аргумент *"arg"* имеет следующую структуру:

Байт	3	2	1	0			
Бит	31..24	23..16	15..8	7	…	1	0
Назначение		Номер модуля	Номер блока			Выполнить запись	Выполнить чтение

Системный вызов применяется на мастер-модуле для принудительного чтения/записи блоков данных с модулей M500 (имеющих в словаре атрибут “чтение/запись по запросу”) в глобальный словарь мастер-модуля.

Примеры вызовов:

Ret_Val:=SYSTEM(35, 16#010302) – Запись блока 3 на модуль 1;

Ret_Val:=SYSTEM(35, 16#050301) – Чтение блока 3 с модуля 5.

16#0024 (36): Чтение признака завершения операции для задачи межконтроллерного обмена по протоколу Modbus

		Мастер-ПК	M841E M921E M902E M915E M903E	M401E	M911	M900/M800
Аргумент – номер задачи связи. Возвращаемая величина – статус цикла обмена:						
0	Запрос выполняется			•		
1	Выполнение запроса завершено			•		

16#0025 (37): Чтение кода завершения операции для задачи межконтроллерного обмена по протоколу Modbus

		Мастер-ПК	M841E M921E M902E M915E M903E	M401E	M911	M900/M800
Аргумент – номер задачи связи. Возвращаемая величина – код завершения предыдущего цикла обмена:						
0	Нет ошибок			•		
2	Удаленное устройство не отвечает (таймаут истек)			•		
3	Системная ошибка при работе с портом / сервер Modbus-TCP недоступен			•		
4	Системная ошибка при обращении к задаче связи			•		
5	Выполняется запрос			•		
6	Ошибка принятых данных			•		
13	Некорректный ответ от slave-устройства (Адрес устройства/Код функции/Количество данных в ответном пакете отличается от запрашиваемого)			•		
201-211	Пришел ответ исключения от Slave устройства (подробнее см. описание протокола Modbus или документацию на Slave-устройство)			•		
201	Illegal function			•		
202	Illegal data address			•		
203	Illegal data value			•		
204	Server device failure			•		
205	Acknowledge			•		
206	Server device busy			•		
208	Memory parity error			•		
210	Gateway path unavailable			•		
211	Gateway target device failed to respond			•		

16#0026 (38): Чтение признака текущего абонента для задачи межконтроллерного обмена по протоколу Modbus

Задачи связи в режиме “Master” дают возможность задать резервного абонента. Системный вызов возвращает признак абонента, с которым выполняется обмен:

		Мастер-ПК	M841E M921E M902E M915E M903E	M401E	M911	M900/M800
Аргумент – номер задачи связи. Возвращаемая величина – признак текущего абонента:						
0	Обмен с основным абонентом			•		
1	Обмен с резервным абонентом			•		

16#0027 (39): Чтение признака доступности удаленных абонентов для задач межконтроллерного обмена

Для всех абонентов, указанных в задачах межконтроллерного обмена (тип TN_ETHERNET и TN_ETHERNET_RES), запускается отдельный поток, который периодически (не чаще, чем раз в секунду) выполняет запрос ping к ним. Системный вызов возвращает битовую маску доступности абонентов:

		Мастер-ПК	M841E M921E M902E M915E M903E	M401E	M911	M900/M800
Аргумент – номер задачи связи. Возвращаемая величина – битовая маска доступности абонентов:						
0	Абонент недоступен		•			
Бит 1	Доступен абонент 1		•			
…	…		•			
Бит N	Доступен абонент N		•			

16#0028 (40): Чтение признаков выполнения приложения.

		Мастер-ПК	M841E M921E M902E M915E M903E	M911	M900/M800
Результат вызова оператора SYSTEM зависит от значения аргумента:					
01h(1)	Режим выполнения (задание)		•	•	
02h(2)	Признак первого цикла программы		•	•	
03h(3)	Признак завершения цикла асинхронного обмена		•	•	

Для мастер-модулей ***M841E/M902E/M903E/M921E/M915E*** при использовании аргумента со значением 3, в старшей части задается номер линии. Например:

```
arg:=16#10003;  (*признак завершения цикла асинхронного обмена по линии 1*)
```

16#0029 (41): Чтение тревоги от дублирующего мастера.

	Мастер-ПК	M841E M921E M902E M915E M903E	M911	M900/M800
Аргумент не используется и должен быть равен нулю.		•		

16#002a (42): Посылка тревоги для дублирующего мастера.

	Мастер-ПК	M841E M921E M902E M915E M903E	M911	M900/M800
Аргумент – код тревоги. Возвращаемое значение – код тревоги от дублирующего мастера.		•		

16#002b (43): Изменение статуса мастера.

	Мастер-ПК	M841E M921E M902E M915E M903E	M911	M900/M800
Аргумент: 0 – автомат, 1 – основной, 2 – резервный		•		

16#002c (44): Чтение ошибок

Чтение ошибок необходимо выполнять ***в конце цикла***, т.к. системный вызов возвращает ошибки, накопленные в текущем цикле работы.

		Мастер-ПК	M501E M841E M921E M902E M903E M915E	M911	M900/M800
Результат вызова оператора SYSTEM зависит от значения аргумента:					
01h(1)	Ошибки зеркализации		•		
02h(2)	Резерв		•		
03h(3)	Ошибки драйвера		•		
04h(4)	Ошибки выполнения приложения		•		
05h(5)	Ошибки питания		•		
06h(6)	Глобальные флаги ошибок		•		
07h(7)	Глобальные флаги (диагностическая плата)		•		
08h(8)	Код перезапуска		•		
09h(9)	Режим работы исполнительной системы		•		
0ah(10)	Положение DIP переключателей (битовая маска)		•		
0bh(11)	Состояние дискретных выходов (битовая маска)		•		
0ch(12)	Состояние дискретных входов (битовая маска)		•		

1) Ошибки зеркализации [битовая маска]
- бит 0 -не задан режим резервирования
- бит 1 -несоответствие режимов резервирования
- бит 2 -ошибка контрольной суммы
- бит 3 -несоответствие размера данных
- бит 4 -несоответствие версии приложения
- бит 5 -несоответствие состава переменных
- бит 6 -несоответствие состава ф.блоков
- бит 7 -несоответствие состава файлов
- бит 8 -ошибка записи файла
- бит 9 -ошибка чтения файла
- бит 10 -установлена блокировка
- бит 11 -ошибка инициализации задачи связи
- бит 12 -ошибка при выполнении запроса (несоответствие фрейма, таймаут, и др.)
- бит 13 -неверно задан IP-адрес

3) Ошибки драйвера [целое]
- 00h(0) -нет ошибок
- 01h(1) -ошибка доступа к разделяемой памяти

02h(2) -ошибка доступа к семафору
03h(3) -ошибка доступа к памяти устройства
04h(4) -ошибка драйвера
05h(5) -PIC не отвечает
06h(6) -несуществующий указатель
07h(7) -некорректный режим работы PIC
08h(8) -несоответствие количества переданных и принятых пакетов
09h(9) -ошибочная команда
0Ah(10) -ошибочное состояние мастера
0Bh(11) -ошибка обработчика прерывания мастера

4) Ошибки выполнения приложения [битовая маска]
бит 0 -целочисленное деление на ноль
бит 1 -Переполнение при преобразовании FPU в целое
бит 2 -FPU:деление на ноль
бит 3 -FPU:неверный формат
бит 4 -FPU:переполнение в плюс бесконечность
бит 5 -FPU:переполнение в минус бесконечность

5) Ошибки питания [битовая маска]
бит 0 -напряжение питания ниже нормы
бит 1 -напряжение батареи ниже нормы
бит 2 -температура вне допустимого диапазона

6) Глобальные флаги ошибок [битовая маска]
бит 0 -наличие ошибок модулей
бит 1 -наличие ошибок каналов
бит 2 -наличие ошибок мастера
бит 3 -наличие ошибок зеркализации
бит 4 -наличие ошибок конфигурации
бит 5 -наличие ошибок последовательных линий
бит 6 -наличие ошибок линий ethernet
бит 7 -наличие ошибок ST-BUS
бит 8 -наличие ошибок сохранения/восстановления базы

7) Глобальные флаги (диагностическая плата) [битовая маска]
бит 0 -наличие динамических ошибок выполнения приложения (runtime)
бит 1 -ненулевой код запуска/обнаружен сброс модуля
бит 2 -приложение находится в режиме отладки
бит 3 -напряжение питания вышло из заданного диапазона (+аккумуляторы)
бит 4 -температура окружающей среды вышла из заданного диапазона
бит 5 -наличие апаратных ошибок работы модулей
бит 6 -наличие ошибок конфигурации модулей
бит 7 -наличие аппаратных ошибок работы юнитов
бит 8 -наличие ошибок чтения метрологических констант
бит 9 -наличие ошибок внешних цепей юнитов
бит 10 -наличие ошибок шины ST-BUS (обнаружено модулями)
бит 11 -наличие ошибок внешних коммуникаций
бит 12 -наличие ошибок внутренних коммуникаций
бит 13 -наличие ошибок горячего резервирования
бит 14 -DIP MODE 1 (switch) холодный старт
бит 15 -DIP MODE 2 (switch) технологический режим
бит 16 -DIP MODE 3 (switch) начальный статус "Основной"/"Резервный"
бит 17 -DIP MODE 4 (switch) резерв
бит 18 -DIP MODE 5 (switch) резерв
бит 19 -DIP MODE 6 (switch) резерв
бит 20 -RUN-STOP (switch) состояние переключателя RUN/STOP
бит 21 -текущий статус "Основной"/"Резервный"
бит 22 -напряжение литиевой батареи ниже 2.5В
бит 23 -наличие аппаратных ошибок мастера
бит 24 -наличие ошибок обмена с модулями
бит 25 -состояние дискретного выхода DO-1
бит 26 -состояние дискретного выхода DO-2

бит 27 -состояние дискретного выхода DO-3
бит 28 -состояние дискретного входа DI-1
бит 29 -состояние дискретного входа DI-2
бит 30 -состояние дискретного входа DI-3
бит 31 -состояние дискретного входа DI-4

8) Код перезапуска [целое]
00h(0) - нормальный режим работы (загрузка приложения / сброс приложения)
01h(1) - нормальное включение питания (перезагрузка через web-конфигуратор)
02h(2) - остановка переключателем STOP
03h(3) - остановка внешним запросом
05h(5) - остановка из-за ошибки приложения
10h(16) - перезапуск по питанию
11h(17) - перезапуск по Watchdog
20h(32) - остановка из-за внутренней ошибки

9) Режим работы исполнительной системы [битовая маска]
бит 0 -приложение в рабочем режиме
бит 1 -обрабатывается предыдущий запрос
бит 2 -приложение в режиме пошаговой отладки
бит 3 -приложение остановлено
бит 4 -приложение блокировано
бит 5 -приложение в "резервном" режиме

16#002d (45): Изменение состояния дискретных выходов (аргумент – маска, десятичное число).

16#002e (46): Чтение признака обмена по линии МКО (мастер-модули **M401E/M501E**)

Аргумент не используется, должен быть равен 0. Системный вызов возвращает признак наличия обмена по линиям зеркализации (МКО с атрибутом “вход/выход”)

		Мастер-ПК	M841E M921E M902E M915E M903E	M401E/M501E	M911	M900/M800
Аргумент – зарезервировано, должен быть равен 0. Возвращаемая величина – битовая маска наличия обмена:						
0	Обмен отсутствует			•		
Бит 0	Признак наличия обмена по линии 1			•		
Бит 1	Признак наличия обмена по линии 2			•		

16#002f (47): Чтение признака системной ошибки при обращении к портам Ethernet

		M903E
Аргумент – номер интерфейса (1 – LAN1, …, 4 – LAN4, 5 – SFP1, 6 – SFP2). Возвращаемая величина – признак ошибки:		
0	Нет ошибки	•
1	Ошибка	•

16#0030 (48): Чтение состояния Link для портов Ethernet

	Мастер-ПК	M501E M841E M921E M902E M903E	M401E	M911	M900/M800

Аргумент – номер интерфейса (1..3 для M841E/M902E, 1..6 для M903E). Возвращаемая величина – состояние Link:						
0	Кабель не подключен		•			
1	Кабель подключен		•			

Для M903E кодировка интерфейсов следующая: 1 – LAN1, …, 4 – LAN4, 5 – SFP1, 6 – SFP2

16#0031 (49): Чтение текущей скорости для портов Ethernet

	Мастер-ПК	M841E M921E M902E M903E	M401E	M911	M900/M800
Аргумент – номер интерфейса (1..3 для M841E/M902E, 1..6 для M903E). Возвращаемая величина – текущая скорость интерфейса (Мбит)		•			

Для M903E кодировка интерфейсов следующая: 1 – LAN1, …, 4 – LAN4, 5 – SFP1, 6 – SFP2

16#0034 (52): Управление возможностью загрузки приложения

		Мастер-ПК	M501E M841E M902E M903E M915E M921E M991S	M401E	M911	M900/M800
Аргумент – 0-разблокировать, 1-блокировать. Возвращаемая величина:			•			
-1	Недоступно в текущем режиме работы		•			
0	Загрузка разблокирована		•			
1	Загрузка заблокирована		•			

Позволяет в динамике управлять доступом к загрузке приложения. Если аргумент равен 1, то загрузка блокируется. Чтобы блокировка сохранялась после перезагрузки, переменной на входе system(52,X) нужно присвоить атрибут “Хранить” (или включить хранение всей базы).
Если на вход передается константа (system(52,1)), то загрузка будет заблокирована навсегда.

Для принудительной разблокировки необходимо запустить мастер-модуль в технологическом режиме (в данном режиме system(52,X) игнорируется, загрузка разрешена):

- M501E/M841E/M902E/M903E: установить при запуске DIP6=ON;
- M915E/M991S: установить при запуске RUN=MODE=ON (IP-адрес LAN1 при этом будет 192.9.200.1).

16#0035 (53): Чтение версий ПО (поддерживается на мастер-модулях **M501E/M903E/M915E**)

В зависимости от значения аргумента “Arg” позволяет получить версии ПО для мастер-модуля и подключенных модулей (для которых в проект добавлена модульная структура).

Аргумент *"Arg"* имеет следующий формат (десятичный вид):

Старшая часть	1 десятичная цифра	4 десятичные цифры
Резерв, должен быть равен 0	Код запроса	Номер модуля

Код запроса:

0 – Версия ПО
1 – Версия операционной системы (только для мастер-модулей)
2 – Версия прошивки коммуникационного контроллера (только для мастер-модулей)
3 – Версия прошивки доп. коммуникационного адаптера (LAN3/4, SFP1/2) мастер-модуля M903E
4 – Версия платы (только для мастер-модулей)

Возвращаемое значение имеет следующий формат (десятичный вид):

Старшая часть кода версии (4 цифры)	Младшая часть кода версии (4 цифры)

Например, версия 1.21 будет иметь вид “10021”

Примеры вызовов:

```
Ret_Val := SYSTEM(53,     0)
```
– версия ПО мастер-модуля (версия исполнительной системы);

```
Ret_Val := SYSTEM(53,   200)
```
– версия ПО модуля с адресом 200;

```
Ret_Val := SYSTEM(53, 10000)
```
– версия ОС мастер-модуля;

```
Ret_Val := SYSTEM(53, 20000)
```
– версия прошивки комм. контроллера мастер-модуля;

```
Ret_Val := SYSTEM(53, 30000)
```
– версия прошивки доп. коммуникационного адаптера (LAN3/4, SFP1/2) мастер-модуля M903E;

```
Ret_Val := SYSTEM(53, 40000)
```
– версия платы мастер-модуля.

Пример разбиения на старшую и младшую части:

```
Ver_h := Ret_Val / 10000;
Ver_l := mod(Ret_Val, 10000);
```

16#0036 (54): Чтение признака ошибки в обмене по шине ST-BUS

		Мастер-ПК	M841E M902E M903E M915E M921E	M911	M900/M800
Аргумент – номер шины (соответствует номеру BUSXXX в web-конфигураторе). Возвращаемая величина – признак ошибки в обмене:					
0	Ошибок нет		•		
Бит 0	Признак ошибки по линии 1		•		
Бит 1	Признак ошибки по линии 2		•		

Возвращаемое значение принимает ненулевое значение, если в обмене хотя бы с одним из модулей, подключенным к указанной шине, есть ошибки.

Кроме того, для M501E/M903E данный вызов позволяет контролировать шину ST-BUS на резервном мастере:

В режиме резервирования процессорной части резервный мастер стал контролировать шину ST-BUS. Если теряется линия зеркализации, но обмен по ST-BUS продолжается, то резервный не переходит в основной режим до тех пор, пока не пропадет обмен и по ST-BUS.

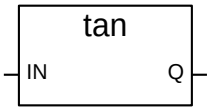
*Для M903E требуется версия ПО 2.50 и версия комм. контроллера не ниже 3.22.

16#0037 (55): Чтение признака ошибки в обмене с модулем ввода/вывода

		Мастер-ПК	M841E M902E M903E M915E M921E	M911	M900/M800
Аргумент – номер модуля ввода/вывода. Возвращаемая величина – признак ошибки в обмене:					
0	Ошибок нет		•		
Бит 0	Признак ошибки по линии 1		•		
Бит 1	Признак ошибки по линии 2		•		

Возвращаемое значение принимает ненулевое значение, если в обмене с указанным модулем хотя бы по одной из линий есть ошибки.

7.4.101 TAN



Вход:

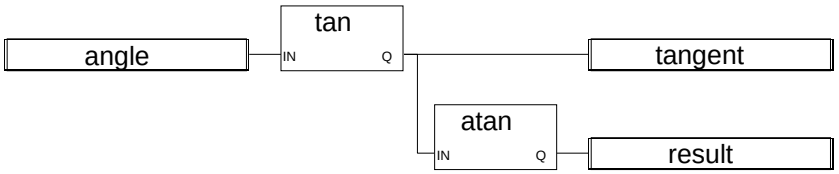
IN	REAL	Вещественная величина, не может равняться PI/2 по модулю PI

Выход:

Q	REAL	Тангенс входа

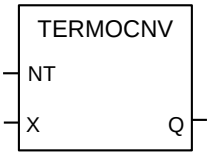
Описание:
Вычисляет тангенс вещественной величины.

(*FBD пример блока “TAN” и “ATAN”*)




```
(* ST Эквивалент : *)
tangent := TAN (angle);
result := ATAN (tangent); (*результат равен углу *)
```

7.4.102 TERMOCNV



Входы:

NT	INTEGER	Номер таблицы преобразования температуры
X	REAL	Значение физической величины (mV/Om)

Выход:

Q	REAL	Значение температуры (Град С)

Назначение

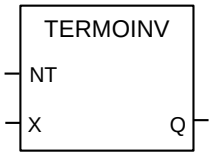
Функция используется для преобразования электрической величины в температуру для различных типов термопар (ТП) и термосопротивлений (ТС) TREI-5B.

Список таблиц преобразования приведен в таблице:

NT	Тип ТП/ТС	Эл. сигнал	Диапазон температуры	Примечание
1	TP S(PP)	19 mV	0 … 1600	
2	TP B(PR)	19 mV	0 … 1800	
3	TP J(FK)	75 mV	-200 … 1000	
4	TP T(MK)	75 mV	-250 … 400	
5	TP E(HK)	75 mV	-100 … 900	
6	TP K(HA)	75 mV	-200 … 1300	
7	TP N(NN)	75 mV	-200 … 1300	

8	TP L(HK)	75 mV	-200 … 800	
9	TP A-1(VR)	75 mV	0 … 2500	
10	TP A-2(VR)	75 mV	0 … 1780	
11	TP A-3(VR)	75 mV	0 …1780	
12	TR 50P	250 Om	-200 … 1100	платина W100=1,3910
13	TR 100P	500 Om	-200 … 1100	платина W100=1,3910
14	TR 50M	100 Om	-200 … 200	медь W100=1,4280
15	TR 100M	200 Om	-200 … 200	медь W100=1,4280
16	TR 21	250 Om	-200 … 600	градуировка 21
17	TR 23	100 Om	-50 … 180	градуировка 23
18	TR 50PT	65 Om	-50 … 80	платина W100=1,3910
19	TR 100PT	130 Om	-50 … 80	платина W100=1,3910
20	TR 100N	250 Om	-50 … 180	никель W100=1,6170
21	TR 50PB	125 Om	-200 … 400	платина W100=1,3910
22	TR 100PB	250 Om	-200 … 400	платина W100=1,3910
23	TR 50PA	250 Om	-200 … 850	платина W100=1,3850
24	TR 100PA	500 Om	-200 … 850	платина W100=1,3850
25	TR 50MA	100 Om	-50 … 200	медь W100=1,4260
26	TR 100MA	200 Om	-50 … 200	медь W100=1,4260
27	TR 50PTA	65 Om	-50 … 80	платина W100=1,3850
28	TR 100PTA	130 Om	-50 … 80	платина W100=1,3850
29	TR 50PBA	125 Om	-200 … 400	платина W100=1,3850
30	TR 100PBA	250 Om	-200 … 400	платина W100=1,3850
31	TR 50PC	250 Om	-200 … 850	платина a=0,00391
32	TR 100PC	500 Om	-200 … 850	платина a=0,00391
33	TR 50PTC	65 Om	-50 … 80	платина a=0,00391
34	TR 100PTC	130 Om	-50 … 80	платина a=0,00391
35	TR 50PBC	125 Om	-200 … 400	платина a=0,00391
36	TR 100PBC	250 Om	-200 … 400	платина a=0,00391
37	TR 50MC	100 Om	-180 … 200	медь a=0,00428
38	TR 100MC	200 Om	-180 … 200	медь a=0,00428

7.4.103 TERMOINV



Входы:

NT	INTEGER	Номер таблицы преобразования температуры
X	REAL	Значение температуры (Град С)

Выход:

Q	REAL	Значение физической величины (mV/Om)

Назначение

Функция используется для обратного преобразования температуры в электрическую величину для различных типов термопар (ТП) и термосопротивлений (ТС) TREI-5B.

Список таблиц преобразования приведен в таблице:

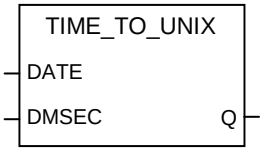
NT	Тип ТП/ТС	Эл. сигнал	Диапазон температуры	Примечание
1	TP S(PP)	19 mV	0 … 1600	

2	TP B(PR)	19 mV	0 … 1800	
3	TP J(FK)	75 mV	-200 … 1000	
4	TP T(MK)	75 mV	-250 … 400	
5	TP E(HK)	75 mV	-100 … 900	
6	TP K(HA)	75 mV	-200 … 1300	
7	TP N(NN)	75 mV	-200 … 1300	
8	TP L(HK)	75 mV	-200 … 800	
9	TP A-1(VR)	75 mV	0 … 2500	
10	TP A-2(VR)	75 mV	0 … 1780	
11	TP A-3(VR)	75 mV	0 …1780	
12	TR 50P	250 Om	-200 … 1100	платина W100=1,3910
13	TR 100P	500 Om	-200 … 1100	платина W100=1,3910
14	TR 50M	100 Om	-200 … 200	медь W100=1,4280
15	TR 100M	200 Om	-200 … 200	медь W100=1,4280
16	TR 21	250 Om	-200 … 600	градуировка 21
17	TR 23	100 Om	-50 … 180	градуировка 23
18	TR 50PT	65 Om	-50 … 80	платина W100=1,3910
19	TR 100PT	130 Om	-50 … 80	платина W100=1,3910
20	TR 100N	250 Om	-50 … 180	никель W100=1,6170
21	TR 50PB	125 Om	-200 … 400	платина W100=1,3910
22	TR 100PB	250 Om	-200 … 400	платина W100=1,3910
23	TR 50PA	250 Om	-200 … 850	платина W100=1,3850
24	TR 100PA	500 Om	-200 … 850	платина W100=1,3850
25	TR 50MA	100 Om	-50 … 200	медь W100=1,4260
26	TR 100MA	200 Om	-50 … 200	медь W100=1,4260
27	TR 50PTA	65 Om	-50 … 80	платина W100=1,3850
28	TR 100PTA	130 Om	-50 … 80	платина W100=1,3850
29	TR 50PBA	125 Om	-200 … 400	платина W100=1,3850
30	TR 100PBA	250 Om	-200 … 400	платина W100=1,3850
31	TR 50PC	250 Om	-200 … 850	платина a=0,00391
32	TR 100PC	500 Om	-200 … 850	платина a=0,00391
33	TR 50PTC	65 Om	-50 … 80	платина a=0,00391
34	TR 100PTC	130 Om	-50 … 80	платина a=0,00391
35	TR 50PBC	125 Om	-200 … 400	платина a=0,00391
36	TR 100PBC	250 Om	-200 … 400	платина a=0,00391
37	TR 50MC	100 Om	-180 … 200	медь a=0,00428
38	TR 100MC	200 Om	-180 … 200	медь a=0,00428

Примечание

Данная операция может быть использована для компенсации температуры холодного спая термопары. При этом температура холодного спая, измеренная термосопротивлением, должна быть преобразована в физическую величину по таблице, которая соответствует типу датчика термопары. Значение физической величины, измеренное термопарой, поправляется на величину компенсации.

7.4.104 TIME_TO_UNIX



Вход:

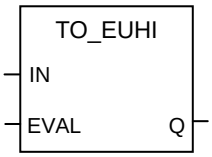
DATE	Integer	Дата в формате Unimod
DMSEC	Integer	Время в формате Unimod

Выход:

Q	Integer	Кол-во секунд с 1970 года

Описание:
Значения DATE и DMSEC могут быть получены, например, из функции GETTIME.

7.4.105 TO_EUHI



Входы:

IN	REAL	32 разрядное вещественное число
EVAL	INTEGER	Значение на выходе в случае ошибки

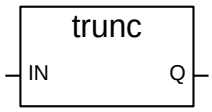
Выход:

Q	INTEGER	Число в формате EUHI

Назначение

Функция используется для преобразования 32 разрядного вещественного числа в формат EUHI (16 разрядное вещественное).

7.4.106 TRUNC



Вход:

IN	REAL	Любая знаковая вещественная величина

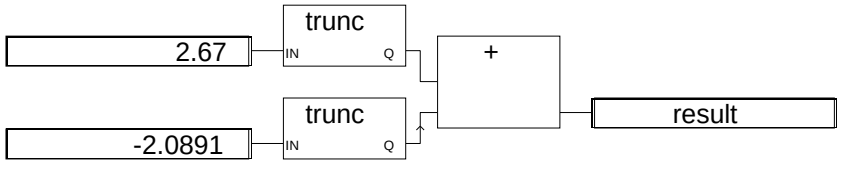
Выход:

Q	REAL	Целая часть вещественной величины

если IN > 0, то полученное целое, меньше либо равно входу
если IN < 0, то полученное целое, больше либо равно входу

Описание:
Отсекает дробную часть.

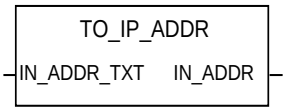
(*FBD пример блока “TRUNC”*)



(* ST Эквивалент: *)
result := TRUNC (+2.67) + TRUNC (-2.0891);

(* result := 2.0 + (-2.0) := 0.0; *)

7.4.107 TO_IP_ADDR



Вход:

IP_ADDR_TXT	Message	IP адрес в текстовом виде (XXX.XXX.XXX.XXX)

Выход:

IP_ADDR	Integer	IP адрес

Описание:
Формирование IP адреса из текстового представления. Используется совместно с функциональным блоком SOCK_OPEN.