6. Базовые операции языка FBD

Базовые операции языка FBD разделены на следующие группы:

-логические операции;
-операции сравнения;
-арифметические операции;
-операции управления программой и таймерами.

6.1 Логические операции

Логические операции в качестве операндов используют переменные типа **BOOLEAN, BYTE, SHORTINT, INTEGER, WORD, DWORD, LONGINT**.

Операции над переменными производятся поразрядно. На входах и выходах логических операций должны быть переменные одинаковых типов. После инициализации программы, переменные принимают значение:

FALSE - для переменных типа **BOOLEAN**

0 – для других типов переменных,

кроме тех глобальных переменных, значения которых при инициализации явно объявлено иначе.

Любой вход и выход логических блоков может быть проинвертирован.

NOT поразрядное инвертирование

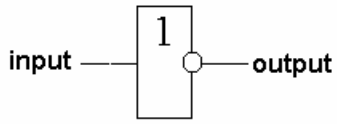


Таблица истинности

Input	**Output**
false	true
true	false

На входе блока могут быть переменные типов **BOOLEAN, BYTE, SHORTINT, INTEGER, WORD, DWORD, LONGINT.** На выходе блока поразрядная инверсия значения входной переменной.

Блок может использоваться без логической инверсии для непосредственной связи входа и выхода, или как элемент задержки в логических операциях.

AND поразрядное логическое "И"

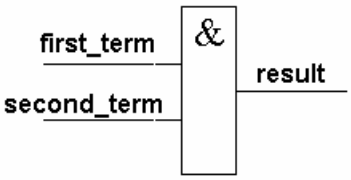


Таблица истинности

first_term	**second_term**	**result**
false	false	false
false	true	false
true	false	false
true	true	true

first_term первая переменная
second_term вторая переменная
result поразрядное логическое "И" первой и второй переменных

Количество входов блока может быть от 2 до 32. Вычисление результата производится сверху вниз. Сначала вычисляется логическое “И” первой и второй переменной, затем вычисляется логическое “И” результата и третьей переменной и т.д.

OR поразрядное логическое "ИЛИ"

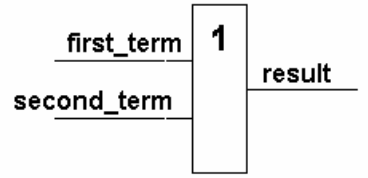


Таблица истинности

first_term	**second_term**	**result**
false	false	false
false	true	true
true	false	true
true	true	true

first_term первая переменная
second_term вторая переменная
result поразрядное логическое "ИЛИ" первой и второй переменных

Количество входов блока может быть от 2 до 32. Вычисление результата производится сверху вниз. Сначала вычисляется логическое “ИЛИ” первой и второй переменной, затем вычисляется логическое “ИЛИ” результата и третьей переменной и т.д.

XOR поразрядное логическое исключающее "ИЛИ"

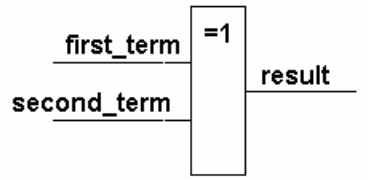


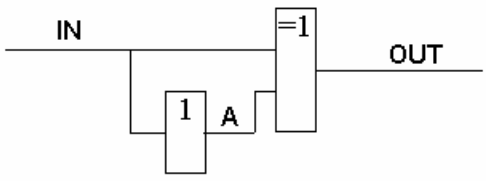
Таблица истинности

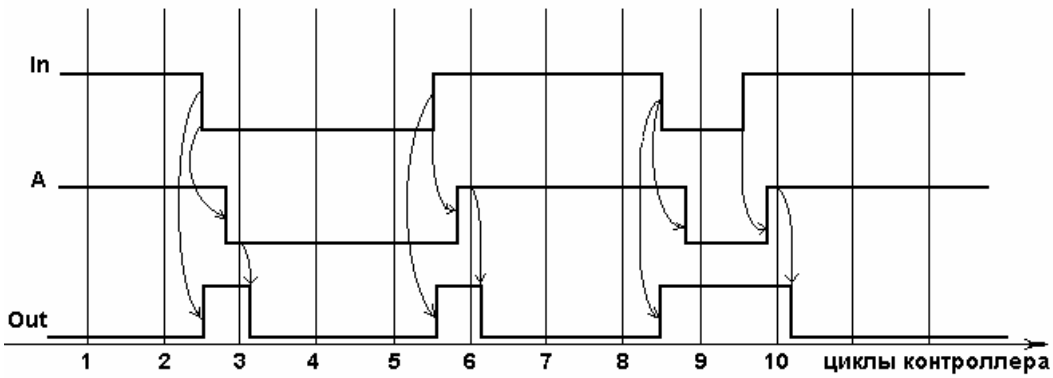
first_term	**second_term**	**result**
false	false	false
false	true	true
true	false	true
true	true	false

first_term первая переменная
second_term вторая переменная
result поразрядное логическое исключающее "ИЛИ" первой и второй переменных

Количество входов блока может быть от 2 до 32. Вычисление результата производится сверху вниз. Сначала вычисляется логическое исключающее “ИЛИ” первой и второй переменной, затем вычисляется логическое исключающее “ИЛИ” результата и третьей переменной и т.д.

Пример. Обнаружение перепада входного сигнала

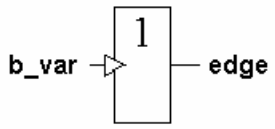




Временная диаграмма работы программы.

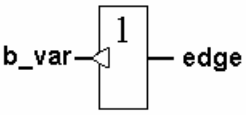
В исходном состоянии переменные IN и A на входе блока “XOR” имеют одинаковые значения. Во 2-м цикле переменная IN принимает значение FALSE. Переменные IN и A на входе блока “XOR” принимают противоположные значения, т.к. переменная **A** повторяет значение переменной IN после того, как выполнится операция “исключающее или”. (Блок “XOR” выполняется в программе первым, потому что он расположен выше блока “NOT”). Таким образом переменная OUT принимает значение TRUE всегда, когда переменная IN изменит свое состояние. Если переменная IN не изменяет свое состояние в следующем цикле, переменная OUT принимает значение FALSE.

PULSE Детектор фронта



b_var переменная типа **BOOLEAN**
edge переменная типа **BOOLEAN** принимает значение **TRUE** в течение одного цикла контроллера, если переменная **b_var** изменила состояние из **FALSE** в **TRUE, FALSE** во всех других случаях.

Тот же блок с инверсией на входе является детектором заднего фронта.

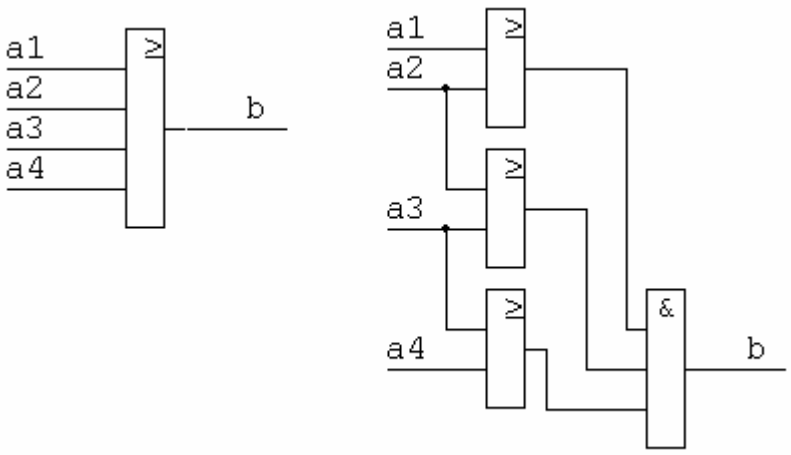


b_var переменная типа **BOOLEAN**
edge переменная типа **BOOLEAN** принимает значение **TRUE** в течение одного цикла контроллера**,** если переменная **b_var** изменила состояние из **TRUE** в **FALSE, FALSE** во всех других случаях.

6.2 Операции сравнения

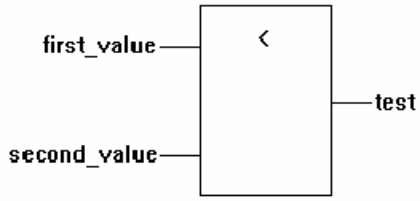
Все операции сравнения в качестве операндов могут использовать переменные любых типов, кроме переменных типа **BOOLEAN**. Входные переменные каждой отдельной операции должны иметь одинаковый тип. На выходах всех функциональных блоков сравнения по результату операции формируются переменные типа **BOOLEAN.** После инициализации программы эти переменные принимают значение **FALSE**, кроме тех глобальных переменных, значение которых при инициализации явно объявлены как **TRUE**.

Для всех операций сравнения количество операндов может достигать 32. При этом результат вычисляется попарно сверху вниз, т.е. сначала вычисляется результат сравнения первой и второй переменной, затем второй и третьей т.д. Результаты всех промежуточных вычислений объединяются логическим И.



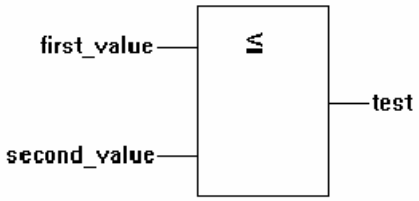
Четырехвходовый блок сравнения и эквивалентная программа на двухвходовых блоках.

Меньше чем



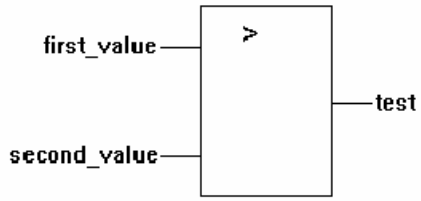
first_value первая переменная
second_value вторая переменная
test переменная типа **BOOLEAN** принимает значение **TRUE**, если **first_value** меньше **second_value**

Меньше или равно



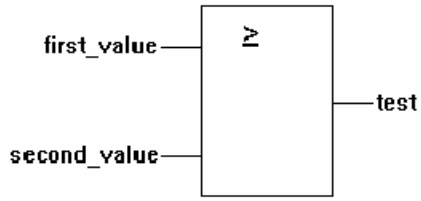
first_value первая переменная
second_value вторая переменная
test переменная типа **BOOLEAN** принимает значение **TRUE**, если **first_value** меньше или равно **second_value**

Больше чем



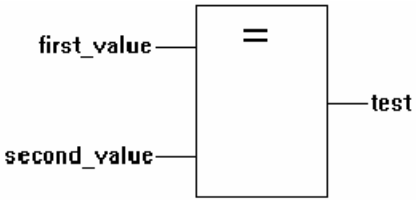
first_value первая переменная
second_value вторая переменная
test переменная типа **BOOLEAN** принимает значение **TRUE**, если **first_value** больше **second_value**

Больше или равно



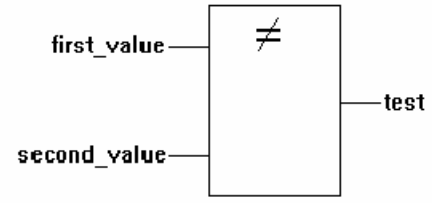
first_value первая переменная
second_value вторая переменная
test переменная типа **BOOLEAN** принимает значение **TRUE**, если **first_value** больше или равно **second_value**

Равно



first_value первая переменная
second_value вторая переменная
test переменная типа **BOOLEAN** принимает значение **TRUE**, если **first_value** равно **second_value**

Не равно



first_value первая переменная
second_value вторая переменная
test переменная типа **BOOLEAN** принимает значение **TRUE,** если **first_value** не равно **second_value**

6.3 Арифметические операции

Все арифметические операции в качестве операндов могут использовать переменные всех типов, за исключением типа BOOLEAN. Входные переменные каждой отдельной операции должны иметь одинаковый тип. После инициализации программы, переменные на выходах всех арифметических функциональных блоков принимают значение **0**, кроме тех глобальных переменных, значение которых при инициализации явно объявлены иначе.

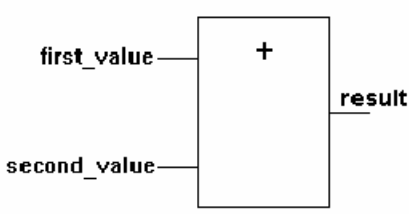
Предупреждение !

1. При арифметических операциях учитывайте ожидаемые диапазоны результатов вычислений, которые не должны выходить из допустимых значений переменных.
2. В случае возникновения ошибки переполнения результаты операции могут быть различными в зависимости от типа переменных, участвующих в операции и способа выполнения операций над переменными типа FLOAT.

Для всех арифметических операций, выполняемых способом программной эмуляции, в системе предусмотрены две глобальные переменные с именами ERR_SEG и ERR_OFS, в которых записывается адрес команды программы, где произошла ошибка. Если эти переменные приняли значение, отличное от нуля (заданное при инициализации), значит произошла ошибка арифметики, что должно быть учтено в алгоритме управления. Кроме этого на терминал выдается сообщение об ошибке. Это может мешать работе сети, если сеть настроена на порт COM1. Для разрешения ситуации средствами DOS направляйте сообщение на другие устройства, например NUL.

Для арифметических операций над переменными типа FLOAT, выполняемых аппаратным сопроцессором, переменные ERR_SEG и ERR_OFS не изменяют своего значения. Сопроцессор возвращает нечисловое значение, что может привести к непредсказуемым последствиям.

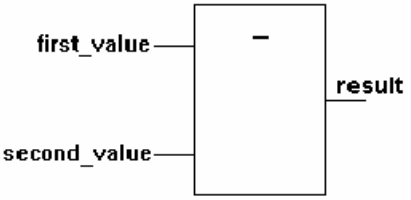
ADD сложение



first_value первая слагаемая переменная
second_value вторая слагаемая переменная
result знаковая сумма первой и второй переменных

Количество входов блока может быть от 2 до 32. Вычисление результата производится сверху вниз. Сначала вычисляется сумма первой и второй переменной, затем вычисляется сумма результата и третьей переменной и т.д.

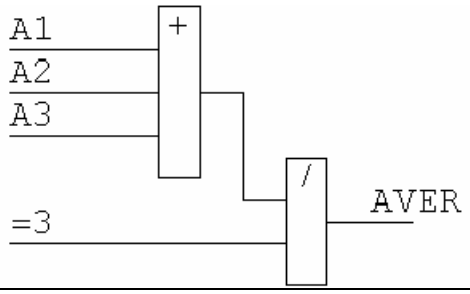
SUB вычитание



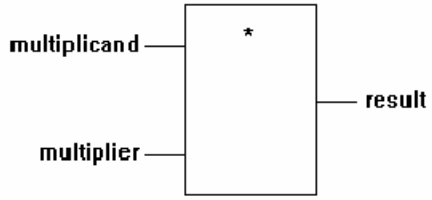
first_term первая уменьшаемая переменная
second_term вторая вычитаемая переменная
result знаковая разность первой и второй переменных

Количество входов блока может быть от 2 до 32. Вычисление результата производится сверху вниз. Сначала вычисляется разность первой и второй переменной, затем вычисляется разность результата и третьей переменной и т.д.

Пример. Вычисление среднего арифметического из 3-х чисел



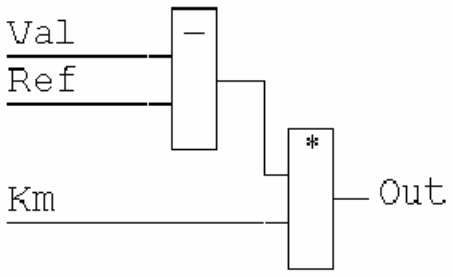
MUL умножение



multiplicand первая умножаемая переменная
multiplier вторая переменная - множитель
result знаковое произведение первой и второй переменных

Количество входов блока может быть от 2 до 32. Вычисление результата производится сверху вниз. Сначала вычисляется произведение первой и второй переменной, затем вычисляется произведение результата и третьей переменной и т.д.

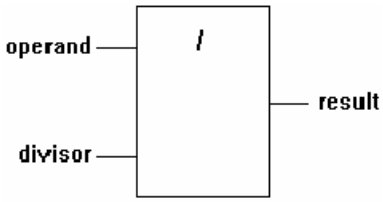
Пример. Пропорциональный регулятор



Val - регулируемый параметр
Ref - заданное значение параметра
Km - коэффициент пропорциональности
Out - сигнал регулирования

Разность (величина рассогласования) между истинным значением регулируемого параметра **Val** и его заданным значением **Ref** усиливается в **Km** раз со знаком и используется в качестве регулирующего воздействия.

DIV деление



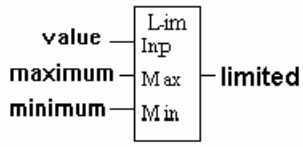
operand первая делимая переменная
divisor вторая переменная - делитель
result знаковое частное первой и второй переменных

Количество входов блока может быть от 2 до 32. Вычисление результата производится сверху вниз. Сначала вычисляется частное от деления первой на вторую переменную, затем вычисляется частное от деления результата на третью переменную и т.д.

Предупреждение !

Пользователь должен принимать специальные меры, чтобы переменная-делитель **divisor** не принимала значение нуля. В том случае, если делитель равен нулю, результат также будет равен нулю, а на терминал будет выведено сообщение об ошибке.

Limit Ограничитель сигнала

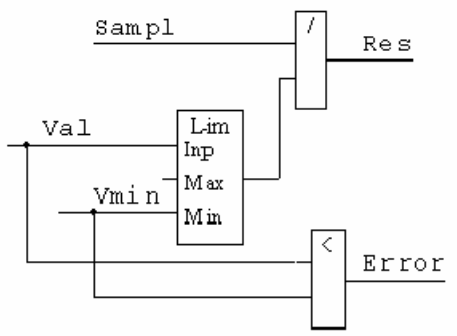


value текущее значение
maximum минимальное заданное значение
limited ограниченное значение :

= minimum если value < minimum
= maximum если value > maximum
= value если minimum < value < maximum

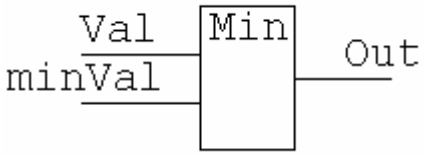
minimum минимальное заданное значение

Пример. Ограничение нижнего предела делителя



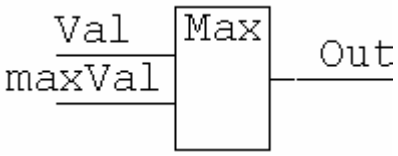
Переменная делителя **Val** ограничена минимальным значением **Vmin**. Переменная **Error** принимает значение **TRUE**, если **Val < Vmin**.

Min ограничение нижнего предела



val текущее значение
minVal минимальное заданное значение
Out ограниченное значение :
Out = minVal если val < minVal
Out = val если val>=minVal

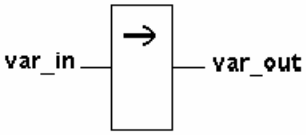
Max ограничение верхнего предела



val текущее значение
maxVal минимальное заданное значение
Out ограниченное значение :

Out = Val если val < =maxVal
Out = maxVal если val>Val

EQU присваивание



Присваивает переменной **var_out** значение переменной **var_in**.

Функция преобразования значения входной переменной **var_in** выбирается автоматически, исходя из типа переменной **var_out** согласно следующих правил:

1. Преобразование **var_in** в тип **BOOLEAN**

Тип **var_in**	Тип и значение **var_out**
BOOLEAN	**var_in** (копирует значение)
INTEGER	**FALSE** для **var_in = 0**, **TRUE** в других случаях
SHORTINT	**FALSE** для **var_in = 0**, **TRUE** в других случаях
BYTE	**FALSE** для **var_in = 0**, **TRUE** в других случаях
LONGINT	**FALSE** для **var_in = 0**, **TRUE** в других случаях
WORD	**FALSE** для **var_in = 0**, **TRUE** в других случаях
FLOAT	**FALSE** для **var_in = 0.0, TRUE** в других случаях
TIMER	**FALSE** для **var_in = 0S, TRUE** в других случаях

2. Преобразование **var_in** в тип **INTEGER, WORD**

Тип **var_in**	Тип и значение **var_out**
BOOLEAN	**0** для **var_in = FALSE, 1** для **var_in = TRUE**
SHORTINT	**var_in** (копирует значение)
INTEGER	**var_in** (копирует значение)
BYTE	**var_in** (копирует значение)
WORD	**var_in** (копирует значение)
DWORD	**var_in** (копирует значение)
LONGINT	**var_in** (копирует значение двух младших байтов)
FLOAT	целая часть от формата с плавающей точкой **INT**(**var_in**). Если результат не помещается, результат преобразования равен нулю и на терминал выводится сообщение об ошибке.
TIMER	время в сотых долях секунды

3. Преобразование **var_in** в тип **SHORTINT, BYTE**

Тип **var_in**	Тип и значение **var_out**
BOOLEAN	**0** для **var_in = FALSE, 1** для **var_in = TRUE**
SHORTINT	**var_in** (копирует значение
BYTE	**var_in** (копирует значение)
INTEGER	**var_in** (копирует значение младшего байта)
DWORD	**var_in** (копирует значение младшего байта)
LONGINT	**var_in** (копирует значение младшего байта)
FLOAT	целая часть от формата с плавающей точкой **INT**(**var_in**). Если результат не помещается, то результат преобразования приравнивается нулю и на терминал выводится сообщение об ошибке.
TIMER	время в сотых долях секунды

4. Преобразование **var_in** в тип **DWORD, LONGINT**

Тип **var_in**	Тип и значение **var_out**
BOOLEAN	**0** для **var_in = FALSE, 1** для **var_in = TRUE**
SHORTINT	**var_in** (копирует значение)
BYTE	**var_in** (копирует значение)
INTEGER	**var_in** (копирует значение)
WORD	**var_in** (копирует значение)

DWORD **var_in** (копирует значение)
LONGINT **var_in** (копирует значение)
FLOAT целая часть от формата с плавающей точкой **INT**(**var_in**)
TIMER время в сотых долях секунды

5. Преобразование **var_in** в тип **FLOAT**

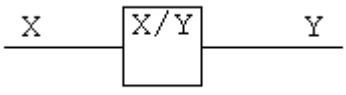
Тип **var_in**	Тип и значение **var_out**
BOOLEAN	**0.0** для **var_in = FALSE, 1.0** для **var_in = TRUE**
INTEGER	**var_in** (копирует значение)
SHORTINT	**var_in** (копирует значение)
BYTE	**var_in** (копирует значение)
WORD	**var_in** (копирует значение)
DWORD	**var_in** (копирует значение)
LONGINT	**var_in** (копирует значение)
FLOAT	**var_in** (копирует значение)
TIMER	время в сотых долях секунды

6. Преобразование **var_in** в тип **TIMER**

Тип **var_in**	Тип и значение **var_out**
BOOLEAN	**0S** для **var_in = FALSE, 0S01** для **var_in = TRUE**
INTEGER	время в сотых долях секунды
SHORTINT	время в сотых долях секунды
BYTE	время в сотых долях секунды
WORD	время в сотых долях секунды
DWORD	время в сотых долях секунды
LONGINT	время в сотых долях секунды
FLOAT	время в сотых долях секунды
TIMER	**var_in** (копирует значение)

Примечание. Если для переменной **var_in** была выполнена команда **TSTART**, то переменная **var_out** будет равна отрезку времени с момента подачи команды. Команда **TSTART** для переменной **var_out** не копируется.
Следите за тем, чтобы переменная **var_in** не принимала отрицательных значений, иначе результат преобразования значения переменной **var_in** в переменную типа **TIMER** будет непредсказуем.

Интерполяция

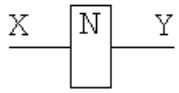


X,Y переменные типа FLOAT
Преобразует переменную X, в переменную Y в соответствии законом преобразования Y=f(X). Преобразование осуществляется в соответствии с таблицей, в которой задаются множество точек (x) и соответствующее им множество точек f(x). Точки могут быть заданы вручную или загружены из файла.

Сглаживание между точками производится с помощью интерполяции. Интерполяция может осуществляться двумя методами:

1. Метод построения полинома степени N по формуле f(X)=a0+a1x^1+a2x^2+a3x^3...aNx^N Степень полинома в зависимости от требуемой точности преобразования может выбираться от 1 до 20. Степень полинома не может быть больше количества записей в таблице минус один.
2. Метод кусочно-линейной интерполяции.

Поразрядное отрицание



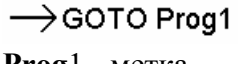
X,У - переменные типа **SHORTINT, WORD, DWORD, INTEGER, BYTE, LONGINT**

Производит поразрядное отрицание X и помещает результат в У.

6.4 Операции управления программой и таймерами

Операции управления программой в качестве входных операндов могут использовать переменные типа **BOOLEAN**.

GOTO переход к метке



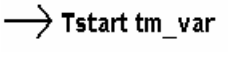
Prog1 - метка

Оператор **GOTO** может быть соединен с переменной типа **BOOLEAN**. Если данная переменная примет логическое значение **TRUE**, будет осуществлен переход к указанной метке. Если оператор **GOTO** не соединен с переменной типа **BOOLEAN**, осуществляется безусловный переход к метке.

RETURN выход из программы

Завершает текущую программу и передает управление следующей за ней программе. Для условного завершения программы оператор **RETURN** может быть соединен с переменной типа **BOOLEAN**. Если данная переменная примет логическое значение **TRUE**, программа завершится. В случае, если оператор **RETURN** встретился внутри библиотечного блока, то управление перейдет к следующему блоку, но не к следующей программе.

TSTART пуск таймера



tm_var переменная таймерного типа

Осуществляется инициализация (сброс) и начинается инкрементирование указанной таймерной переменной с периодом 0.01 сек. до момента выполнения команды **TSTOP**.
Оператор **TSTART** может быть соединен с переменной типа **BOOLEAN**. Если данная переменная примет логическое значение **TRUE**, оператор будет выполнен.

TSTOP останов таймера

→ TSTOP tm_var

tm_var переменная таймерного типа

Останавливает инкрементирование указанной таймерной переменной, состояние переменной сохраняется. Оператор **TSTOP** может быть соединен с переменной типа **BOOLEAN**. Если данная переменная примет логическое значение **TRUE**, оператор будет выполнен.

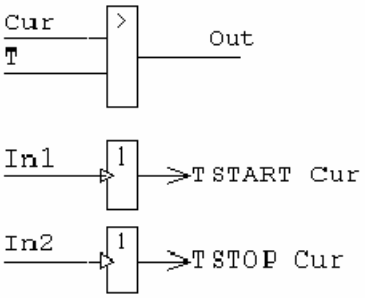
TCONTINUE Продолжить счет таймера

→ TCONTINUE tm_var

tm_var переменная таймерного типа

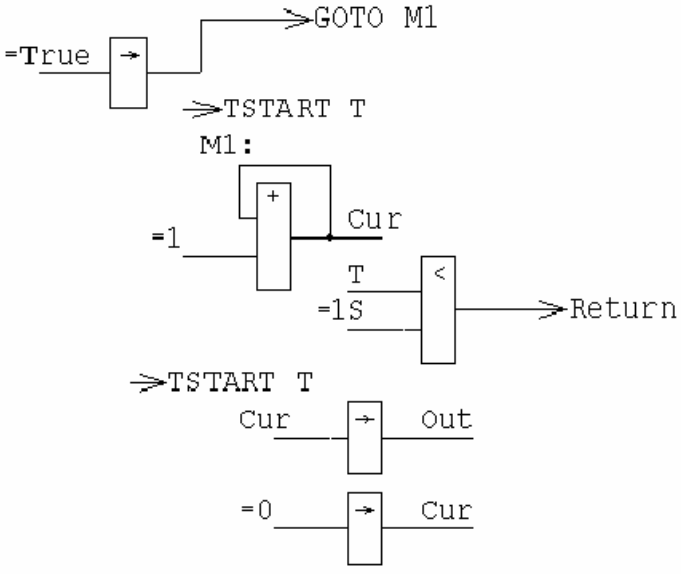
Продолжает инкрементирование указанной таймерной переменной, счет которой был остановлен оператором **TSTOP**. Оператор **TCONTINUE** может быть соединен с переменной типа **BOOLEAN**. Если данная переменная примет логическое значение **TRUE**, оператор будет выполнен.

Пример. Обнаружение заданного интервала времени T



Таймер **Cur** запускается, когда переменная **In1** принимает значение **TRUE,** и останавливается, когда переменная **In2** принимает значение **TRUE.** Переменная **Out** принимает значение **TRUE** в том случае, когда интервал времени, измеренный таймером **Cur** будет больше интервала, заданного переменной **T**.

Пример. Расчет частоты цикла контроллера



Оператор **GOTO** расположен выше оператора присваивания поэтому выполняется первым. Во время первого цикла на входе оператора **GOTO** установлено состояние **FALSE**, и переход к метке **M1** игнорируется. Управление передается оператору присваивания, который устанавливает на выходе логическое значение **TRUE.** Все последующие входы в программу будут начинаться с метки **M1**. Инициализируется таймерная переменная **T**. Переменная **Cur** увеличивает свое значение на **1** в каждом цикле контроллера. По истечении 1 сек. таймерная переменная **T** инициализируется вновь, значение переменной **Cu**r переписывается в переменную **Out**, переменной **Cur** присваивается значение 0 и счет начинается заново. Переменная **Out** содержит значение частоты циклов контроллера в герцах.

GSTART Старт программы

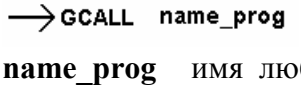
name_prog имя любой программы проекта.
Включает любую программу в цикл контроллера. Начиная с этого момента программа исполняется в каждом цикле контроллера. Оператор **GSTART** может быть соединен с переменной типа **BOOLEAN**. Если данная переменная примет логическое значение **TRUE**, оператор будет выполнен.

GSTOP Останов программы

name_prog имя любой программы проекта.

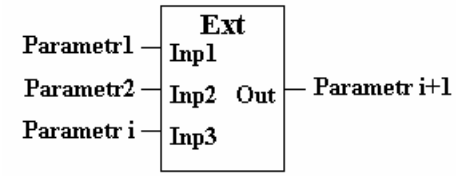
Исключает любую программу из цикла контроллера. Оператор **GSTOP** может быть соединен с переменной типа **BOOLEAN**. Если данная переменная примет логическое значение **TRUE**, оператор будет выполнен.

GCALL выполнить программу



name_prog имя любой программы проекта, которая должна быть выполнена. Оператор **GCALL** может быть соединен с переменной типа **BOOLEAN**. Если данная переменная примет логическое значение **TRUE**, оператор будет выполнен.

EXT выполнить внешнюю процедуру с передачей параметров



EXT - внешняя процедура в виде объектного модуля
Parametr - переменная любого типа, являющаяся параметром для операции EXT

Примечание

1. Количество входов и выходов фунции **EXT**, их название и функциональное назначение (Inp, I/O) определяется пользователем в диалоговом режиме (см. кн.1, гл.9 настоящего Руководства).

2. На выходы операции **Out** в качестве параметра не могут быть назначены константы.