ЦИФРОВЫЕ РЕШЕНИЯ

Библиотека Standard

ALTA

Руководство пользователя

06.2026
версия 1.12

Содержание

1 Цель документа

Библиотека Standard — это базовый набор функциональных блоков и функций, соответствующих стандарту МЭК 61131-3, обеспечивающих реализацию универсальных операций и типовых задач при программирования ПЛК.

Данный документ содержит описание алгоритмов работы блоков, входные/выходные параметры и типы данных каждого блока, и предназначен для ознакомления с составом библиотеки.

Для полного понимания принципов работы в среде необходимо ознакомиться с руководством пользователя ALTA IDE.

2 Установка библиотеки

Библиотека Standard установлена в ALTA IDE по умолчанию. Удаление данной библиотеки из проекта невозможно.

3 Описание библиотеки Standard

Библиотека Standard содержит:

- таймеры;
- счетчики;
- логические функциональные блоки (триггеры);
- детекторы импульсов;
- операторы сдвига;
- строковые функции;
- математические функции;
- операторы выборки;
- валидаторы;
- операции с временными типами данных;
- преобразователи.

3.1 Таймеры

3.1.1 TP

Функциональный блок TP (Pulse Timer) — повторитель импульсов.

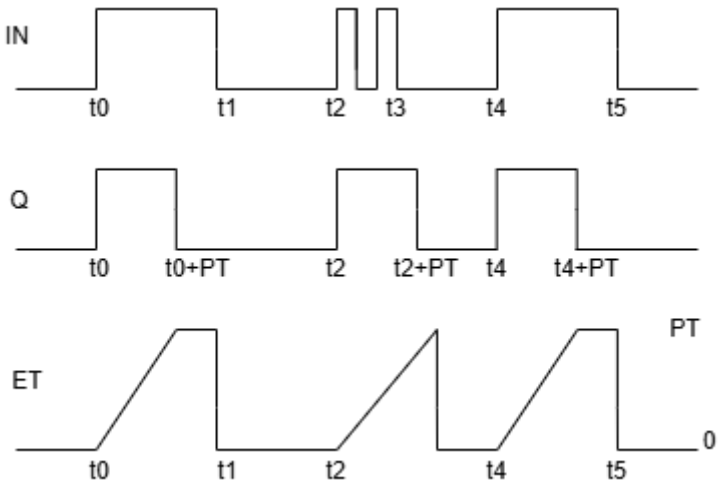

Рисунок 3.1

Имя переменной	Тип данных	Описание
Входные переменные		
IN	BOOL	Вход импульса
PT	TIME	Заданная длительность импульса
Выходные переменные		
Q	BOOL	Выход импульса
ET	TIME	Внутреннее время

Используется для генерирования импульса с заданной продолжительностью. Если IN становится «TRUE», Q становится «TRUE», и начинается отсчет внутреннего времени (ET). Если внутреннее время достигает значения PT, Q становится «FALSE» (независимо от IN). Отсчет внутреннего времени останавливается/сбрасывается, если IN становится «FALSE». Если внутреннее время не достигло значения PT, импульс IN не влияет на внутреннее время. Если внутреннее время достигло значения PT, и IN равен «FALSE», отсчет внутреннего времени останавливается/сбрасывается, и Q становится «FALSE».

Пример:

```
VAR
    TPInst:   TP;   //экземпляр TP
    Btn:      BOOL; //кнопка-сигнал
    PulseOut: BOOL; //импульсный выход
END_VAR

TPInst (IN := Btn, PT := T#200ms);

PulseOut := TPInst.Q;
```

3.1.2 TON

Функциональный блок TON (On Delay Timer) — таймер с задержкой включения.

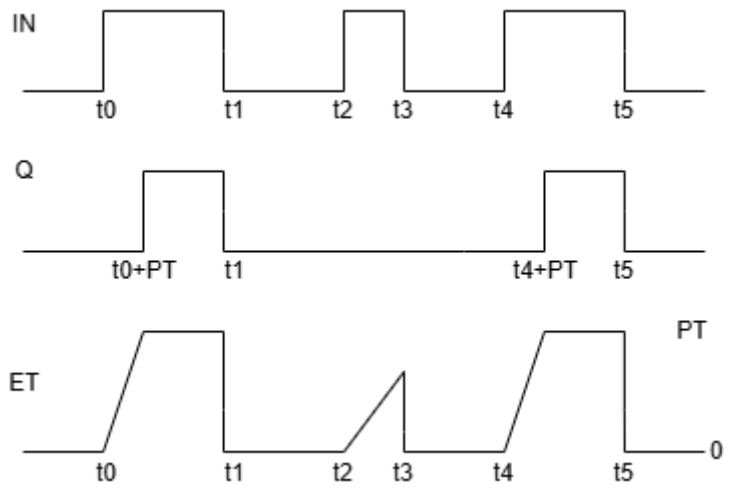

Рисунок 3.2

Имя переменной	Тип данных	Описание
Входные переменные		
IN	BOOL	Вход таймера
PT	TIME	Заданное время задержки включения
Выходные переменные		
Q	BOOL	Выход таймера
ET	TIME	Внутреннее время

Если IN становится «TRUE», запускается отсчет внутреннего времени (ET). Если внутреннее время достигает значения PT, Q становится «TRUE». Если IN становится «FALSE», Q становится «FALSE», а подсчет внутреннего времени останавливается/сбрасывается. Если IN становится «FALSE» до того, как внутреннее время достигло значения PT, подсчет внутреннего времени останавливается/сбрасывается, а выход Q не устанавливается в «FALSE».

Пример:

```
VAR
    fbTON:   TON;  //экземпляр таймера TON
    Start: BOOL; //входной сигнал
    Motor: BOOL; //выход
END_VAR

//вызов таймера каждый цикл
fbTON (IN := Start, PT := T#3s);
//выход активируется после выдержки
Motor := fbTON.Q;
```

3.1.3 TOF

Функциональный блок TOF (off-delay timer) — таймер с задержкой отключения.

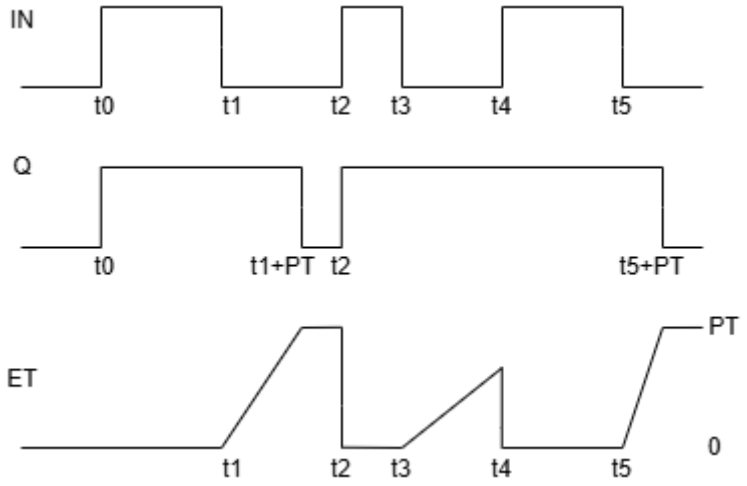

Рисунок 3.3

Имя переменной	Тип данных	Описание
Входные переменные		
IN	BOOL	Вход таймера
PT	TIME	Заданное время задержки выключения
Выходные переменные		
Q	BOOL	Выход таймера
ET	TIME	Внутреннее время

Если IN становится «TRUE», Q становится «TRUE».

Если IN становится «FALSE», запускается отсчет внутреннего времени (ET).

Если внутреннее время достигает значения PT, Q становится «FALSE».

Если IN становится «TRUE», Q становится «TRUE», а подсчет внутреннего времени останавливается/сбрасывается.

Если IN становится «TRUE» до того, как внутреннее время достигло значения PT, подсчет внутреннего времени останавливается/сбрасывается, а выход Q не устанавливается в «FALSE».

Пример:

```
VAR
    fbTOFF:   TOF;  //экземпляр таймера TOF
    RunCmd: BOOL; //команда "работать"
    Fan:    BOOL; //выход на вентилятор
END_VAR

fbTOFF (IN := RunCmd, PT := T#10s);

Fan := fbTOFF.Q;
```

3.2 Счетчики

3.2.1 CTU

Функциональный блок CTU (Up Counter) — инкрементный счётчик.

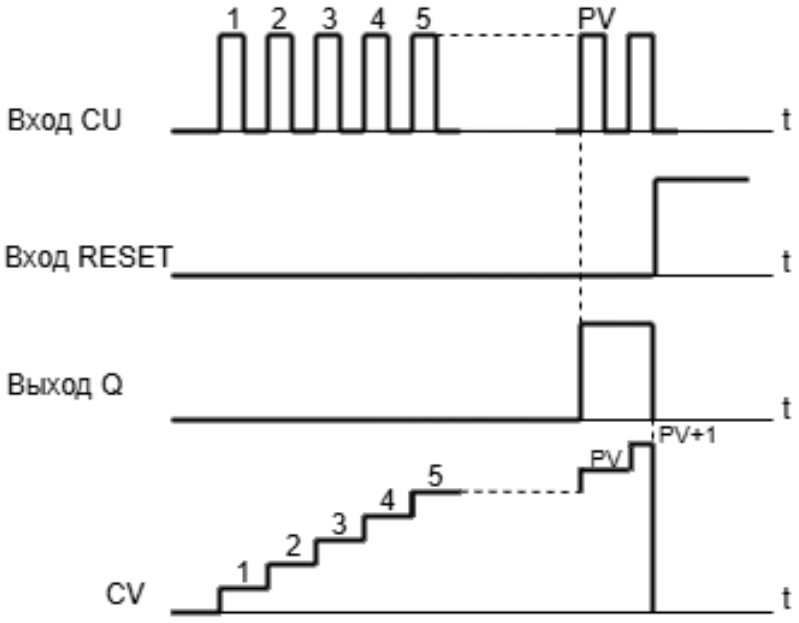

Рисунок 3.4

Имя переменной	Тип данных	Описание
Входные переменные		
CU	BOOL	Вход счета - увеличение на 1 при срабатывании
RESET	BOOL	Сброс счётчика в 0
PV	WORD	Предустановленное значение (уставка)
Выходные переменные		
Q	BOOL	Признак достижения уставки
CV	WORD	Текущее значение счетчика

По каждому фронту на входе CU (переход из FALSE в TRUE) выход CV увеличивается на 1. Выход Q устанавливается в TRUE, когда счетчик достигнет значения заданного PV. Счетчик CV сбрасывается в 0 по входу RESET = TRUE.

Пример:

```
VAR
    Cnt:      CTU;  //экземпляр счетчика CTU
    Pulse:    BOOL; //импульсы счета
    ResetBtn: BOOL; //сброс
    Done:     BOOL;
    Value:    WORD;
END_VAR

Cnt (CU := Pulse, RESET := ResetBtn, PV := 10); //считать до 10

Done := Cnt.Q;   //TRUE, когда CV >= 10
Value := Cnt.CV; //текущее значение счетчика
```

3.2.2 CTD

Функциональный блок CTD (Counter Down) — декрементный счётчик.

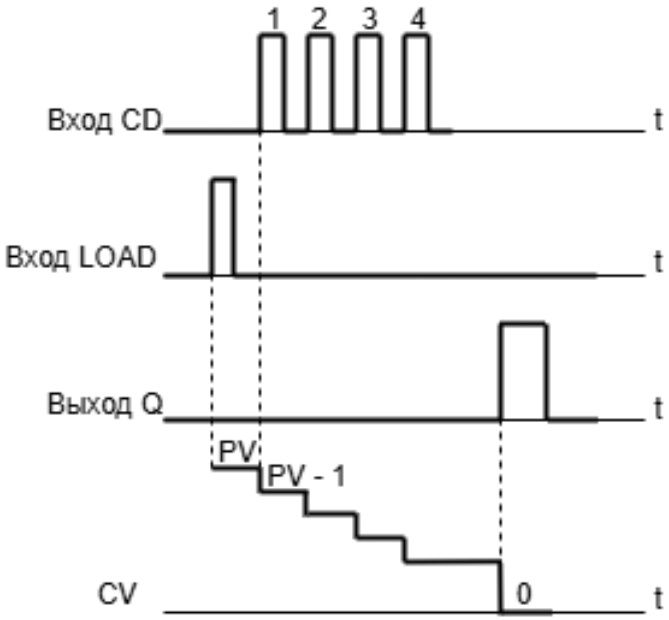

Рисунок 3.5

Имя переменной	Тип данных	Описание
Входные переменные		
CD	BOOL	Вход счета - уменьшение на 1 при срабатывании
LOAD	BOOL	Вход загрузки предустановленного значения
PV	WORD	Предустановленное значение (уставка)
Выходные переменные		
Q	BOOL	Признак достижения 0
CV	WORD	Текущее значение счетчика

По каждому фронту на входе CD (переход из FALSE в TRUE) выход CV уменьшается на 1. Когда счетчик достигнет 0, счет останавливается, выход Q переключается в TRUE. Счетчик CV загружается начальным значением, равным PV по входу LOAD = TRUE.

Пример:

```
VAR
    CntD:     CTD;  //экземпляр счетчика CTD
    DecPulse: BOOL; //импульсы декремента
    LoadBtn:  BOOL; //загрузить начальное значение
    Zero:     BOOL;
    Value:    WORD;
END_VAR

CntD (CD := DecPulse, LOAD := LoadBtn, PV := 10); //загрузка 10, затем считать
 вниз
Zero := CntD.Q;   //TRUE, когда CV = 0
Value := CntD.CV; //текущее значение счетчика
```

3.2.3 CTUD

Функциональный блок CTUD (Count Up/Down) — инкрементный/декрементный счетчик.

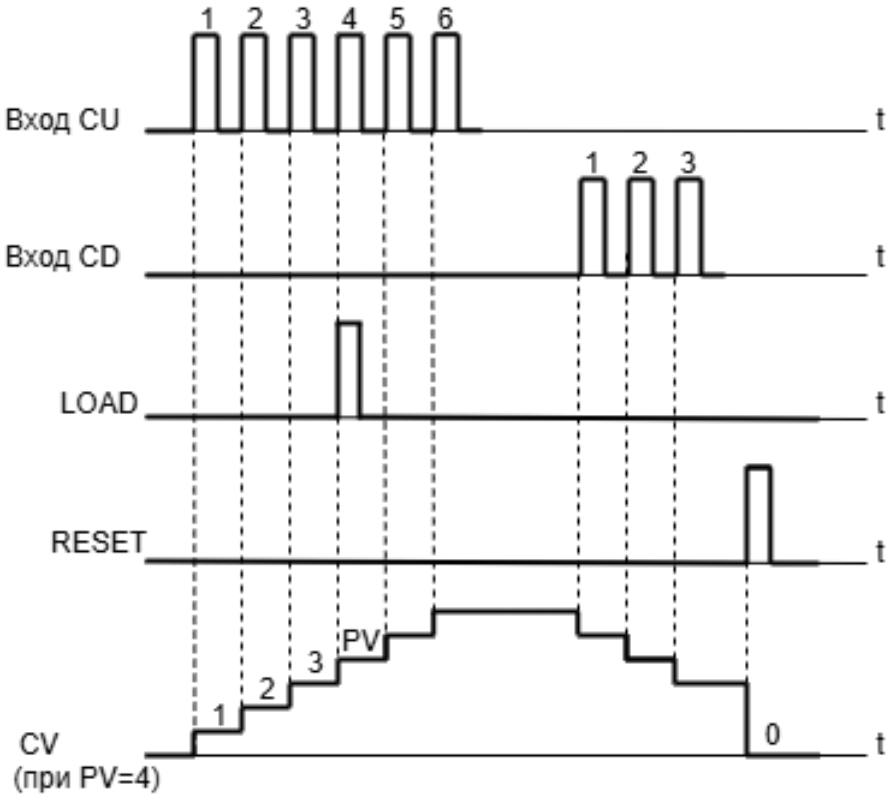

Рисунок 3.6

Имя переменной	**Тип данных**	**Описание**
Входные переменные		
CU	BOOL	Вход счета - увеличение на 1 при срабатывании
CD	BOOL	Вход счета - уменьшение на 1 при срабатывании
RESET	BOOL	Сброс счетчика в 0
LOAD	BOOL	Вход загрузки предустановленного значения
PV	WORD	Предустановленное значение (уставка)
Выходные переменные		
QU	BOOL	Признак достижения уставки
QD	BOOL	Признак достижения 0
CV	WORD	Текущее значение счетчика

По входу RESET счетчик CV сбрасывается в 0, по входу LOAD загружается значением PV.

По фронту на входе CU счетчик увеличивается на 1. По фронту на входе CD счетчик уменьшается на 1 (до 0).

QU устанавливается в TRUE, когда CV больше или равен PV.

QD устанавливается в TRUE, когда CV равен 0.

Пример:

```
VAR
    Cnt:      CTUD; //экземпляр счетчика CTUD
    UpPulse:  BOOL; //импульсы счета вверх
    DnPulse:  BOOL; //импульсы счета вниз
    ResetBtn: BOOL; //сброс
    LoadBtn:  BOOL; //загрузка PV и CV
    QU_reach: BOOL;
    QD_zero:  BOOL;
    Value:    WORD;
END_VAR

Cnt (CU := UpPulse, CD := DnPulse, RESET := ResetBtn, LOAD := LoadBtn, PV := 10);

QU_reach := Cnt.QU;  //TRUE, когда CV >= 10
QD_zero := Cnt.QD;   //TRUE, когда CV = 0
Value := Cnt.CV;     //текущее значение счетчика
```

3.3 Логические функциональные блоки (триггеры)

3.3.1 SR

Функциональный блок SR (Set-Reset) — переключатель с доминантой включения.

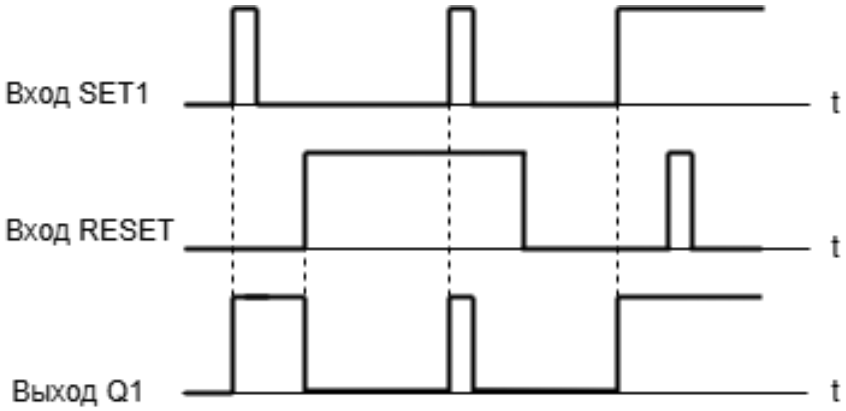

Рисунок 3.7

Имя переменной	Тип данных	Описание
Входные переменные		
SET1	BOOL	Вход установки
RESET	BOOL	Вход сброса
Выходные переменные		
Q1	BOOL	Запоминаемый выход (состояние триггера)

Функциональный блок с доминирующим входом SET1. Выход Q1 становится «TRUE», когда вход SET1 становится «TRUE». Это состояние сохраняется, даже если SET1 возвращается обратно в «FALSE». Выход Q1 возвращается в «FALSE», когда вход RESET становится «TRUE».

Если входы SET1 и RESET находятся в «TRUE» одновременно, доминирующий вход SET1 установит выход Q1 в «TRUE». Когда функциональный блок вызывается первый раз, начальное состояние Q1 это «FALSE».

Пример:

```
VAR
    Latch:   SR;   //экземпляр переключателя SR
    StartPB: BOOL; //установить
    StopPB:  BOOL; //сбросить
    MotorOn: BOOL;
END_VAR

Latch (SET1 := StartPB, RESET := StopPB);

MotorOn := Latch.Q1;
```

3.3.2 RS

Функциональный блок RS (Reset-Set) — переключатель с доминантой выключения.

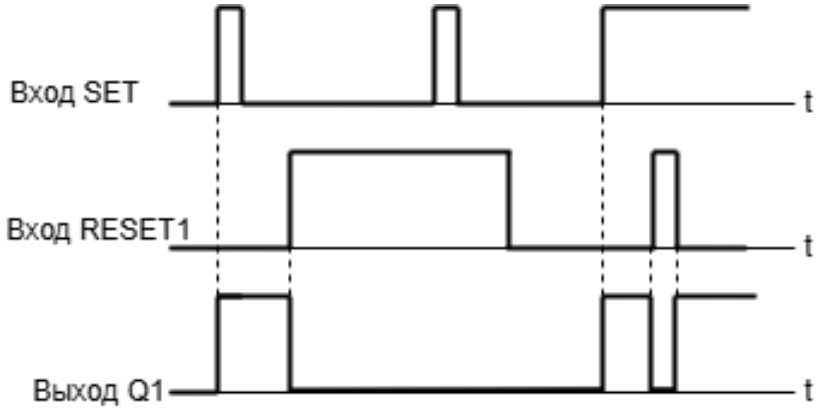

Рисунок 3.8

Имя переменной	Тип данных	Описание
Входные переменные		
SET	BOOL	Вход установки
RESET1	BOOL	Вход сброса
Выходные переменные		
Q1	BOOL	Запоминаемый выход (состояние триггера)

Функциональный блок с доминирующим входом RESET1. Выход Q1 становится «TRUE», когда вход SET становится «TRUE». Это состояние сохраняется, даже если SET возвращается обратно в «FALSE». Выход Q1 возвращается в «FALSE», когда вход RESET1 становится «TRUE».

Если входы SET и RESET1 находятся в «TRUE» одновременно, доминирующий вход RESET1 установит выход Q1 в «FALSE». Когда функциональный блок вызывается первый раз, начальное состояние Q1 это «FALSE».

Пример:

```
VAR
    Latch:   RS;   //экземпляр переключателя RS
    StartPB: BOOL; //установить
    StopPB:  BOOL; //сбросить, доминирует
    MotorOn: BOOL;
END_VAR

Latch (SET := StartPB, RESET1 := StopPB);

MotorOn := Latch.Q1;
```

3.4 Детекторы импульсов

3.4.1 R_TRIG

Функциональный блок R_TRIG (Rising Trigger) — детектор импульсов по переднему фронту.

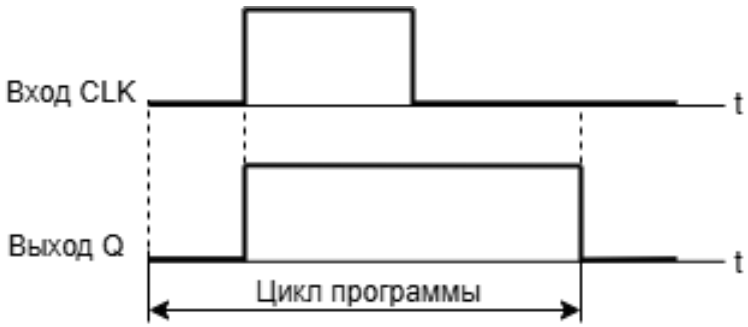

Рисунок 3.9

Имя переменной	Тип данных	Описание
Входные переменные		
CLK	BOOL	Вход сигнала, по переднему фронту которого выполняется детектирование
Выходные переменные		
Q	BOOL	Выход импульса

Индикатор нарастания фронта, который генерирует на выходе одиночный импульс при нарастании фронта сигнала. Выход Q становится «TRUE», если происходит переход из «FALSE» в «TRUE» на входе CLK . Выход остается в состоянии «TRUE» от одного выполнения блока до следующего (один цикл); затем выход возвращается в «FALSE».

Пример:

```
VAR
    rt:     R_TRIG; //экземпляр R_TRIG
    Button: BOOL;   //входной сигнал
    Pulse:  BOOL;   //импульс 1 цикл
END_VAR

rt (CLK := Button);

Pulse := rt.Q; //TRUE только при переходе Button FALSE -> TRUE

IF (rt.Q) THEN
    dwCount := dwCount + 1;
END_IF
```

3.4.2 F_TRIG

Функциональный блок F_TRIG (Falling Trigger) — детектор импульсов по заднему фронту.

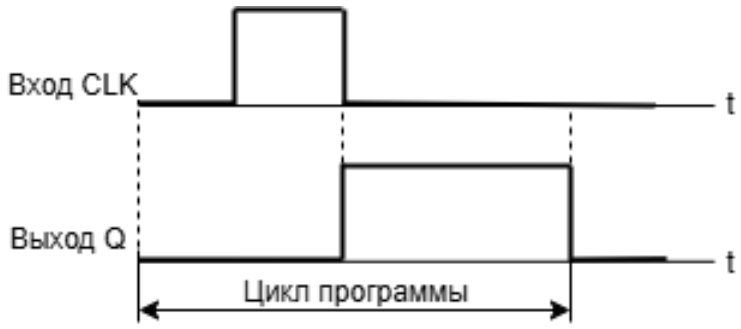

Рисунок 3.10

Имя переменной	Тип данных	Описание
Входные переменные		
CLK	BOOL	Вход сигнала, по заднему фронту которого выполняется детектирование
Выходные переменные		
Q	BOOL	Выход импульса

Индикатор спада фронта, который генерирует на выходе одиночный импульс при спаде фронта сигнала. Выход Q становится «TRUE», если происходит переход из «TRUE» в «FALSE» на входе CLK . Выход будет оставаться в состоянии «TRUE» от одного выполнения блока до следующего; затем выход возвращается в «FALSE».

Пример:

```
VAR
    ft:     F_TRIG; //экземпляр F_TRIG
    Sensor: BOOL;   //входной сигнал
    Pulse:  BOOL;   //импульс 1 цикл
END_VAR

Ft (CLK := Sensor);

Pulse := ft.Q; //TRUE только при переходе Sensor TRUE -> FALSE
IF (rt.Q) THEN
    dwCount := dwCount + 1;
END_IF
```

3.5 Операторы сдвига

3.5.1 SHL

Функция SHL (Shift Left) — побитовый сдвиг влево.

Возвращает в OUT побитный сдвиг операнда IN на N бит влево с заполнением битов справа нулями.
`OUT:=SHL(in,n)`

Входные переменные и результат типа BOOL, BYTE, WORD, DWORD или LWORD.

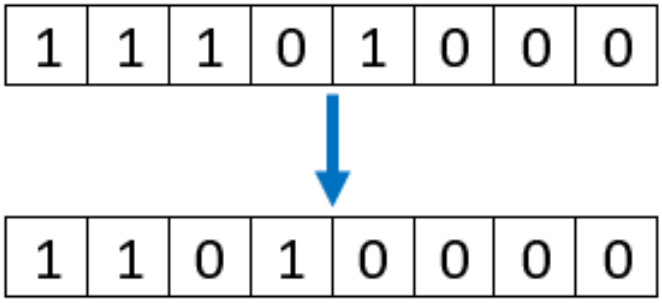

Рисунок 3.11 – Сдвиг влево, n=1

В следующем примере представлены различные результаты out_byte и out_word в зависимости от типа входной переменной (BYTE и WORD), хотя их числовые значения равны.

Пример:

```
VAR
    in_byte:  BYTE := 16#45;      // 2#0100_0101
    in_word:  WORD := 16#45;      // 2#0100_0101
    out_byte: BYTE;
    out_word: WORD;
    n:        BYTE := 2;
END_VAR

out_byte := SHL (in_byte, n); //Результат 2#0001_0100, 16#14
out_word := SHL (in_word, n); //Результат 2#1_0001_0100, 16#0114
```

3.5.2 SHR

Функция SHR (Shift Right) — побитовый сдвиг вправо.

Возвращает в OUT побитный сдвиг операнда IN на N бит вправо с заполнением битов слева нулями.
`OUT:=SHR(in,n)`

Входные переменные и результат типа BOOL, BYTE, WORD, DWORD или LWORD.

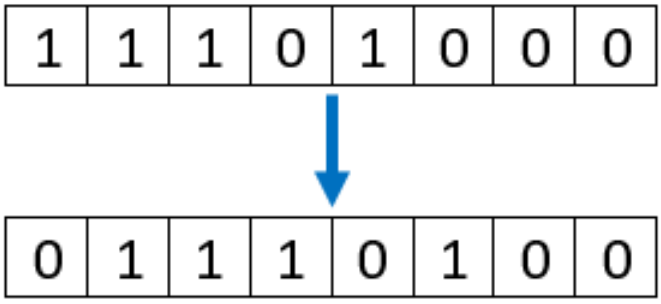

Рисунок 3.12 – Сдвиг вправо, n=1

Следующий пример подчеркивает зависимость результата от типа входной переменной.

Пример:

```
VAR
    in_byte:  BYTE := 16#45;     // 2#0100_0101
    in_word:  WORD := 16#45;     // 2#0100_0101
    out_byte: BYTE;
    out_word: WORD;
    n:        BYTE := 2;
END_VAR

out_byte := SHR (in_byte, n); //Результат 2#0001_0001, 16#11
out_word := SHR (in_word, n); //Результат 2#0000_0000_0001_0001, 16#0011
```

3.5.3 ROL

Функция ROL (Rotate Left) — циклический сдвиг влево.

Возвращает в OUT циклический сдвиг аргумента IN на N бит влево, младшие биты последовательно заменяются старшими.
`OUT:=ROL(in,n)`

Входные переменные и результат типа BOOL, BYTE, WORD, DWORD или LWORD.

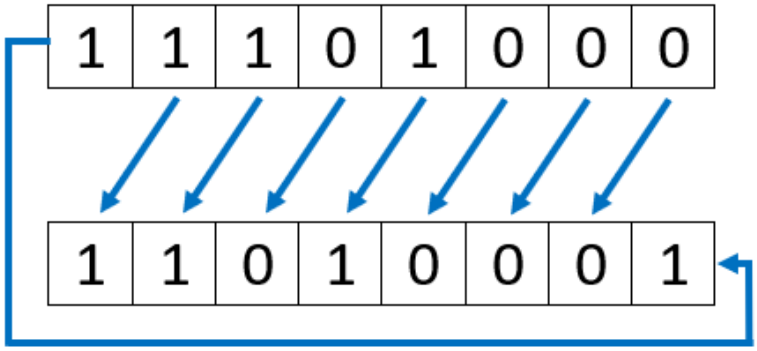

Рисунок 3.13 – Циклический сдвиг влево, n=1

В следующем примере представлены различные результаты OUT_byte и OUT_word в зависимости от типа входной переменной (BYTE и WORD), хотя числовые их значения равны.

Пример:

```
VAR
    in_byte:  BYTE := 16#45;
    in_word:  WORD := 16#45;
    out_byte: BYTE;
    out_word: WORD;
    n:        BYTE := 2;
END_VAR

out_byte := ROL (in_byte, n); //Результат 16#15
out_word := ROL (in_word, n); //Результат 16#0114
```

3.5.4 ROR

Функция ROR (Rotate Right) — циклический сдвиг вправо.

Возвращает в OUT циклический сдвиг аргумента IN на N бит вправо, старшие биты последовательно заменяются младшими.
`OUT:=ROR(in,n)`

Входные переменные и результат типа BOOL, BYTE, WORD, DWORD или LWORD.

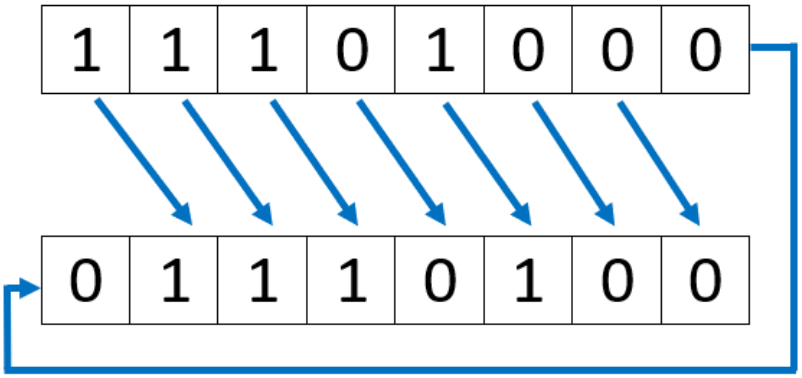

Рисунок 3.14 – Циклический сдвиг вправо, n=1

Следующий пример подчеркивает зависимость результата от типа входной переменной.

Пример:

```
VAR
    in_byte:  BYTE := 16#45;
    in_word:  WORD := 16#45;
    out_byte: BYTE;
    out_word: WORD;
    n:        BYTE := 2;
END_VAR

out_byte := ROR (in_byte, n); //Результат 16#51
out_word := ROR (in_word, n); //Результат 16#4011
```

3.5.5 TO_BIG_ENDIAN, TO_LITTLE_ENDIAN

Функции интерпретации порядка байт при обмене данными.

- TO_BIG_ENDIAN — старший байт передаётся первым.
- TO_LITTLE_ENDIAN — младший байт передаётся первым

Входные переменные и результат любого типа, для которого имеет смысл побайтовое представление.

Пример:

```
VAR
    w:   WORD := 16#1234;
    q:   WORD := 16#1234;
    wBE: WORD;
    qLE: WORD;
END_VAR

wBE := TO_BIG_ENDIAN (w);    //Возвращаемое значение 16#3412
qLE := TO_LITTLE_ENDIAN (q); //Возвращаемое значение 16#1234
```

3.5.6 FROM_BIG_ENDIAN, FROM_LITTLE_ENDIAN

Функция интерпретации порядка байт при обмене данными.

- FROM_BIG_ENDIAN — интерпретирует входное значение как записанное в big-endian, и преобразует его в порядок байт, принятых в системе.
- FROM_LITTLE_ENDIAN — интерпретирует входное значение как записанное в little-endian, и преобразует его в порядок байт, принятых в системе.

Входные переменные и результат любого типа, для которого имеет смысл побайтовое представление.

Пример:

```
VAR
    wFBE:  WORD := 16#3412; (*байты 34 12, как если бы пришло от big-endian
значение 16#1234*)
    wFLE:  WORD := 16#1234; (*байты 12 34, как если бы пришло от little-endian
значение 16#1234*)
    valBE: WORD;
    valLE: WORD;
END_VAR

valBE := FROM_BIG_ENDIAN (wFBE);    (*на little-endian CPU возвращаемое значение
16#1234*)
valLE := FROM_LITTLE_ENDIAN (wFLE); (*на little-endian CPU возвращаемое значение
16#1234*)
```

3.6 Строковые функции

3.6.1 LEN

Функция, возвращающая длину строки.

Аргумент STR типа STRING, возвращаемое значение типа DINT.

Пример:

```
VarSTRING1 := LEN ('ALTA'); //возвращаемое значение 4
```

3.6.2 LEFT

Функция, возвращающая левую часть строки заданной длины.

Входная строка STR типа STRING, размер SIZE типа INT, возвращаемое значение STRING.

Пример:

```
VarSTRING1 := LEFT ('ALTA',3); //возвращаемое значение 'ALT'
```

3.6.3 RIGHT

Функция, возвращающая правую часть строки заданной длины.

Входная строка STR типа STRING, размер SIZE типа INT, возвращаемое значение STRING.

Пример:

```
VarSTRING1 := RIGHT ('ALTA',3); //возвращаемое значение 'LTA'
```

3.6.4 MID

Функция, возвращающая часть строки заданной длины с заданной позиции.

Входная строка STR типа STRING, размер LEN и POS типа INT, возвращаемое значение STRING.

`MID (STR, LEN, POS)` означает: вырезать LEN символов из STR строки, начиная с POS.

Пример:

```
VarSTRING1 := MID ('ALTA',2,2); //возвращаемое значение 'LT'
```

3.6.5 CONCAT

Конкатенация (объединение) двух строк.

Обе входных строки STR1 и STR2 и результат типа STRING.

Пример:

```
VarSTRING1 := CONCAT ('ALTA','IDE'); //возвращаемое значение 'ALTAIDE'
```

3.6.6 FIND

Функция, предназначенная для поиска позиции заданного контекста в строке.

Входные переменные STR1 и STR2 типа STRING, возвращаемое значение INT.

`FIND(STR1, STR2)` означает: найти позицию в строке STR1, где впервые встречается подстрока STR2.

Нумерация позиций в строке начинается с 1. Если STR2 не найдена, STR1 возвращает 0.

Пример:

```
VarINT1 := FIND ('ALTA IDE','ID'); //возвращаемое значение 6
```

3.6.7 INSERT

Функция, предназначенная для вставки одной строки в заданную позицию другой строки.

Входные переменные STR1 и STR2 - типа STRING, POS - типа INT, возвращаемое значение - строка STRING.

`INSERT(STR1, STR2, POS)` означает: вставить STR2 в STR1 в позиции POS.

Пример:

```
VarSTRING1 := INSERT ('ALTA','IDE',2); //возвращаемое значение 'ALIDETA'
```

3.6.8 DELETE

Функция, предназначенная для удаления части строки с заданной позиции.

Входные переменные STR типа STRING, LEN и POS типа INT, возвращаемое значение строка STRING.

`DELETE(STR, LEN, POS)` означает: удалить LEN символов из STR, начиная с позиции POS.

Пример:

```
VarSTRING1 := DELETE ('ALTAIDE',2,3); //возвращаемое значение 'ALTDE'
```

3.6.9 REPLACE

Функция, предназначенная для замены части строки другой строкой с заданной позиции указанной длины.

Входные переменные STR1 и STR2 типа STRING, LEN и POS типа INT, возвращаемое значение строка STRING.

`REPLACE(STR1, STR2, LEN, POS)` означает: заменить LEN символов строки STR1 на STR2 начиная с позиции POS.

Пример:

```
VarSTRING1 := REPLACE ('ALTA','IDE',2,2); //возвращаемое значение 'AIDEA'
```

3.6.10 STRING_EQUAL, WSTRING_EQUAL

Функция, предназначенная для сравнения входных переменных и возврата значения TRUE при совпадении значений.

Входные переменные могут быть типа STRING и WSTRING, возвращаемое значение типа BOOL.

Пример:

```
OUT := STRING_EQUAL (‘ALTA’,‘IDE’);    //возвращаемое значение ‘FALSE’
OUT := STRING_EQUAL (‘ALTA’,‘ALTA’);   //возвращаемое значение ‘TRUE’
OUT := WSTRING_EQUAL ("ALTA","IDE");   //возвращаемое значение ‘FALSE’
OUT := WSTRING_EQUAL ("ALTA","ALTA");  //возвращаемое значение ‘TRUE’
```

3.6.11 STRING_GREATER, WSTRING_GREATER

Функция, предназначенная для сравнения входных переменных по лексикографическому признаку и возврата значения TRUE если значение первой переменной больше значения второй.

Входные переменные могут быть типа STRING и WSTRING, возвращаемое значение типа BOOL.

Пример:

```
VAR
    IN1: STRING := ‘ALTA’;
    IN2: STRING := ‘IDE’;
    IN3: WSTRING := "IDE";
    IN4: WSTRING := "ALTA";
    GS:  BOOL;
    GW:  BOOL;
END_VAR

GS := STRING_GREATER (IN1,IN2); (*возвращаемое значение FALSE, т.к. 'A'
лексикографически меньше 'I'*)
GW := WSTRING_GREATER (IN3,IN4); (*возвращаемое значение TRUE, т.к. 'I'
лексикографически больше 'A'*)
```

3.6.12 STRING_LESS, WSTRING_LESS

Функция, предназначенная для сравнения входных переменных по лексикографическому признаку и возврата значения TRUE если значение первой переменной меньше значения второй.

Входные переменные могут быть типа STRING и WSTRING, возвращаемое значение типа BOOL.

Пример:

```
VAR
    IN1: STRING := ‘ALTA’;
    IN2: STRING := ‘IDE’;
    IN3: WSTRING := "IDE";
    IN4: WSTRING := "ALTA";
    GS:  BOOL;
    GW:  BOOL;
END_VAR

GS := STRING_LESS (IN1,IN2); (*возвращаемое значение TRUE, т.к. 'A'
лексикографически меньше 'I'*)
GW := WSTRING_LESS (IN3,IN4); (*возвращаемое значение FALSE, т.к. 'I'
лексикографически больше 'A'*)
```

3.7 Математические функции

3.7.1 ABS

Функция, предназначенная для возврата модуля входного числа.
Входная переменная может быть любого числового типа, с соответствующим возвращаемым значением.
Пример:

```
q := ABS (-5); //возвращаемое значение 5
```

3.7.2 SQRT

Функция, предназначенная для возврата квадратного корня входного числа.
Входная переменная может быть типа REAL и LREAL, с соответствующим возвращаемым значением.
Пример:

```
q := SQRT (16.0); //возвращаемое значение 4.0
```

3.7.3 LN

Функция, предназначенная для возврата натурального логарифма входного числа.
Входная переменная может быть типа REAL и LREAL, с соответствующим возвращаемым значением.
Пример:

```
q := LN (36.6); //возвращаемое значение 3.6000482404073
```

3.7.4 LOG

Функция, предназначенная для возврата десятичного логарифма входного числа.
Входная переменная может быть типа REAL и LREAL, с соответствующим возвращаемым значением.
Пример:

```
q := LOG (36.6); //возвращаемое значение 1.5634810853944
```

3.7.5 EXP

Функция, предназначенная для возврата значения экспоненты, возведенной в степень входного числа.
Входная переменная может быть типа REAL и LREAL, с соответствующим возвращаемым значением.
Пример:

```
q := EXP (2.0); //возвращаемое значение 7.3890561
```

3.7.6 SIN

Функция, предназначенная для возврата значения sin (синуса) входного числа.
Входная переменная может быть типа REAL и LREAL, с соответствующим возвращаемым значением.
Пример:

```
q := SIN (0.5); //возвращаемое значение 0.4794255386
```

3.7.7 COS

Функция, предназначенная для возврата значения cos (косинуса) входного числа.

Входная переменная может быть типа REAL и LREAL, с соответствующим возвращаемым значением.

Пример:

```
q := COS (0.7); //возвращаемое значение 0.7648421873
```

3.7.8 TAN

Функция, предназначенная для возврата значения tg (тангенса) входного числа.

Входная переменная может быть типа REAL и LREAL, с соответствующим возвращаемым значением.

Пример:

```
q := TAN (0.33); //возвращаемое значение 0.342654129
```

3.7.9 ASIN

Функция, предназначенная для возврата значения arcsin (арксинуса) входного числа.

Входная переменная может быть типа REAL и LREAL, с соответствующим возвращаемым значением.

Пример:

```
q := ASIN (0.5); //возвращаемое значение 0.5235987756
```

3.7.10 ACOS

Функция, предназначенная для возврата значения arccos (арккосинуса) входного числа.

Входная переменная может быть типа REAL и LREAL, с соответствующим возвращаемым значением.

Пример:

```
q := ACOS (0.7); //возвращаемое значение 0.7953988302
```

3.7.11 ATAN

Функция, предназначенная для возврата значения arctg (арктангенса) входного числа.

Входная переменная может быть типа REAL и LREAL, с соответствующим возвращаемым значением.

Пример:

```
q := ATAN (0.33); //возвращаемое значение 0.3187475604
```

3.7.12 EXPT

Функция, предназначенная для возврата значения входного числа IN1 в степени IN2.

Входная переменная IN1 может быть типа REAL и LREAL, IN2 любого числового типа, возвращаемое значение типа REAL и LREAL.

Пример:

```
var1 := EXPT (7.1,2); //возвращаемое значение 50.41
```

3.7.13 TRUNC

Функция, предназначенная для возврата целой части входного числа.

Входная переменная может быть типа REAL и LREAL, возвращаемое значение типа LINT.

Пример:

```
x := TRUNC (7.9);  //возвращаемое значение 7
y := TRUNC (-0.9); //возвращаемое значение 0
```

3.7.14 ROUND

Функция, предназначенная для округления входного числа до ближайшего целого.

Входная переменная может быть типа REAL и LREAL, с соответствующим возвращаемым значением.

Пример:

```
q := ROUND (2.3); //возвращаемое значение 2.0
```

3.7.15 ATAN2

Функция, предназначенная для возврата значения угла направления вектора.

Входная переменная может быть типа REAL и LREAL, с соответствующим возвращаемым значением.

Пример:

```
q := ATAN2 (0.0,-1.0); //возвращаемое значение 3.141592653589793
```

3.8 Операторы выборки

3.8.1 MAX

Функция, предназначенная для возврата наибольшего значения из двух входных переменных.

Входные переменные и возвращаемое значение могут быть любого числового типа данных.

Пример:

```
OUT := MAX (10,12); //возвращаемое значение 12
```

3.8.2 MIN

Функция, предназначенная для возврата наименьшего значения из двух входных переменных.

Входные переменные и возвращаемое значение могут быть любого числового типа данных.

Пример:

```
OUT := MIN (10,12); //возвращаемое значение 10
```

3.8.3 LIMIT

Функция, предназначенная для возврата ограниченного значения входной переменной.

Входные переменные и возвращаемое значение могут быть любого числового типа данных.

```
OUT := LIMIT(Min, IN, Max);
```

MIN и MAX задают нижнюю и верхнюю границы значений.

Пример:

```
OUT := LIMIT (0,5,10);   //возвращаемое значение 5
OUT := LIMIT (0,-2,10);  //возвращаемое значение 0
OUT := LIMIT (0,-15,10); //возвращаемое значение 10
```

3.9 Валидаторы

3.9.1 IS_VALID

Функция, предназначенная для проверки корректности значения: не содержит NaN (Not-a-Number, «не число») и Inf (Infinity, «бесконечность»). Возвращает значение TRUE, если значение корректно.

Входные переменные могут быть типа REAL и LREAL, возвращаемое значение типа BOOL.

Пример:

```
VAR
    x:   LREAL := 12.5;
    OUT: BOOL;
END_VAR

OUT := IS_VALID (x); //возвращаемое значение TRUE, т.к. x — корректное число
```

3.9.2 IS_VALID_BCD

Функция, предназначенная для проверки корректности значения двоично-десятичного кода и возврата значения TRUE, если значение корректно.

Входные переменные могут быть типа BOOL, BYTE, WORD, DWORD, LWORD, возвращаемое значение типа BOOL.

Пример:

```
VAR
    bcdVal: WORD := 16#1234;  //BCD: 1-2-3-4
    OUT:    BOOL;
END_VAR

OUT := IS_VALID_BCD (bcdVal); //возвращаемое значение TRUE
```

3.10 Операции с временными типами данных

Подробнее о временных типах данных см. в Руководстве пользователя ALTA IDE.

3.10.1 CONCAT_DATE, _TOD, _LTOD, _DT, _LDT

Функция, предназначенная для конкатенации (объединения) целочисленных переменных с возвратом значения во временном формате.

Входные переменные могут быть

- любого целочисленного типа, возвращаемое значение DATE, TOD, LTOD, DT, LDT;
- типа DATE и TOD, возвращаемое значение DT;
- типа DATE и LTOD, возвращаемое значение DT.

Пример:

```
VAR
    y:      INT := 2026;
    m:      INT := 11;
    d:      INT := 15;
    myDate: DATE;
END_VAR

myDate := CONCAT_DATE (y, m, d); //возвращаемое значение 2026-11-15
```

3.10.2 SPLIT_DATE, _TOD, _LTOD, _DT, _LDT

Функция, предназначенная для разделения временных переменных.

Входные переменные могут быть типов DATE, TOD, LTOD, DT, LDT, возвращаемое значение INT.

ПРИМЕЧАНИЕ
В данной реализации возвращаемое значение всегда будет равно 0. Функцию можно использовать для разделения входной временной переменной на выходные переменные целочисленного типа.

Пример:

```
VAR

    myDate: DATE := D#2026-02-20;
    y:      INT;
    m:      INT;
    d:      INT;
    f:      INT;
END_VAR

f := SPLIT_DATE (myDate, y, m, d); //выходные переменные y=2026, m=02, d=20
```

3.10.3 ADD_TIME, _TOD, _DT, INSTANT

ПРИМЕЧАНИЕ
В текущей версии ALTA IDE сложение временных типов с помощью функции ADD не поддерживается. Воспользуйтесь оператором "+"

Функции, предназначенные для сложения переменных временных типов.

Наименование функции	Входные переменные	Выход функции	Пример
ADD_TIME	TIME	TIME	T#1s + T#500ms = T#1s500ms
ADD_TOD_TIME	TIME, TOD	TOD	Прибавляет временной интервал к времени суток, время суток смещается вперед на интервал: T#10s + TOD#12:00:00 = TOD#12:00:10
ADD_DT_TIME	DT, TIME	DT	Прибавляет временной интервал к дате-времени, время суток смещается вперед на интервал: T#1h + DT#2026-02-20-12:00:00 = DT#2026-02-20-13:00:00
ADD_INSTANT_TIME	INSTANT, TIME	INSTANT	T#1h + 25ns = 1h25ns

3.10.4 SUB_TIME, _DATE, _TOD, _DT, INSTANT

ПРИМЕЧАНИЕ
В текущей версии ALTA IDE вычитание временных типов с помощью функции SUB не поддерживается. Воспользуйтесь оператором "-"

Функции, предназначенные для вычитания переменных временных типов.

Наименование функции	Входные переменные	Выход функции	Пример
SUB_TIME	TIME	TIME	T#15s - T#8s = T#7s
SUB_DATE_DATE	DATE	TIME	D#2026-02-20 - D#2026-02-18 = T#48h
SUB_TOD_TIME	TOD, TIME	TOD	TOD#12:00:10 - T#5s = TOD#12:00:05
SUB_TOD_TOD	TOD	TIME	TOD#12:00:10 - TOD#12:00:03 = TIME#7s
SUB_DT_TIME	DT, TIME	DT	DT#2026-02-20-12:00:10 - T#5s = DT#2026-02-20-12:00:05

Наименование функции	Входные переменные	Выход функции	Пример
SUB_DT_DT	DT	TIME	DT#2026-02-20-12:00:10 - DT#2026-02-20-12:00:03 = T#7s
SUB_INSTANT_INSTANT	INSTANT	INSTANT	12ns - 5 ns= T#7ns
SUB_INSTANT_INSTANT	INSTANT	TIME	12ns - T#7ns = 5 ns

3.10.5 MUL_TIME

ПРИМЕЧАНИЕ
В текущей версии ALTA IDE умножение временных типов с помощью функции MUL не поддерживается. Воспользуйтесь оператором "*"

Функция, предназначенная для умножения временной переменной на число.

Входная переменная TIME и переменная любого числового типа данных, возвращаемое значение типа TIME.

Пример:

```
t := MUL_TIME (T#2s, 3);//возвращаемое значение T#6s
```

3.10.6 DIV_TIME

ПРИМЕЧАНИЕ
В текущей версии ALTA IDE деление временных типов с помощью функции DIV не поддерживается. Воспользуйтесь оператором "/"

Функция, предназначенная для деления временной переменной на число.

Входная переменная TIME и переменная любого числового типа данных, возвращаемое значение типа TIME.

Пример:

```
t1 := DIV_TIME (T#10s, 4);//возвращаемое значение T#2s500ms
```

3.10.7 TIME

Функция, предназначенная для возврата значения местного времени, начиная с 00.00 текущих суток.

Возвращаемое значение типа TIME.

Пример:

```
VAR
    CurTime: TIME;
END_VAR

CurTime := TIME (); //возвращаемое значение - текущее время, например T#6h41m30s
```

3.10.8 INSTANT_NOW

Функция, предназначенная для возврата значения времени от начала запуска приложения пользователя на устройстве.

Возвращаемое значение типа INSTANT.

Пример:

```
VAR
    StartTime: INSTANT;
END_VAR

StartTime := INSTANT_NOW (); //возвращаемое значение - время от начала запуска
приложения, например 1h15m30s
```

3.11 Преобразователи

Преобразование типов происходит с помощью функций-преобразователей.

Пример:

```
VAR
    wVal: WORD := 16#00FA;  //250
    iVal: INT;
END_VAR

iVal := WORD_TO_INT (wVal); //iVal = 250
```

Тип данных	Может быть преобразован
LWORD	DWORD, WORD, BYTE, BOOL, LREAL, LINT, DINT, INT, SINT, ULINT, UDINT, UINT, USINT, STRING, WSTRING, DATE, LDATE, DT, LDT, TOD, LTOD, TIME, LTIME
DWORD	LWORD, WORD, BYTE, BOOL, REAL, LINT, DINT, INT, SINT, ULINT, UDINT, UINT, USINT, STRING, WSTRING
WORD	LWORD, DWORD, BYTE, BOOL, LINT, DINT, INT, SINT, ULINT, UDINT, UINT, USINT
BYTE	LWORD, DWORD, WORD, BOOL, CHAR, LINT, DINT, INT, SINT, ULINT, UDINT, UINT, USINT, STRING, WSTRING
BOOL	LWORD, DWORD, WORD, BYTE, LINT, DINT, INT, SINT, ULINT, UDINT, UINT, USINT
CHAR	BYTE, WORD, DWORD, LWORD, STRING, WCHAR
WCHAR	WORD, DWORD, LWORD, WSTRING, CHAR
REAL	DWORD, STRING, WSTRING, LREAL, LINT, DINT, INT, SINT, ULINT, UDINT, UINT, USINT
LREAL	LWORD, STRING, WSTRING, REAL, LINT, DINT, INT, SINT, ULINT, UDINT, UINT, USINT, TIME
SINT	LWORD, DWORD, WORD, BYTE, LREAL, REAL, LINT, DINT, INT, ULINT, UDINT, UINT, USINT
INT	LWORD, DWORD, WORD, BYTE, LREAL, REAL, LINT, DINT, SINT, ULINT, UDINT, UINT, USINT
DINT	LWORD, DWORD, WORD, BYTE, BOOL, LREAL, REAL, LINT, INT, SINT, ULINT, UDINT, UINT, USINT
LINT	LWORD, DWORD, WORD, BYTE, STRING, WSTRING, LREAL, REAL, DINT, INT, SINT, ULINT, UDINT, UINT, USINT, TIME, TOD, DATE, BOOL
USINT	LWORD, DWORD, WORD, BYTE, STRING, LREAL, REAL, LINT, DINT, INT, SINT, ULINT, UDINT, UINT
UINT	LWORD, DWORD, WORD, BYTE, STRING, LREAL, REAL, LINT, DINT, INT, SINT, ULINT, UDINT, USINT
UDINT	LWORD, DWORD, WORD, BYTE, STRING, LREAL, REAL, LINT, DINT, INT, SINT, ULINT, UINT, USINT
ULINT	LWORD, DWORD, WORD, BYTE, STRING, LREAL, REAL, LINT, DINT, INT, SINT, UDINT, UINT, USINT, DATE, DT, TOD, TIME
STRING	LINT, DINT, REAL, LREAL, WSTRING, CHAR
WSTRING	LINT, DINT, REAL, LREAL, WCHAR
TIME	STRING, WSTRING, LWORD, LINT, ULINT, LREAL
DATE	STRING, WSTRING, LWORD, LINT, ULINT
TOD	STRING, WSTRING, LWORD, LINT, ULINT
DT	STRING, WSTRING, LWORD, LINT, ULINT, DATE, TOD

ЦИФРОВЫЕ
РЕШЕНИЯ

ООО "Овен Цифровые решения"
Россия, г. Москва, пл. Семёновская, д. 1А, помещ. 3/1
support@owendigital.ru
www.owendigital.ru

рег.:1-RU-156534-1.12