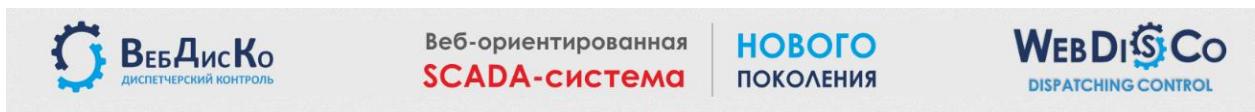


ООО «ФИОРД-ПРО»

## WebDisCo 2.4: Скрипты

---



ООО «ФИОРД-ПРО»



25.12.2023

## Оглавление

Соглашения о реализации языка скриптов для работы в среде WebDisCo .....	7
Функции языка скриптов для работы с отчетами и графиками в среде WebDisCo .....	8
Список специальных функций BASIC для работы в среде WebDisCo.....	9
1. WD_READ Переменной BASIC присвоить значение переменной WebDisCo.....	9
LET <переменная BASIC> =WD_READ "<путь переменной WebDisCo>" "<имя переменной WebDisCo>" "<параметр переменной WebDisCo>" .....	9
Аргументы:.....	9
Параметр переменной WebDisCo .....	9
Пример 1:.....	10
Пример 2:.....	10
2. WD_WRITE Переменной1 WebDisCo присвоить значение переменной2 WebDisCo.....	11
LET <переменная BASIC> = WD_WRITE "<путь переменной1 WebDisCo>" "<имя переменной1 WebDisCo>" "<путь переменной2 WebDisCo>" "<имя переменной2 WebDisCo>" "<параметр переменной WebDisCo>" .....	11
Аргументы:.....	11
Пример: .....	11
Запись данных истории для функции WD_WRITE.....	11
3. WD_ASSIGN Переменной WebDisCo присвоить значение переменной BASIC.....	12
LET <переменная1 BASIC> = WD_ASSIGN "<путь переменной WebDisCo>" "<имя переменной WebDisCo>" <переменная2 BASIC> .....	12
Аргументы: .....	12
Пример: .....	12
Запись данных истории для функции WD_ASSIGN .....	12
4. WD_SENDB Переменной WebDisCo присвоить значение переменной BASIC и выполнить анимацию «Команда» по присвоенному значению переменной WebDisCo .....	13
LET <переменная1 BASIC> = WD_SENDB "<путь переменной WebDisCo>" "<имя переменной WebDisCo>" <переменная2 BASIC> .....	13
Аргументы: .....	13
Пример: .....	13
Запись данных истории для функции WD_SENDB .....	14
5. WD_SEND Выполнить анимацию «Команда» по значению переменной WebDisCo.....	14
LET <переменная BASIC> = WD_SEND "<путь переменной WebDisCo>" "<имя переменной WebDisCo>" .....	14
Аргументы: .....	14
Пример: .....	14
Запись данных истории для функции WD_SEND .....	14

6.	WD_DELAY Сделать задержку в скрипте.....	14
	LET <переменная BASIC> = WD_DELAY <число микросекунд> .....	14
	Аргументы: .....	15
	Задержка для корректной записи истории переменной WebDisCo при выполнении функций WD_SENDB и WD_SEND .....	15
7.	WD_ALARMCOUNT Переменной BASIC присвоить значение числа активных тревог переменной BASIC.....	17
	LET <переменная BASIC> = WD_ALARMCOUNT.....	17
	Аргументы: .....	17
	Пример: .....	17
8.	WD_WRITELOG Сделать запись в Обозревателе событий .....	17
	LET <переменная BASIC> = WD_WRITELOG "Текст сообщения" .....	17
	Аргументы: .....	17
	Пример 1: .....	17
	Пример 2: .....	18
9.	WD_PARM Значение переменной BASIC, с которым был вызван скрипт.....	18
	LET <переменная BASIC> = WD_PARM .....	18
	Аргументы: .....	18
	Пример: .....	18
	Пример1 программы с разными ветками исполнения в зависимости от входного значения .....	18
	Пример2 программы с разными ветками исполнения в зависимости от входного значения .....	19
10.	WD_STOP Остановить выполнение всех включенных и работающих в данный момент скриптов с заданным именем .....	19
	LET <переменная BASIC> = WD_STOP "Имя скрипта" .....	19
	Аргументы: .....	20
	Пример: .....	20
11.	WD_START Запустить выполнение скрипта с заданным именем и параметром .....	20
	LET <переменная2 BASIC> = WD_START "<Имя скрипта>" <переменная1 BASIC> .....	20
	Аргументы: .....	20
	Пример: .....	20
	Использование скриптов в WebDisCo.....	21
	Таблица скриптов в режиме Разработки .....	21
	Редактор скриптов.....	22
	Таблица скриптов в режиме Исполнения .....	23

Способы вызова скриптов в режиме Исполнения .....	24
Соглашения о выполнении скриптов в WebDisCo .....	26
Сообщения Обозревателя событий WebDisCo при использовании BASIC.....	28
Список сообщений BASIC в Обозревателе событий .....	28
Примеры:.....	28
Пример программы на BASIC для работы с WebDisCo.....	28
Список стандартных встроенных функций BASIC для работы в среде WebDisCo.....	32
Работа с числами .....	32
ABS .....	32
ACS .....	32
ASN .....	32
ATN .....	32
BIN .....	32
COS .....	33
EXP.....	33
INT .....	33
LN .....	33
LOG .....	33
MOD .....	33
PI.....	33
POW .....	33
RND.....	34
SGN .....	34
SIN .....	34
SQR .....	34
TAN .....	34
Работа со строками .....	34
CHR\$.....	35
CODE.....	36
LEFT\$ .....	36
LEN.....	36
MID\$.....	37
RIGHT\$.....	37
SPC.....	37
STR\$.....	37

TL\$ .....	37
VAL .....	37
Логические операции.....	37
Операции сравнения.....	38
Массивы .....	38
Операции вывода .....	38
PRINT .....	38
READ..DATA.....	39
Список стандартных встроенных операторов BASIC для работы в среде WebDisCo.....	40
REM или ' .....	40
SWAP .....	40
LET .....	40
GOTO .....	40
FOR...TO...{STEP}...NEXT .....	42
Ограничения: .....	42
Пример: .....	42
IF...THEN...{ELSE}...ENDIF .....	43
Ограничения: .....	43
Пример : .....	43
WHILE <условие> DO...ENDWHILE .....	44
Ограничения: .....	45
Пример: .....	45
REPEAT... UNTIL <условие>.....	46
Ограничения: .....	47
Пример: .....	48
Пользовательские функции.....	48
DEF FN <имя функции> ({аргументы})=...функция.....	48
Ограничения: .....	48
FN <имя функции> ({аргументы}) .....	48
Пример 1 (функция с одним аргументом) .....	48
Пример 2 (функция с двумя аргументами) .....	49
Пример 3 (функция без аргументов) .....	49
Подпрограммы .....	49
RETURN .....	49
END.....	49

GOSUB .....	49
Ограничения: .....	49
Пример: .....	50
Пользовательские процедуры .....	50
DEFPROC <имя процедуры>...ENDPROC .....	50
CALLPROC <номер строки процедуры>.....	50
Ограничения: .....	52
Приложение: Комбинации клавиш редактора скриптов.....	52
Стандартный набор комбинаций клавиш .....	53
Комбинации клавиш по умолчанию.....	54
Комбинации клавиш истории действий.....	55

## Соглашения о реализации языка скриптов для работы в среде WebDisCo

В качестве языка скриптов в среде WebDisCo реализована ограниченная версия языка программирования BASIC. Скрипты выполняются сервером WebDisCo.

Соглашения о реализации BASIC для работы в WebDisCo:

1. Имена специальных функций BASIC для работы в среде WebDisCo имеют префикс WD\_. Например, WD\_Read, WD\_ASSIGN,...
2. Функции и операторы BASIC могут вводиться в любом (верхнем или нижнем) регистре. Переменные BASIC задаются в виде имен без кавычек. Например, N1, a\$
3. Имена переменных BASIC являются регистрозависимыми. То есть, имя переменной BASIC N не то же самое, что n.
4. Переменные WebDisCo в специальных встроенных функциях BASIC для работы с WebDisCo задаются в виде двух строковых литералов BASIC (то есть, в кавычках), первый из которых задает полный путь к переменной WebDisCo в дереве проекта WebDisco, а второй – имя переменной WebDisCo.
5. Такой формат принят потому, что имена переменных могут включать точки (например, при считывании орс тегов в WebDisCo автоматически создаются теги и потом переменные к ним с точками.) и при задании переменной WebDisCo в виде одного строкового литерала выделить из него, где путь и где имя переменной.
6. Каждая точка в имени пути переменной WebDisCo рассматривается разделитель между уровнями в иерархии узлов дерева переменных WebDisCo. Для корректного использования этой нотации в скриптах нельзя использовать узлы переменных, в именах которых есть точки. Иначе будет некорректно искаться соответствующий узел.
7. Путь для переменных WebDisCo в корне дерева переменных задается в виде пустого текстового литерала: "" .
8. В скриптах рекомендуется в программах на языке BASIC использовать номера строк, хотя бы на этапе отладки скриптов. Они помогут выявлять ошибки в программах, так как будут указываться в сообщении об ошибке.
9. Сообщения об ошибках в скриптах выдаются только для тех скриптов, которые выполняются одним из трех способов в WebDisCo.
10. Функции языка скриптов учитывают наличие качества переменных в режиме Исполнения, начиная с версии WebDisCo 2.4. Например, качество переменной в режиме Исполнения можно получить функцией WD\_READ с третьим аргументом "quality".
11. Скрипты с привязкой к переменным с плохим качеством не выполняются в режиме Исполнения.
12. В одной строке разрешено писать только одну строку языка BASIC.
13. Если в одной строке скрипта в WebDisCo надо записать несколько строк BASIC, то для разделения строк надо использовать символ \n.

### **Пример:**

Вот вариант записи текста скрипта в несколько строк:

FOR I = 1 TO 10

```
let m = I  
wd_assign "n" "Число2" m  
NEXT I
```

А вот так в одну строку

```
FOR I = 1 TO 10 \n let m = I \n wd_assign "n" "Число2" m NEXT I
```

## **Функции языка скриптов для работы с отчетами и графиками в среде WebDisCo**

В среде WebDisCo реализована возможность генерации отчетов и графиков из скриптов в отдельном документе «Функции языка скриптов для работы с отчетами и графиками в среде WebDisCo».

## Список специальных функций BASIC для работы в среде WebDisCo

### 1. WD\_READ Переменной BASIC присвоить значение переменной WebDisCo

**LET <переменная BASIC> =WD\_READ " <путь переменной WebDisCo>" " <имя переменной WebDisCo>" "<параметр переменной WebDisCo>"**

В <переменную BASIC> записывается последнее сохраненное значение параметров переменной WebDisCo. <Переменная WebDisCo> может быть любого типа – бит, число, или текст. <Переменная BASIC> принимает тип, соответствующий типу переменной WebDisCo (если третий аргумент – value, для битов и чисел – тип NUMBER, для текстов – STRING) и третьему аргументу.

#### **Аргументы:**

2 аргумента или 3 аргумента.

Аргументы: 1 – полный путь к имени переменной WebDisCo без имени переменной, 2 – имя переменной WebDisCo без пути, 3 (необязательно) – параметр переменной. Если это параметр не задан, то параметр переменной по умолчанию принимается равным "value".

#### **Параметр переменной WebDisCo**

<Параметр переменной WebDisCo> (третий аргумент функции WD\_READ) задает, что будет возвращаться. <Параметр переменной WebDisCo> можно вводить в любом регистре. Поэтому "value", "Value" или "VALUE" – это будет восприниматься как одно и то же. Значение <Параметр переменной WebDisCo> "value" и "quality" получаются (имеют смысл) только в режиме Исполнения. Остальные значения берутся из базы данных проекта (таблицы переменных):

The screenshot shows the WebDisCo software interface with two main windows. On the left is a tree view of the project structure under 'Проект'. On the right are two tables: 'Таблица переменных' (Variables Table) and 'Таблица требований' (Requirements Table).

**Таблица переменных (Variables Table):**

Название	Тип	Начальное знач...	Команда	История	Тревога	Мертвая зона	Комментарий /...	Связь с тегом
AltPressure	число	0	Нет	Нет		0		
AvgRPM	число	0	Нет	Да		0		
BrakeState	бит	false	Нет	Нет	Нет	0		
GenActivePower	число	0	Нет	Да		0	комментарий GenR...	
GenApparentPower	число	0	Нет	Нет		0		
GenRPM	число	0	Нет	Нет		0		
GenReactivePower	число	0	Да	Да		0	комментарий GenR...	
Humidity	число	0	Нет	Нет		0		
MaxRPM	число	0	Нет	Да		0		
MinRPM	число	0	Нет	Да		0		
MotorState	бит	false	Нет	Нет	Нет	0		
PitchAnglePlastet	число	0	Нет	Нет		0		

**Таблица требований (Requirements Table):**

Название	Тип	Уровень	Описание требований	Уведом...	Уведом...
None					

<Параметр переменной WebDisCo> (третий аргумент функции WD\_READ) может принимать следующие значения:

1. "value" - последнее сохраненное в режиме Исполнения значение переменной WebDisCo без учета качества. <Переменная BASIC> принимает тип, соответствующий типу переменной WebDisCo: для битов и чисел – тип NUMBER, для текстов – STRING.

2. "quality" - значение качества переменной WebDisCo в режиме Исполнения. 0 – плохое качество, 1 – хорошее качество. <Переменная BASIC> принимает тип NUMBER.
3. "type" – тип переменной WebDisCo: 0 – битовая переменная, 1 – числовая переменная, 2 – строка . <Переменная BASIC> принимает тип NUMBER.
4. "desc" – Комментарий переменной, <Переменная BASIC> принимает тип STRING.
5. "command" - значение признака Команда переменной WebDisCo. 0 – не является командой, 1 – команда. <Переменная BASIC> принимает тип NUMBER.
6. "archive" - значение признака История переменной WebDisCo. 0 – не имеет признака История, 1 – имеет признак История. <Переменная BASIC> принимает тип NUMBER.
7. "system" – является ли системной переменной WebDisCo. 0 – нет, 1 – да. <Переменная BASIC> принимает тип NUMBER.
8. "alarm" - значение признака Тревога переменной WebDisCo. 0 – не имеет признака Тревога для битовой переменной и для других типов переменных WebDisCo, 1 – имеет признак Тревога для битовой переменной. <Переменная BASIC> принимает тип NUMBER.

***Пример 1:***

```
LET N1 = WD_READ "basic" "Число" "value"
PRINT "BASIC переменной N1 присвоено значение числовой WebDisCo переменной basic.Число =
", N1, ""
LET N2 = WD_READ "basic" "Бит" "value"
PRINT "BASIC переменной N2 присвоено значение битовой WebDisCo переменной basic.Бит = ", N2,
"""
LET N3 = WD_READ "basic" "Строка" "value"
196 PRINT "BASIC Переменной N3 присвоено значение текстовой WebDisCo переменной
basic.Строка= ", N3, ""
285 LET N = WD_READ "opc.opc1" "Group1.Name101" "value"
290 PRINT "BASIC переменной N присвоено значение WebDisCo переменной
opc.opc1.Group1.Name101= ", N, ""
```

***Пример 2:***

```
LET Name = "GenActivePower"
LET Node ="fda.Test_FDA.fda"
LET V = WD_READ Node Name "value"
PRINT "V=",V, "\n"
LET Q = WD_READ Node Name "quality"
PRINT "Q=",Q, "\n"
LET T = WD_READ Node Name "type"
PRINT "T=",T, "\n"
LET D = WD_READ Node Name "desc"
PRINT "D=",D, "\n"
LET C = WD_READ Node Name "command"
PRINT "C=",C, "\n"
LET A = WD_READ Node Name "archive"
PRINT "A=",A, "\n"
LET S = WD_READ Node Name "system"
PRINT "S=",S, "\n"
LET AL = WD_READ Node Name "alarm"
```

```
PRINT "AL=",AL, "\n"
```

## 2. WD\_WRITE Переменной1 WebDisCo присвоить значение переменной2 WebDisCo

**LET <переменная BASIC> = WD\_WRITE "<путь переменной1 WebDisCo>" "<имя переменной1 WebDisCo>" "<путь переменной2 WebDisCo>" "<имя переменной2 WebDisCo>" "<параметр переменной WebDisCo>"**

В <переменную1 WebDisCo> записывается последнее сохраненное значение переменной2 WebDisCo. Типы переменных WebDisCo должны совпадать. <Переменная1 WebDisCo> может быть любого типа – бит, число, или текст. <Переменная BASIC> принимает тип, соответствующий типу переменной WebDisCo. Для битов и чисел – тип NUMBER, для текстов – STRING. В <переменную BASIC> записывается значение переменной1 WebDisCo.

### **Аргументы:**

4 или 5 аргументов.

Аргументы: 1 – полный путь к имени переменной1 WebDisCo без имени переменной, 2 – имя переменной1 WebDisCo без пути, 3 – полный путь к имени переменной1 WebDisCo без имени переменной1, 4 – имя переменной2 WebDisCo без пути, 5 (необязательно) – параметр переменной ("value" (только значение), "quality" (только качество), "value\_quality" (значение вместе с качеством)). Если 5-й параметр не задан, то параметр переменной по умолчанию принимается равным "value\_quality". <Параметр переменной WebDisCo> можно вводить в любом регистре. Поэтому "value", "Value" или "VALUE" – это будет восприниматься как одно и то же.

Если в качестве пути переменной1 WebDisCo указан SYSTEM, то при выполнении в режиме Исполнения будет выдаваться сообщение об ошибке, так как системные переменные не могут меняться пользователем.

Если переменная1 WebDisCo является битовой переменной с признаком тревога, то при выполнении скрипта в режиме Исполнения будет выдаваться сообщение об ошибке.

### **Пример:**

```
291 LET M = WD_WRITE "n" "Число2" "opc.opc1" "Group1.Name101"  
292 PRINT "Значение WebDisCo переменной opc.opc1.Group1.Name101 записано в  
WebDisCo переменную n.Число2, значение M=" ,M, ""
```

### **Запись данных истории для функции WD\_WRITE**

Если переменная1 WebDisCo имеет признак История, то при каждом присваивании в историю этой переменной будет записываться присваиваемое значение, даже, если оно совпадает с текущим значением этой переменной. Например, если выполнить семь раз (в цикле или планировщиком) скрипт WD\_WRITE "n" "Число2" "opc.opc1" "Group1.Name101", а значение переменной2 WebDisCo opc.opc1.Group1.Name101 не изменилось, то в истории переменной "n" "Число2" значение будет записано семь раз.

Если разработчик не хочет, чтобы данные истории дублировались при каждом присваивании переменной WebDisCo, он должен проверить, что новое значение этой переменной отличается от старого. Например, так:

```
LET N1 = WD_READ "n" "Число2" "value"  
LET N2 = WD_READ "opc.opc1" "Group1.Name101"  
IF N1 <> N2 THEN LET M=WD_WRITE "n" "Число2" "opc.opc1" "Group1.Name101"
```

### 3. WD\_ASSIGN Переменной WebDisCo присвоить значение переменной BASIC

**LET <переменная1 BASIC> = WD\_ASSIGN "<путь переменной WebDisCo>" "<имя переменной WebDisCo>" <переменная2 BASIC>**

Сначала в переменную WebDisCo записывается значение <переменной2 BASIC>. Переменная WebDisCo получает качество Хорошее (1). Потом в <переменную1 BASIC> записывается присвоенное значение <переменной2 BASIC> (этого требует синтаксис операции LET). <Переменная1 BASIC> принимает тип BASIC, соответствующий типу переменной WebDisCo. Для битов и чисел – тип NUMBER, для текста – STRING.

**Аргументы:**

3 аргумента.

Аргументы: 1 – полный путь к имени переменной WebDisCo без имени переменной, 2 – имя переменной WebDisCo без пути., 3 – имя переменной BASIC.

Если в качестве пути переменной WebDisCo указан SYSTEM, то при выполнении в режиме Исполнения будет выдаваться сообщение об ошибке, так как системные переменные не могут меняться пользователем.

**Пример:**

```
299 LET N1 = 2.62  
300 LET M1 =WD_ASSIGN "basic" "Число1" N1  
301 PRINT "WebDisCo переменной basic.Число1 присвоено значение BASIC переменной  
N1=", N1," M1=", M1, ""
```

**Запись данных истории для функции WD\_ASSIGN**

Если переменная WebDisCo имеет признак История, то при каждом присваивании в историю этой переменной будет записываться присваиваемое значение, даже, если оно совпадает с текущим значением этой переменной (например, 0.136). Например, если выполнить пять раз (в цикле или планировщиком) скрипт LET M = WD\_ASSIGN "cnt" "Energy\_per\_value" 0.136, то в истории переменной cnt.Energy\_per\_value значение 0.136 будет записано пять раз.

Время события	Название	Значение
29.09.2023, 15:45:00.933	среднее потребление энергии на среднюю единицу продукции	0.136
29.09.2023, 15:44:59.449	среднее потребление энергии на среднюю единицу продукции	0.136
29.09.2023, 15:44:57.786	среднее потребление энергии на среднюю единицу продукции	0.136
29.09.2023, 15:44:56.018	среднее потребление энергии на среднюю единицу продукции	0.136
29.09.2023, 15:44:54.541	среднее потребление энергии на среднюю единицу продукции	0.136
29.09.2023, 15:44:41.905	среднее потребление энергии на среднюю единицу продукции	0

Если разработчик не хочет, чтобы данные истории дублировались при каждом присваивании переменной WebDisCo, он должен проверить, что новое значение этой переменной отличается от старого. Например, так:

```
LET N1 = WD_READ "cnt" "Energy_per_value" "value"
IF N1 <> 0.136 THEN LET M = WD_ASSIGN "cnt" "Energy_per_value" 0.136
```

#### 4. WD\_SENDB Переменной WebDisCo присвоить значение переменной BASIC и выполнить анимацию «Команда» по присвоенному значению переменной WebDisCo

**LET <переменная1 BASIC> = WD\_SENDB "<путь переменной WebDisCo>" "<имя переменной WebDisCo>" <переменная2 BASIC>**

Основное назначение этой функции – выполнить анимацию «Команда» по значению <переменной2 BASIC>. Сначала в переменную WebDisCo записывается значение <переменной2 BASIC> (при этом переменная WebDisCo получает качество Хорошее) и выполняется анимация WebDisCo «Команда» (отправляется значение переменной WebDisCo).

При использовании в скрипте функции WD\_SENDB для корректной записи истории для переменной WebDisCo (если для переменной установлен признак История) использовать функцию [WD\\_DELAY](#).

##### **Аргументы:**

3 аргумента.

Аргументы: 1 – полный путь к имени переменной WebDisCo без имени переменной, 2 – имя переменной WebDisCo без пути,. 3 – имя переменной2 BASIC.

Переменная WebDisCo должна иметь установленный признак «Команда», иначе при выполнении этой функции будет выдаваться в Обозревателе событий сообщение об ошибке. Потом в <переменную1 BASIC> записывается присвоенное значение <переменной2 BASIC> (этого требует синтаксис операции LET). <Переменная BASIC1> принимает тип BASIC, соответствующий типу переменной WebDisCo. Для битов и чисел – тип NUMBER, для текста – STRING.

В качестве <переменной2 BASIC> можно задавать константу:

```
298 LET M2 = wd_sendb "mbtcp.modbus-node-2" "mbt5" 3.62
```

##### **Пример:**

```
296 LET M1 = wd_sendb "mbtcp.modbus-node-2" "mbt5" N1
```

```
297 PRINT "Из WebDisCo переменной mbtcp.modbus-node-2.mbt5 отправлена команда из  
BASIC переменной N1=", N1, " M1=", M1, ""
```

```
298 LET M2 = wd_sendb "mbtcp.modbus-node-2" "mbt5" 3.62
```

### **Запись данных истории для функции WD\_SENDB**

Если переменная WebDisCo имеет признак История, то при выполнении функции WD\_SENDB в историю этой переменной будет записываться присваиваемое значение, если оно не совпадает с текущим значением этой переменной.

## **5. WD\_SENDB Выполнить анимацию «Команда» по значению переменной WebDisCo**

**LET <переменная BASIC> = WD\_SENDB "<путь переменной WebDisCo>" "<имя  
переменной WebDisCo>"**

Назначение этой функции – выполнить анимацию «Команда» по значению переменной WebDisCo. (отправляется значение переменной WebDisCo). Если переменная WebDisCo имеет плохое качество, то функция не выполняется и делается запись об этом в Обозревателе событий.

При использовании в скрипте функции WD\_SENDB для корректной записи истории для переменной WebDisCo (если для переменной установлен признак История) использовать функцию [WD\\_DELAY](#).

#### **Аргументы:**

2 аргумента.

Аргументы: 1 – полный путь к имени переменной WebDisCo без имени переменной, 2 – имя переменной WebDisCo без пути.

Переменная WebDisCo должна иметь установленный признак «Команда», иначе при выполнении этой функции будет выдаваться в Обозревателе событий сообщение об ошибке. Потом в <переменную BASIC> записывается присвоенное значение переменной WebDisCo (этого требует синтаксис операции LET). <Переменная BASIC> принимает тип BASIC, соответствующий типу переменной WebDisCo. Для битов и чисел – тип NUMBER, для текста – STRING.

#### **Пример:**

```
296 LET M3 = wd_send "mbtcp.modbus-node-2" "mbt103"
```

```
297 PRINT "Из WebDisCo переменной mbtcp.modbus-node-2.mbt103 отправлена команда", "M3=",  
M3, ""
```

### **Запись данных истории для функции WD\_SEND**

Если переменная WebDisCo имеет признак История, то при выполнении функции WD\_SEND в историю этой переменной будет записываться присваиваемое значение, если оно не совпадает с текущим значением этой переменной.

## **6. WD\_DELAY Сделать задержку в скрипте**

**LET <переменная BASIC> = WD\_DELAY <число микросекунд>**

Назначение этой функции – выполнить задержку при выполнении скрипта на указанное число микросекунд. Аргумент <число микросекунд> должен быть NUMBER и быть не отрицательным

числом. Если это не так, то функция не выполняется и делается запись об этом в Обозревателе событий.

#### *Аргументы:*

1 аргумент.

Аргумент: 1 – не отрицательное число

Примеры:

wd\_delay 700000

WD\_DELAY(500000)

#### **Задержка для корректной записи истории переменной WebDisCo при выполнении функций WD\_SENDB и WD\_SEND**

Для корректной записи истории переменной WebDisCo требуется делать задержку, чтобы эти команды успели выполниться и изменить значение переменной WebDisCo.

Пример: Переменная WebDisCo basic.Coil2-wo с признаками История и Команда. Ее значение меняется в цикле 3 раза и, чтобы исторические данные были корректно записаны, требуется использовать небольшую задержку, как это показано в примере.

Пример:

FOR I = 1 TO 10

wd\_sendb "basic" "Coil2-wo" 1

WD\_DELAY(100000)

wd\_sendb "basic" "Coil2-wo" 0

WD\_DELAY(100000)

NEXT I

Исторические данные		
Время события	Название	Значение
24.10.2023, 08:45:09.876	basic.Coil2-wo	0
24.10.2023, 08:45:09.768	basic.Coil2-wo	1
24.10.2023, 08:45:09.658	basic.Coil2-wo	0
24.10.2023, 08:45:09.549	basic.Coil2-wo	1
24.10.2023, 08:45:09.438	basic.Coil2-wo	0
24.10.2023, 08:45:09.328	basic.Coil2-wo	1

Задержка в скрипте для записи истории переменной с привязкой к тегу должна быть больше периода опроса коммуникации. Иначе при опросе коммуникация прочитает непонятное

значение. Например, в Modbus TCP период опроса по умолчанию стоит 100 мс. Поэтому задержку в скрипте надо делать, например, в 120 мсек.

Подключение Modbus TCP								
Название	IP-адрес	Номер порта	Номер узла	Таймаут, мс	Период опроса, мс	Переподключение	Устройство	Включено
Modbus_sim	127.0.0.1	502	1	990	100	Да		Включено

Пример: Переменная WebDisCo mbtcp.Modbus\_sim с признаками История и Команда, а также с привязкой тегу Modbus TCP.. Ее значение меняется в цикле 10 раз и, чтобы исторические данные были корректно записаны, требуется использовать небольшую задержку, как это показано в примере.

```
FOR I = 1 TO 10
```

```

LET RR = WD_READ "mbtcp.Modbus_sim" "Coil2" "Value"
PRINT "RR=",RR,"\\n"
IF RR = 0 THEN LET RR1 = wd_sendb "mbtcp.Modbus_sim" "Coil2" 1
WD_DELAY(120000)
IF RR = 1 THEN LET RR1 = wd_sendb "mbtcp.Modbus_sim" "Coil2" 0
WD_DELAY(120000)
```

```
NEXT I
```

Исторические данные		
Время события	Название	Значение
24.10.2023, 08:53:43.035	mbtcp.Modbus_sim.Coil2	0
24.10.2023, 08:53:42.605	mbtcp.Modbus_sim.Coil2	1
24.10.2023, 08:53:42.495	mbtcp.Modbus_sim.Coil2	0
24.10.2023, 08:53:42.056	mbtcp.Modbus_sim.Coil2	1
24.10.2023, 08:53:41.947	mbtcp.Modbus_sim.Coil2	0
24.10.2023, 08:53:41.622	mbtcp.Modbus_sim.Coil2	1
24.10.2023, 08:53:41.512	mbtcp.Modbus_sim.Coil2	0
24.10.2023, 08:53:41.075	mbtcp.Modbus_sim.Coil2	1
24.10.2023, 08:53:40.967	mbtcp.Modbus_sim.Coil2	0
24.10.2023, 08:53:40.644	mbtcp.Modbus_sim.Coil2	1

## 7. WD\_ALARMCOUNT Переменной BASIC присвоить значение числа активных тревог переменной BASIC

**LET <переменная BASIC> = WD\_ALARMCOUNT**

В <переменную BASIC> записывается число активных тревог. <Переменная BASIC> принимает тип BASIC NUMBER.

### Аргументы:

Нет аргументов.

### Пример:

302 LET M2 = WD\_ALARMCOUNT

303 PRINT "\BASIC переменной M2 присвоено значение числа тревог = ", M2, ""

## 8. WD\_WRITELOG Сделать запись в Обозревателе событий

Назначение этой функции – сделать запись в Обозревателе событий (раздел Прочее). В <переменную BASIC> записывается текст сообщения.

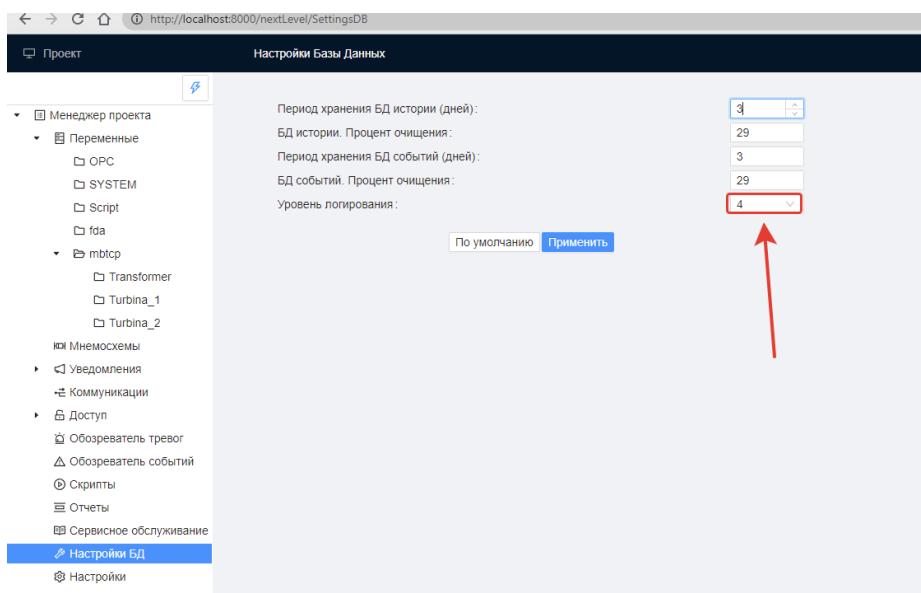
**LET <переменная BASIC> = WD\_WRITELOG "Текст сообщения"**

### Аргументы:

1 аргумент.

Аргументы: 1 – Текст сообщения.

<Переменная BASIC> принимает тип BASIC STRING. Чтобы выполнялись записи в Обозреватель сообщений из скриптов, необходима установка в Настройках БД уровня логирования равным 4.

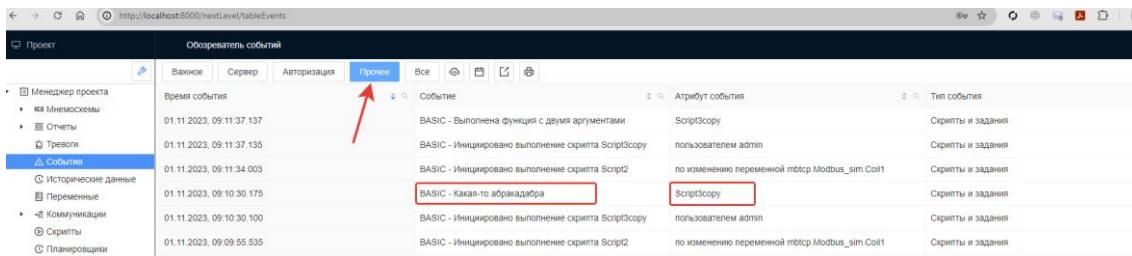


### Пример 1:

В скрипте Script3сору выполнена следующая функция:

306 LET M4 = WD\_WRITELOG "Какая-тоabra Kadabra"

Тогда в Обозревателе событий появится следующее сообщение:



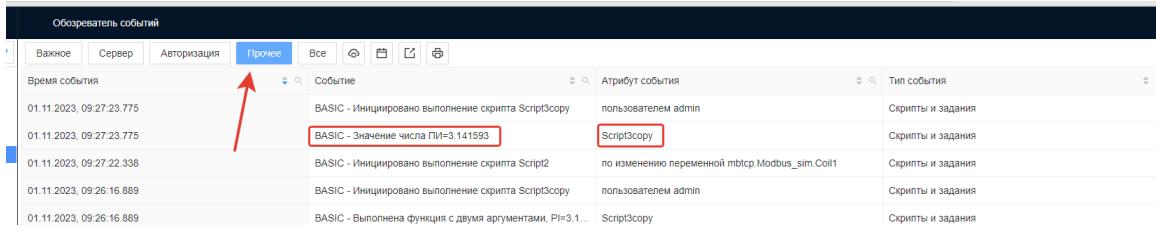
Время события	Событие	Атрибут события	Тип события
01.11.2023, 09:11:37.137	BASIC - Выполнена функция с двумя аргументами	Script3Copy	Скрипты и задания
01.11.2023, 09:11:37.135	BASIC - Инициировано выполнение скрипта Script3Copy	пользователем admin	Скрипты и задания
01.11.2023, 09:11:34.003	BASIC - Инициировано выполнение скрипта Script2	по изменению переменной mbtcp.Modbus_sim.Coil1	Скрипты и задания
01.11.2023, 09:10:30.175	<b>BASIC - Какая-то абракадабра</b>	Script3Copy	Скрипты и задания
01.11.2023, 09:10:30.100	BASIC - Инициировано выполнение скрипта Script3Copy	пользователем admin	Скрипты и задания
01.11.2023, 09:09:55.535	BASIC - Инициировано выполнение скрипта Script2	по изменению переменной mbtcp.Modbus_sim.Coil1	Скрипты и задания

### Пример 2:

В скрипте Script3Copy выполнена следующая функция:

WD\_WRITELOG "Значение числа ПИ="+STR\$(PI)

Тогда в Обозревателе событий появится следующее сообщение:



Время события	Событие	Атрибут события	Тип события
01.11.2023, 09:27:23.775	BASIC - Инициировано выполнение скрипта Script3Copy	пользователем admin	Скрипты и задания
01.11.2023, 09:27:23.775	<b>BASIC - Значение числа ПИ=3.141593</b>	Script3Copy	Скрипты и задания
01.11.2023, 09:27:22.338	BASIC - Инициировано выполнение скрипта Script2	по изменению переменной mbtcp.Modbus_sim.Coil1	Скрипты и задания
01.11.2023, 09:26:16.889	BASIC - Инициировано выполнение скрипта Script3Copy	пользователем admin	Скрипты и задания
01.11.2023, 09:26:16.889	BASIC - Выполнена функция с двумя аргументами. PI=3.1...	Script3Copy	Скрипты и задания

## 9. WD\_PARM Значение переменной BASIC, с которым был вызван скрипт

### LET <переменная BASIC> = WD\_PARM

В <переменную BASIC> записывается следующее значение:

9. 0, если скрипт был вызван по инициативе оператора (из анимации «Выполнить скрипт»). <Переменная BASIC> принимает тип BASIC NUMBER.
10. 1, если скрипт был вызван планировщиком. <Переменная BASIC> принимает тип BASIC NUMBER.
11. Значение битовой переменной WebDisCo, которая привязана к скрипту. <Переменная BASIC> принимает тип BASIC NUMBER. Рекомендуется дополнительно использовать функцию WD\_READ с параметром "quality" для чтения качества переменной WebDisCo.
12. Значению второго параметра функции WD\_START, если скрипт был вызван из этой функции. <Переменная BASIC> принимает тип второго параметра функции WD\_START (BASIC NUMBER или BASIC STRING).

### Аргументы:

Нет аргументов.

### Пример:

11 LET B = WD\_PARM

12 PRINT "Скрипт вызван по изменению переменной Бит1, которая приняла значение = ", B, ""

### Пример1 программы с разными ветками исполнения в зависимости от входного значения

Если измененное значение, по которому вызван скрипт, равно 0, то ничего не выполняется:

```
10 LET B = WD_PARM  
20 if B = 0 then gosub 80  
30 FOR I = 1 TO 100  
40 let m = I  
50 wd_assign "n" "Число2" m  
60 NEXT I  
70 wd_assign "n" "бит1" 1  
80 END
```

### *Пример2 программы с разными ветками исполнения в зависимости от входного значения*

В зависимости от измененного значения, по которому вызван скрипт, выполняются разные действия.

```
10 LET B = WD_PARM  
20 if B = 0 then gosub 130  
30 FOR I = 1 TO 100  
40 let m = I  
50 wd_assign "n" "Число2" m  
60 NEXT I  
70 wd_assign "n" "бит1" 1  
80 END  
130 FOR I = 1 TO 30  
140 let m = I*3  
150 wd_assign "n" "Число2" m  
160 NEXT I  
170 END
```

## **10. WD\_STOP Остановить выполнение всех включенных и работающих в данный момент скриптов с заданным именем**

### **LET <переменная BASIC> = WD\_STOP "Имя скрипта"**

В <переменную BASIC> записывается значение 1, если скрипт был остановлен при выполнении, и 0 во всех остальных случаях (отключен, не найден,...). <Переменная BASIC> принимает тип BASIC NUMBER.

**Аргументы:**

1 аргумент.

Аргументы: 1 – Имя скрипта.

Эта функция предназначена для таких целей, как принудительный останов зациклившихся скриптов или останов выполнения скрипта при определенной ситуации. Останов выполнения скрипта происходит сразу после выполнения очередного оператора языка BASIC.

**Пример:**

11 LET B = WD\_STOP "Скрипт1"

## **11. WD\_START Запустить выполнение скрипта с заданным именем и параметром**

**LET <переменная2 BASIC> = WD\_START "<Имя скрипта>" <переменная1 BASIC>**

В <переменную BASIC> записывается значение 1, если скрипт был успешно запущен и 0 во всех остальных случаях (отключен, не найден,...).

**Аргументы:**

2 аргумента.

Аргументы: 1 – Имя скрипта, 2 – переменная1 BASIC.

<Переменная1 BASIC> может иметь тип NUMBER или STRING.

<Переменная2 BASIC> принимает тип BASIC NUMBER.

Эта функция предназначена для запуска одного скрипта из другого. Запускаемый скрипт будет выполняться параллельно с запускающим скриптом. Скрипт, который запускается должен иметь статус «Включен», в противном случае он не будет запущен.

Наличие функции WD\_START позволяет использовать какой-то скрипт многократно и одновременно в других скриптах. Например, для выполнения какого-то типового действия.

2 аргумента.

Аргументы: 1 – имя вызываемого скрипта, 2 – число или строка

Аргумент 2 позволяет передать в вызываемый скрипт значение, которое может быть обработано функцией WD\_PARM в вызываемом скрипте.

**Пример:**

11 LET B = WD\_START "Скрипт1" "Передаем в функцию Скрипт1 текст"

12 LET B = WD\_START "Скрипт1" 1.23

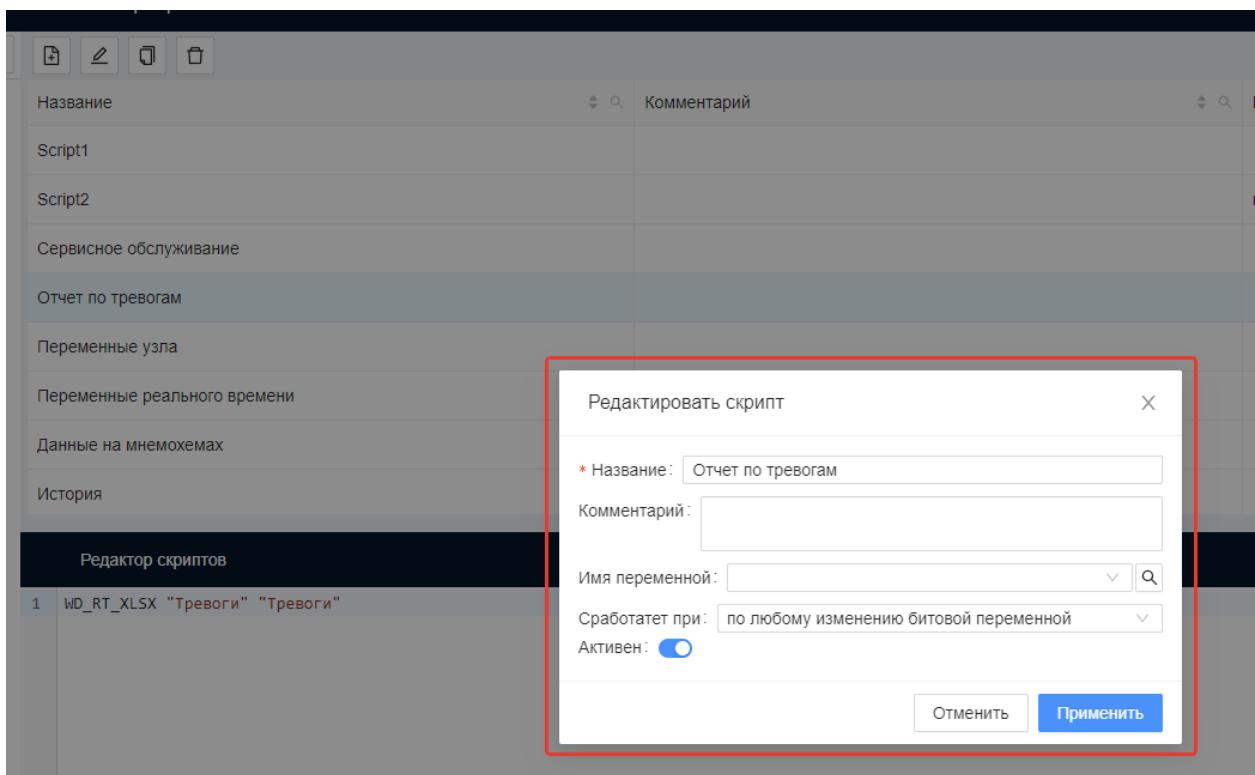
## Использование скриптов в WebDisCo

### Таблица скриптов в режиме Разработки

Для работы в режиме Разработки со скриптами в WebDisCo используется пункт менеджера проекта **Скрипты**. В режиме Разработки этот пункт меню доступен всем пользователям.

Работа с таблицей скриптов выполняется только в режиме Разработки с помощью панели инструментов, в которой определены кнопки добавления, редактирования, копирования и удаления данных о скрипте:

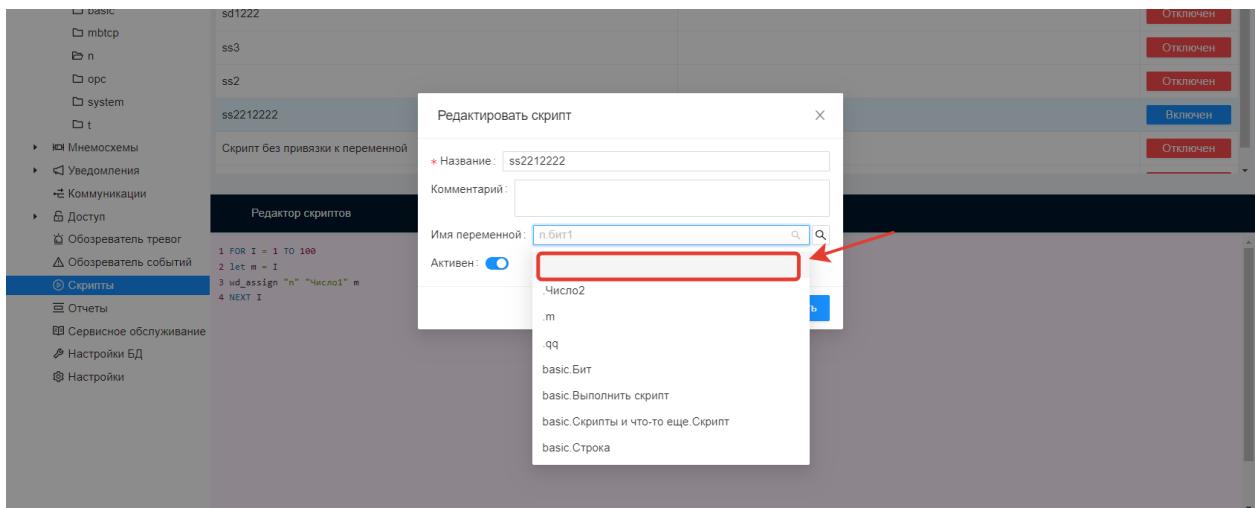
Каждый скрипт имеет следующие атрибуты - имя, комментарий, признак активности и имя привязанной переменной.



Признак активности используется в режиме Исполнения. Если он не установлен, то скрипт не будет запускаться в режиме Исполнения.

Имя привязанной переменной используется в режиме Исполнения для того, чтобы указать при изменении какой битовой переменной будет запускаться скрипт. Так как битовая переменная имеет два состояния, то могут выполняться различные действия в зависимости от того, какое значение приняла эта битовая переменная. Для этого можно использовать специальную функцию [WD\\_PARM](#). Примеры такого подхода приведены в разделе об этой функции.

Для того, чтобы отвязать от скрипта переменную надо в выпадающем списке выбрать пустую строку. Пустая строка - самая первая в выпадающем списке.



## Редактор скриптов

Редактор скриптов позволяет редактировать текст скриптов.

Скриншот интерфейса WebDisCo 2.4 в режиме Редактор скриптов. В левом меню выбран пункт "Скрипты". В центральной части экрана отображается таблица скриптов с двумя строками: Script1 и Script2. Кнопка "Изменить" для Script1 выделена красным квадратом. В правой части таблицы приведены столбцы: Название, Комментарий, Имя переменной и Состояние. Код скрипта Script1 выделен красным квадратом и показан в нижней панели "Редактор скриптов". Код скрипта:

```

1 LET B = #D_TODAY+"100:00"
2 LET E = #D_TODAY+"123:59"
3 PRINT "B="..B, " E=", E, "\n"
4 LET H = STR$(#D_HOUR_OF_DAY)
5 PRINT "HOUR=", H, "\n"
6 LET M = STR$(#D_MINUTE_OF_HOUR)
7 PRINT "MINUTE=", M, "\n"
8 LET S = STR$(#D_SECOND_OF_MINUTE)
9 PRINT "SECOND=", S, "\n"
10 LET N = "ГРАФИК->НЧ-->НЧ-->5"
11 PRINT N
12 LET H = #D_HISTORY_CHART "trend" N "fda.Test_FDA.fda" "RotorSpeed" B E
13 #D_HISTORY_CHART2PDF N N "L"

```

При этом можно использовать удобный набор комбинаций клавиш.

## Таблица скриптов в режиме Исполнения

Для работы в режиме Разработки со скриптами в WebDisCo используется пункт менеджера проекта **Скрипты**. В режиме Разработки этот пункт меню доступен только супер администратору admin. Он может просматривать текст скриптов и менять их состояние. Редактирование скриптов в режиме Исполнения невозможно.

Скриншот интерфейса WebDisCo 2.4 в режиме Таблица скриптов. В левом меню выбран пункт "Скрипты". В центральной части экрана отображается таблица скриптов с двумя строками: Script1 и Script2. Кнопка "Изменить" для Script1 выделена красным квадратом. В правой части таблицы приведены столбцы: Название, Комментарий, Имя переменной и Состояние. Кнопка "Включить" для Script1 выделена красным квадратом и показана в нижней панели "Редактор скриптов". Код скрипта Script1 выделен красным квадратом и показан в нижней панели "Редактор скриптов". Код скрипта:

```

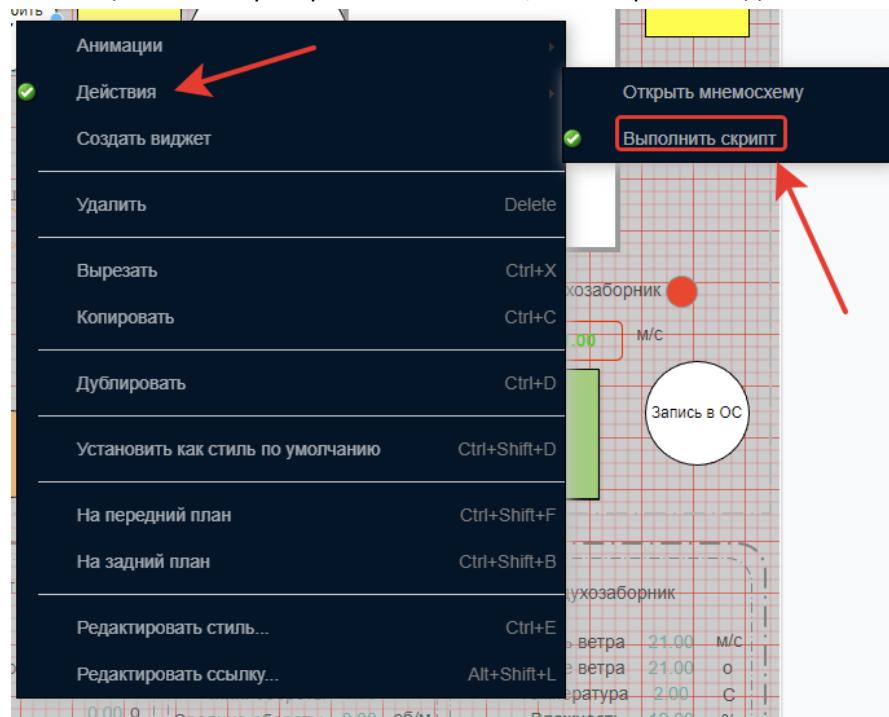
1 LET B = #D_TODAY+"100:00"
2 LET E = #D_TODAY+"123:59"
3 PRINT "B="..B, " E=", E, "\n"
4 LET H = STR$(#D_HOUR_OF_DAY)
5 PRINT "HOUR=", H, "\n"
6 LET M = STR$(#D_MINUTE_OF_HOUR)
7 PRINT "MINUTE=", M, "\n"
8 LET S = STR$(#D_SECOND_OF_MINUTE)
9 PRINT "SECOND=", S, "\n"
10 LET N = "ГРАФИК->НЧ-->НЧ-->5"
11 PRINT N
12 LET H = #D_HISTORY_CHART "trend" N "fda.Test_FDA.fda" "RotorSpeed" B E
13 #D_HISTORY_CHART2PDF N N "L"

```

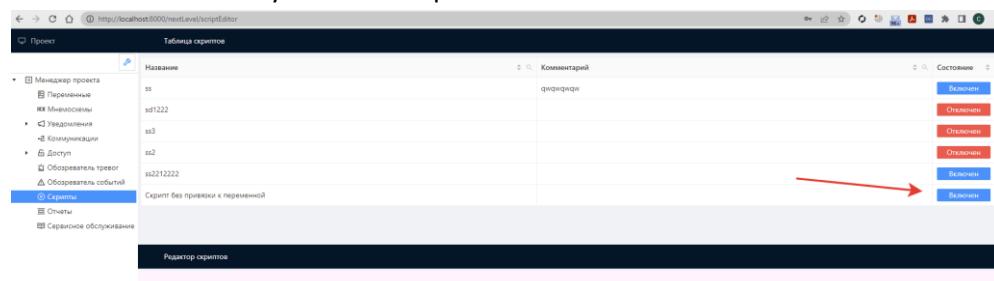
## Способы вызова скриптов в режиме Исполнения

Скрипты в WebDisCo в режиме Исполнения могут вызываться тремя способами:

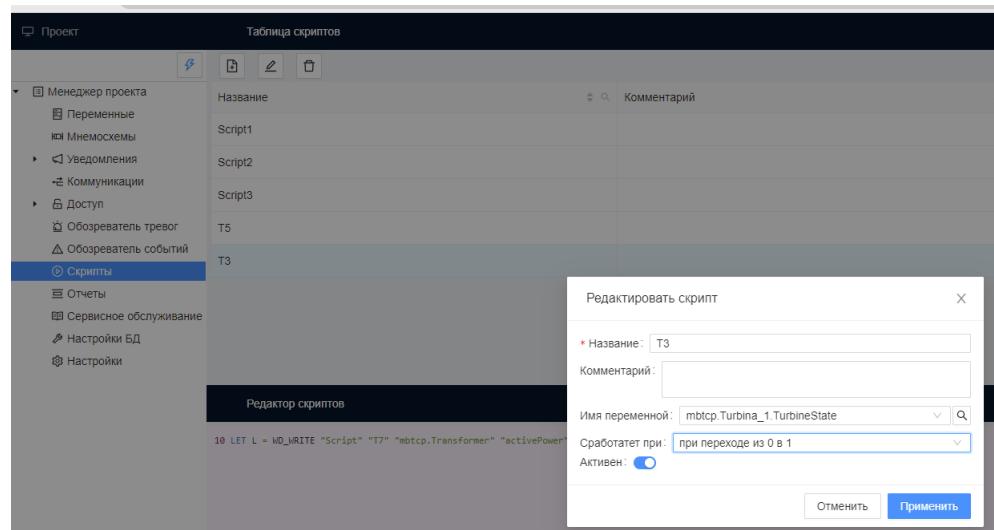
1. По инициативе оператора в мнемосхеме, на которой есть действие **Выполнить скрипт**.



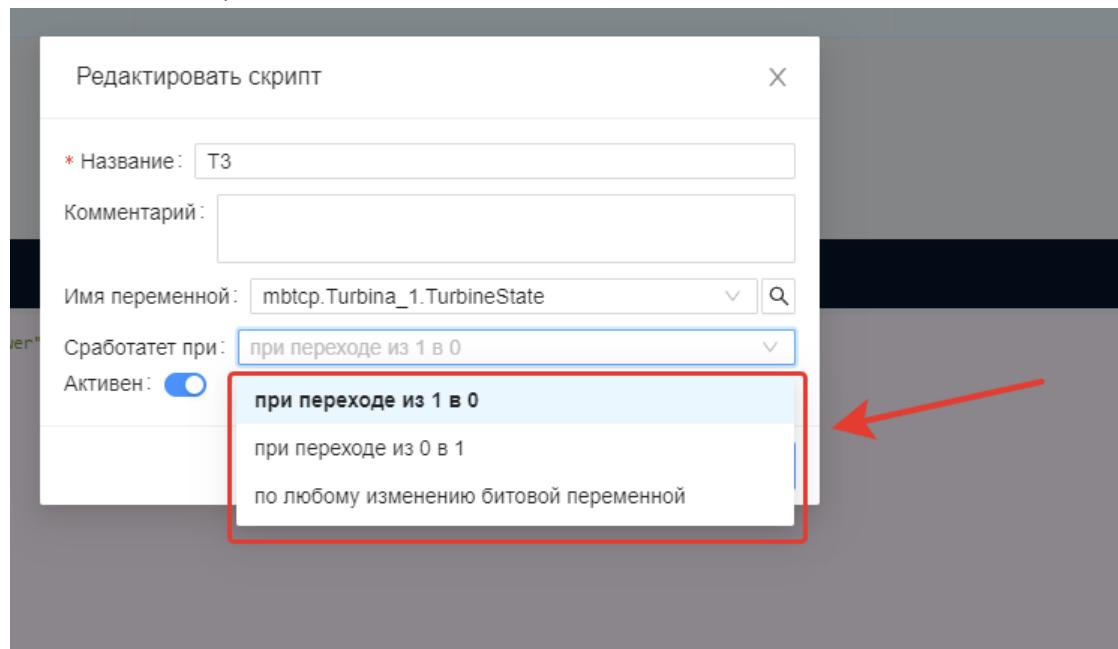
- Скрипт будет выполняться в режиме Исполнения WebDisCo, если у него в момент выполнения установлен признак **Включен**.



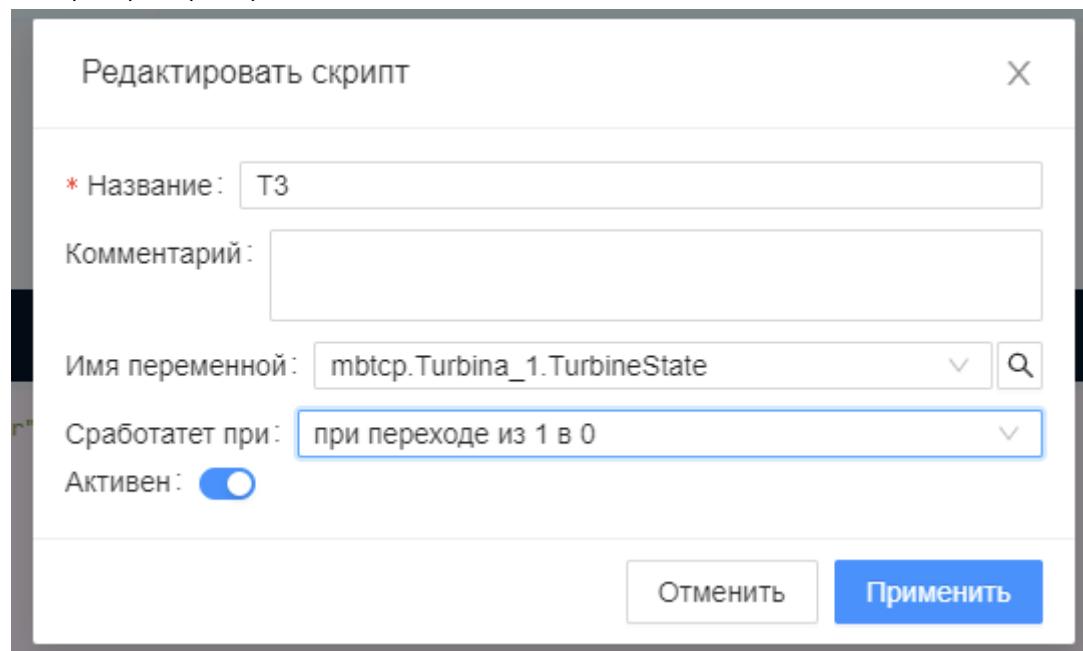
2. Автоматически по изменению битовой переменной, привязанной к какому-либо скрипту.



- Если у скрипта нет привязанной переменной, то он не будет выполняться по изменению переменной.
- Скрипт будет автоматически выполняться в режиме Исполнения WebDisCo, если у этого скрипта в момент выполнения установлен признак **Включен**.
- Для выполнения скрипта по изменению битовой переменной можно установить 3 разных варианта изменения:
  - При любом изменении
  - При изменении состояния из 0 (false) в 1 (true)
  - При изменении состояния из 1 (true) в 0 (false)



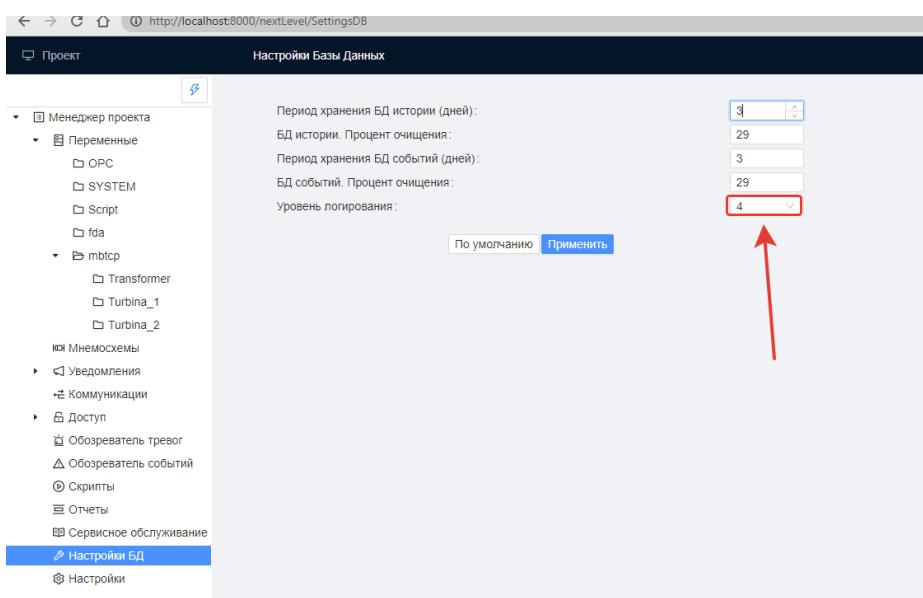
На картинке ниже скрипт ТЗ будет автоматически выполняться при изменении значения битовой переменной mbtcp.Turbina\_1.TurbineState при изменении значения переменной из 1 (true) в 0 (false).



3. Автоматически по расписанию, заданному в таблице заданий путем использования диспетчера заданий CRON. Запуск скрипта по расписанию выполняется только для включенного скрипта.

## Соглашения о выполнении скриптов в WebDisCo

1. Скрипты исполняются только в режиме Исполнения WebDisCo. Синтаксический анализ текста конкретного скрипта выполняется только в режиме Исполнения при вызове этого скрипта одним из указанных выше способов. Сообщения об этом (в том числе, об ошибках) заносятся в Обозреватель сообщений в раздел Прочее.
2. Чтобы выполнялись записи в Обозреватель сообщений различной информации о скриптах (ошибках, состоянии, включении или отключении,...) необходима установка в Настройках БД уровня логирования равным 4.



3. Если в тексте скрипта обнаружились ошибки, то скрипт не выполняется.
4. Скрипт может быть привязан только к битовой переменной. Если при выборе имени привязанной переменной для скрипта будет выбрана переменная другого типа (число или строка), то такая переменная не будет разрешена (скрипт не будет добавлен).
5. К любой битовой переменной допускается привязать не более одного скрипта. Это объясняется тем, что в случае нескольких скриптов - при изменении привязанной переменной порядок выполнения скриптов невозможно задать, он заранее не известен. Поэтому было принято решение не допускать такой возможности.
6. Необходимые варианты для определения порядка работы скрипта можно задать в тексте самого скрипта, например, с помощью специальной функции WD\_PARM или путем изменения битовой переменной, привязанной к другому скрипту.
7. Состояние скрипта (Включен или Отключен) может быть изменено как в режиме Разработки, так и в режиме Исполнения.
8. Если включенный скрипт запущен с помощью диспетчера заданий CRON, то при отключении задания выполнение скрипта прекращается. И наоборот – если задание включается, то связанный с ним скрипт будет запущен при наступлении события из расписания (если при этом скрипт включен).

**Пример:**

Здесь скрипт меняет значение переменной WebDisCo "н.бит1", к которой привязан другой скрипт.

```
10 LET B = WD_PARM  
20 if B = 0 then gosub 130  
30 FOR I = 1 TO 100  
40 let m = I  
50 wd_assign "n" "Число2" m  
60 NEXT I  
70 wd_assign "n" "бит1" 1  
80 END  
130 FOR I = 1 TO 30  
140 let m = I*3  
150 wd_assign "n" "Число2" m  
160 NEXT I  
170 END
```

9. При выполнении действия **Выполнить скрипт** функция WD\_PARM возвращает значение 0. При этом у скрипта, вызываемого этим действием, не обязательно должна быть привязанная к нему переменная.
10. Если в момент исполнения скриптов происходит переход в режим Разработка, то выполнение всех скриптов прерывается.
11. Если в момент исполнения скрипта его состояние меняется с **Включен** на **Отключен**, то его выполнение прерывается.

## Сообщения Обозревателя событий WebDisCo при использовании BASIC

При использовании BASIC в Обозреватель событий WebDisCo записываются диагностические сообщения в раздел Прочее. Например, сообщения об ошибках в скрипте BASIC.

### Список сообщений BASIC в Обозревателе событий

Текст сообщения
BASIC не выполняется в режиме Разработки
Ошибка при запуске BASIC: %s
Ошибка при выполнении программы BASIC: %s: %s
Программа BASIC запущена
Ошибка - неправильный тип переменной WebDisCo (%s.%s)
Переменная WebDisCo (%s.%s) не найдена
Ошибка - аргумент не STRING
Ошибка - типы переменных WebDisCo не совпадают
Ошибка - тип переменной WebDisCo (%s.%s) несовместим с типом NUMBER
Ошибка - тип переменной WebDisCo (%s.%s) несовместим с типом STRING
Ошибка - у переменной WebDisCo (%s.%s) нет признака Команда
Ошибка - переменная WebDisCo (%s.%s) не найдена

### Примеры:

События	Сервер	Авторизация	Прочее	Все	Избранное	Атрибут события	Приоритет события
Время события							
24.05.2022, 12:23:58.571						Сообщения от BASIC - Какая-то абракадабра	Прочее
24.05.2022, 12:17:19.846						Сообщения от BASIC - Программа BASIC запущена	Прочее
24.05.2022, 12:17:19.792						Сообщения от BASIC - Какая-то абракадабра	Прочее
24.05.2022, 12:15:27.384						Сообщения от BASIC - Ошибка при выполнении программы BASIC	line 296 Прочее
24.05.2022, 12:15:27.239						Сообщения от BASIC - WD_SEND	Ошибка - у переменной WebDisCo (mbtcp.modbus-node-2.mbt5) нет признака Команда Прочее

### Пример программы на BASIC для работы с WebDisCo

10 PRINT "HELLO, I AM EMBEDDED BASIC IN YOUR GOLANG!"

20 LET S = S + PI

60 REM

80 REM

90 FOR I = 1 TO 10

120 PRINT "I=", I

130 NEXT I

140 REM

160 REM

170 LET R = RND 30

180 IF R < 2 THEN LET R=2

190 PRINT "\tRandom value =", R, ""

191 LET N1 = WD\_READ "basic" "Число"

192 PRINT "\BASIC переменной N1 присвоено значение числовой WebDisCo переменной basic.Число = ", N1, ""

193 LET N2 = WD\_READ "basic" "Бит"

194 PRINT "\BASIC переменной N2 присвоено значение битовой WebDisCo переменной basic.Бит = ", N2, ""

195 LET N3 = WD\_READ "basic" "Строка"

196 PRINT "\BASIC Переменной N3 присвоено значение текстовой WebDisCo переменной basic.Строка= ", N3, ""

285 LET N = WD\_READ "opc.opc1" "Group1.Name101"

286 PRINT "\BASIC переменной N присвоено значение WebDisCo переменной opc.opc1.Group1.Name101= ", N, ""

287 LET M = WD\_WRITE "n" "Число2" "opc.opc1" "Group1.Name101"

288 PRINT "Значение WebDisCo переменной opc.opc1.Group1.Name101 записано в WebDisCo переменную n.Число2, значение M=", M, ""

289 LET N1 = 21

290 LET N11 = 141

294 LET M1 = wd\_assign "mbtcp.modbus-node-2" "mbt103" N11

295 PRINT "WebDisCo переменной mbtcp.modbus-node-2.mbt103 присвоено значение BASIC переменной N11=", N11, " M1=", M1, ""

296 LET M3 = wd\_send "mbtcp.modbus-node-2" "mbt103"

297 PRINT "Из WebDisCo переменной mbtcp.modbus-node-2.mbt103 отправлена команда", "M3=", M3, ""

298 LET M1 = wd\_sendb "mbtcp.modbus-node-2" "mbt5" 3.62

299 PRINT "Из WebDisCo переменной mbtcp.modbus-node-2.mbt5 отправлена команда из BASIC переменной N1=", N1, " M1=", M1, ""

302 LET M2 = WD\_ALARMCOUNT

303 PRINT "\BASIC переменной M2 присвоено значение числа тревог = ", M2, ""

304 LET M3 = wd\_userlogged

```
305 PRINT "\BASIC переменной M3 присвоено значение числа авторизовавшихся клиентов = ", M3,  
""  
  
306 LET M4 = WD_WRITELOG "Какая-то абракадабра"  
  
307 PRINT "\Сделана запись в ОС: ", M4, ""  
  
491 LET N1 = WD_READ "basic" "Число"  
  
492 LET S = 54 + N1 * 2  
  
493 LET M1 = WD_ASSIGN "basic" "Число2" S
```

В консоли сервера WebDisCo по результатам работы этой программы можно увидеть следующие сообщения:

**HELLO, I AM EMBEDDED BASIC IN YOUR GOLANG!**

```
I= 1  
I= 2  
I= 3  
I= 4  
I= 5  
I= 6  
I= 7  
I= 8  
I= 9  
I= 10 Random value = 27  
BASIC переменной N1 присвоено значение числовой WebDisCo переменной basic.Число = 33  
BASIC переменной N2 присвоено значение битовой WebDisCo переменной basic.Бит = 1  
BASIC Переменной N3 присвоено значение текстовой WebDisCo переменной basic.Строка= что-от интересное в строке  
BASIC переменной N присвоено значение WebDisCo переменной  
opc.opc1.Group1.Name101= 1
```

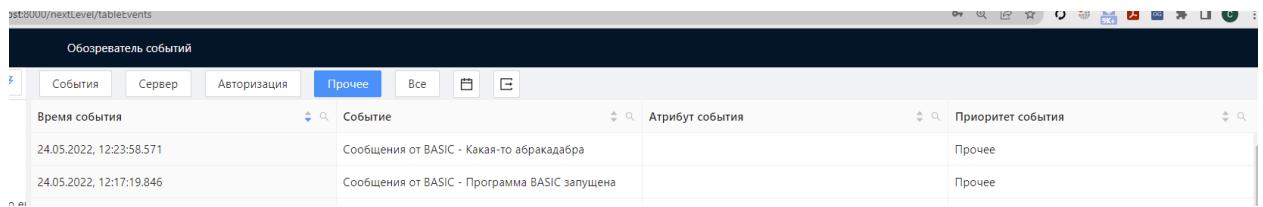
Значение WebDisCo переменной opc.opc1.Group1.Name101 записано в WebDisCo переменную n.Число2, значение M= 1

WebDisCo переменной mbtcp.modbus-node-2.mbt103 присвоено значение BASIC переменной N11= 141 M1= 141

Из WebDisCo переменной mbtcp.modbus-node-2.mbt103 отправлена команда M3= 141 sendb, val 3.62

Из WebDisCo переменной mbtcp.modbus-node-2.mbt5 отправлена команда из BASIC переменной N1= 21 M1= 3.620000  
BASIC переменной M2 присвоено значение числа тревог = 0  
BASIC переменной M3 присвоено значение числа авторизовавшихся клиентов = 1  
Сделана запись в ОС: Какая-то абракадабра

Также в Обозревателе событий будет сделана следующая запись, как результат работы специальной функции [WD\\_WRITELOG](#): LET M4 = WD\_WRITELOG "Какая-тоabra Kadabra"



The screenshot shows the Windows Event Viewer interface. The title bar reads 'Обозреватель событий'. The tabs at the top are 'События', 'Сервер', 'Авторизация', 'Прочее' (which is selected), and 'Все'. Below the tabs is a search bar. The main area is a table with four columns: 'Время события', 'Событие', 'Атрибут события', and 'Приоритет события'. There are two entries:

Время события	Событие	Атрибут события	Приоритет события
24.05.2022, 12:23:58.571	Сообщения от BASIC - Какая-тоabra Kadabra		Прочее
24.05.2022, 12:17:19.846	Сообщения от BASIC - Программа BASIC запущена		Прочее

## Список стандартных встроенных функций BASIC для работы в среде WebDisCo

Реализация языка BASIC в среде WebDisCo сделана на основе проекта на Golang: <https://github.com/skx/gobasic>. На этом сайте приведены многочисленные примеры использования функций Basic. В некоторых случаях полезно обратиться к этим примерам <https://github.com/skx/gobasic/tree/master/examples>.

### Работа с числами

Некоторые операции с числами имеют общепринятые обозначения: + (сложение), - (вычитание), \* (умножение), / (деление).

#### ABS

Абсолютная величина

**Пример:**

```
PRINT "ABS(N)=", ABS(N), "\n"
```

#### ACS

Арккосинус (результат в радианах)

**Пример:**

```
PRINT "ACS(N)=", ACS(N), "\n"
```

#### ASN

Арксинус (результат в радианах)

**Пример:**

```
PRINT "ASN(N)=", ASN(N), "\n"
```

#### ATN

Арктангенс (результат в радианах)

**Пример:**

```
PRINT "ATN(N)=", ATN(N), "\n"
```

#### BIN

Конвертирует число в двоичное представление

**Пример:**

```
400 LET A = BIN 00001111
```

```
410 LET B = BIN 11110000
```

```
420 LET C = A OR B
```

```
430 IF C = BIN 11111111 THEN PRINT "OR worked"
```

**COS**

Косинус (аргумент в радианах)

**Пример:**

```
PRINT "COS(N)=", COS(N), "\n"
```

**EXP**

Экспоненциальная функция

**Пример:**

```
PRINT "EXP(N)=", EXP(N), "\n"
```

**INT**

Целая часть

**Пример:**

```
PRINT "INT(N)=", INT(N), "\n"
```

**LN**

Натуральный логарифм

**Пример:**

```
PRINT "LN(N)=", LN(N), "\n"
```

**LOG**

Десятичный логарифм

**Пример:**

```
PRINT "LOG(N)=", LOG(N), "\n"
```

**MOD**

Вычислить значение по модулю

```
PRINT "MOD(N)=", MOD(N), "\n"
```

**Пример:**

**PI**

Число «Пи»

**Пример:**

```
PRINT "PI=", PI, "\n"
```

**POW**

Число в степени

**Пример:**

```
PRINT "POW(N)=", POW(N), "\n"
```

**RND**

Генерация случайных чисел

**Пример:**

```
LET R = RND 30
```

**SGN**

Знак числа

**Пример:**

```
PRINT "SGN(N)=", SGN(N), "\n"
```

**SIN**

Синус (аргумент в радианах)

**Пример:**

```
PRINT "SIN(N)=", SIN(N), "\n"
```

**SQR**

Квадратный корень

**Пример:**

```
PRINT "SQR(N)=", SQR(N), "\n"
```

**TAN**

Тангенс (аргумент в радианах)

**Пример:**

```
PRINT "TAN(N)=", T AN(N), "\n"
```

**Работа со строками**

**Пример:**

```
10 REM
```

```
20 REM This is a horrid script which converts the case of strings.
```

```
30 REM First upper->lower, then the reverse.
```

```
40 REM
```

```
100 LET A="STEVE IS LOWER-CASE"
```

```
110 LET L=( LEN A ) - 1
```

```
120 FOR I=0 TO L
```

```
130 LET A$ = MID$ A, I, 1
```

```
140 IF A$ >= "A" AND A$ <= "Z" THEN GOSUB 8000
150 PRINT A$
160 NEXT I
170 PRINT ""
200 LET A="steve is in upper-case, now!"
210 LET L=(LEN A) - 1
220 FOR I=0 TO L
230 LET A$ = MID$ A, I, 1
240 IF A$ >= "a" AND A$ <= "z" THEN GOSUB 9000
250 PRINT A$
260 NEXT I
270 PRINT ""
300 END
8000 REM REM
8010 REM Converts the character in A$ to lower-case
8020 REM
8030 LET A$ = CHR$ ( ( CODE A$ ) + 32 )
8045 RETURN
9000 REM
9010 REM Converts the character in A$ to upper-case
9020 REM
9030 LET A$ = CHR$ ( ( CODE A$ ) - 32 )
9040 RETURN
```

### **CHR\$**

Возвращает значение типа STRING, которое содержит символ, связанный с указанным кодом символа.

***Пример:***

```
10 REM
20 REM Prints out characters.
```

```
30 REM  
40 PRINT "This program outputs some printable ASCII characters:"  
50 FOR a=32 TO 126  
60 PRINT CHR$ a ,  
70 IF ( a % 8) = 7 THEN PRINT "";  
80 NEXT a  
90 PRINT ""
```

### CODE

Преобразует заданный символ в целочисленное значение.

**Пример:**

```
10 REM  
20 REM Prints out ASCII values  
30 REM  
40 PRINT "*** is ", CODE "*", ""  
50 PRINT " ' is ", CODE " ", ""  
100 LET A="Steve"  
110 LET L=(LEN A ) - 1  
120 FOR I=0 TO L  
130 LET X = MID$ A, I, 1  
140 PRINT "Character ", I, "is", X, "with code", CODE X, ""  
150 NEXT I
```

### LEFT\$

Возвращает крайние левые символы.

**Пример:**

```
LEFT$ "STEVE", 2
```

Возвращает два крайних левых символа слова "STEVE" ("ST").

### LEN

Возвращает длину строки.

**Пример:**

LEN "STEVE"

Возвращает длину строки «STEVE» (5).

### MID\$

MID возвращает N символов из заданного смещения.

*Пример:*

```
LastWord = MID$ ("Mid Function Demo", 14, 4)
```

Возвращает Demo

### RIGHT\$

Возвращает символы справа.

*Пример:*

```
RIGHT$ "STEVE", 2
```

Возвращает два крайних правых символа слова "STEVE" ("VE").

### SPC

Выводит строку с указанным числом пробелов.

*Пример:*

### STR\$

STR\$ возвращает представление числа в виде строки.

*Пример:*

Записать в журнал событий строки со значением числа 25

```
WD_WRITELOG STR$(25)
```

### TL\$

TL\$ возвращает строку за вычетом первого символа.

*Пример:*

### VAL

Конвертирует строку в число.

*Пример:*

```
LET N="231"
```

N присвоится значение 231.

## Логические операции

Логические операции имеют общепринятые обозначения: AND (логическое И), OR (логическое ИЛИ), OR (логическое исключающее ИЛИ).

## Операции сравнения

Операции сравнения имеют общепринятые обозначения: > (больше), >= (больше равно), < (меньше), <= (меньше равно), <> (не равно).

## Массивы

Массивы используются так же, как обычные переменные, но их необходимо объявить с помощью оператора DIM. Поддерживаются только одномерные и двумерные массивы. Доступ к отдельным элементам осуществляется с помощью смещений в скобках после имени переменной:

```
10 DIM a(10,10)
```

```
20 LET a[1,1]=10
```

```
30 PRINT a[1,1], "\n"
```

Массивы индексируются от 0 до N, поэтому при размере массива десять вы можете получить доступ к одиннадцати элементам:

```
10 DIM a(10)
```

```
20 a[0] = 0
```

```
30 a[1] = 1
```

```
40 ..
```

```
90 a[9] = 9
```

```
100 a[10] = 10
```

## Операции вывода

### PRINT

*Пример:*

```
10 REM This program demonstrates printing Ints & strings
```

```
20 PRINT "Hello, world"
```

```
30 LET a = 3
```

```
40 PRINT "The contents of the variable 'a' are", a, ""
```

```
50 LET a$ = "My name is String"
```

```
60 PRINT "The contents of the variable 'a'$ are", a$, ""
```

```
80 LET a$ = "Steve"
```

```
90 LET b = LEN a$
```

```
100 PRINT "String "" a$ "" is ", b, "characters long"
```

```
110 PRINT "'Steve' is STILL ", LEN a$, "characters long"
```

```
120 PRINT "'Steve' is STILL ", LEN( a$ ), "characters long"
```

### **READ..DATA**

DATA содержит список значений, которые последовательно назначаются с помощью команды READ, считывает значение из DATA оператора и присваивает его переменной. Внутренний указатель отслеживает последний DATA прочитанный элемент и перемещает его на одну позицию вперед с каждым READ.

***Пример:***

```
10 REM
20 REM This program prints the output of reading from DATA
30 REM
40 FOR n=1 TO 6
50 READ D
60 DATA 2,4,"Six"
70 PRINT D, ""
80 NEXT n
90 DATA 8,"Ten",12
```

## Список стандартных встроенных операторов BASIC для работы в среде WebDisCo

### REM или '

Однострочный комментарий (в BASIC нет понятия многострочных комментариев).

#### *Пример:*

```
10 REM This program demonstrates printing Ints & strings
```

```
10 ' This program demonstrates printing Ints & strings
```

### SWAP

Разрешить замену содержимого двух переменных. Полезно для сортировки массивов.

#### *Пример:*

```
330 FOR I = 1 TO 10
```

```
340 FOR J = 0 TO 9
```

```
350 LET N=J+1
```

```
360 IF A[I] < A[N] THEN SWAP A[I], A[N]
```

```
370 NEXT J
```

```
380 NEXT I
```

### LET

Присваивает значение (которое может быть результатом выражения) переменной.

#### *Пример:*

```
410 LET term=4
```

### GOTO

Переходит к пронумерованной или помеченной строке в программе.

#### *Пример:*

```
00 REM This program tests GOTO-handling.
```

```
10 GOTO 80
```

```
20 GOTO 70
```

```
30 GOTO 60
```

```
40 PRINT "Hello-GOTO!"
```

```
50 END  
60 GOTO 40  
70 GOTO 30  
80 GOTO 20
```

**FOR...TO...{STEP}...NEXT**

Цикл. Вложенные циклы FOR...NEXT разрешены:

```
FOR I = 1 TO 2
```

```
FOR J = 1 TO 2
```

```
    PRINT "I,J=",I,".",J,"\n"
```

```
NEXT J
```

```
NEXT I
```

Также внутри блока FOR...NEXT разрешены другие операторы цикла, такие как WHILE...ENDWHILE и REPEAT...UNTIL и условный оператор IF.

***Ограничения:***

- Запрещены операторы GOTO и GOSUB изнутри блока FOR...NEXT. Если такой оператор встречается внутри блока, то выдается сообщение об ошибке в Обозревателе событий.

BASIC - Ошибка при выполнении скрипта 'Проверка Whil...	(строка 60) Запрещены GOTO изнутри блоков WHILE,FO...
---	---

- Запрещены операторы GOTO и GOSUB внутрь блока FOR...NEXT. Если такой оператор выполняет переход внутрь блока FOR...NEXT, то выдается сообщение об ошибке в Обозревателе событий. Например,

BASIC - Ошибка при выполнении скрипта 'Проверка Whil...	Запрещены GOTO внутрь блоков WHILE,FOR,IF
---	---

***Пример:***

```
10 REM
```

```
20 REM This program demonstrates the use of FOR-loops.
```

```
30 REM
```

```
100 PRINT "IN ONES"
```

```
110 FOR I = 1 to 10 STEP 1
```

```
120 PRINT "\t",I,""
```

```
130 NEXT I
```

```
200 PRINT "IN TWOS"
```

```
210 FOR I = 0 to 10 STEP 2
```

```
220 PRINT "\t",I,""
```

```
230 NEXT I
```

```
300 PRINT "Backwards"
```

310 FOR I = 10 to 0 STEP -1

320 PRINT "\t",I,""

330 NEXT I

400 PRINT "With a variable"

410 LET term=4

420 FOR I = 1 TO term STEP 1

430 PRINT "\t", I, ""

440 NEXT I

500 PRINT "With an expression"

510 FOR I = 1 TO 3 \* 4 + 5

520 PRINT "\t", I, ""

530 NEXT I

### **IF...THEN...{ELSE}...ENDIF**

Условный оператор.

Оператор IF...THEN...{ELSE}...ENDIF должен быть на нескольких строках. Между THEN и ELSE, а также между ELSE и ENDIF (или THEN и ENDIF) допускается любое число операторов. Вложенные IF...THEN...{ELSE}...ENDIF не разрешены.

#### ***Ограничения:***

- Выражение IF <условие> THEN должно быть на отдельной строке. Все операторы на этой же строке после THEN будут игнорироваться
- Выражение ELSE должно быть на отдельной строке. Все операторы на этой же строке после ELSE будут игнорироваться
- Выражение ENDIF должно быть на отдельной строке. Все операторы на этой же строке после ENDIF будут игнорироваться
- При возникновении вложенного оператора Оператор IF...THEN...{ELSE}...ENDIF в Обозревателе событий будет выдаваться сообщение об ошибке, и выполнение скрипта будет прервано.
- Запрещены операторы GOTO и GOSUB изнутри блока IF...THEN...{ELSE}...ENDIF. Если такой оператор встречается внутри блока IF...THEN...{ELSE}...ENDIF, то выдается сообщение об ошибке в Обозревателе событий.
- Запрещены операторы GOTO и GOSUB внутрь блока IF...THEN...{ELSE}...ENDIF. Если такой оператор выполняет переход внутрь блока IF...THEN...{ELSE}...ENDIF, то выдается сообщение об ошибке в Обозревателе событий.
- Запрещены вложенные блоки IF...THEN...{ELSE}...ENDIF

#### ***Пример :***

**IF x > 0 THEN**

```
PRINT "OK1"  
  
PRINT "OK2"  
  
ELSE  
  
PRINT "BAD1"  
  
LET a= "BAD2"  
  
FOR I=1 TO 5  
  
PRINT "I",I, "\n"  
  
NEXT I  
  
ENDIF
```

### **WHILE <условие> DO...ENDWHILE**

Оператор WHILE <условие> DO...ENDW служит для повторения набора операций, пока заданное условие выполнено. Условие проверяется в начале цикла.

Оператор WHILE...DO...ENDWHILE должен быть на нескольких строках. Между DO и ENWHILE допускается любое число операторов. Вложенные WHILE...DO...ENDWHILE не разрешены внутри блока WHILE...DO...ENDWHILE, однако разрешены другие операторы цикла, такие как FOR...NEXT и REPEAT...UNTIL. Например, такой скрипт является правильным:

```
LET b = 1  
  
LET a = 2  
  
PRINT "b1=", b, "\n"  
  
WHILE b<14 AND a = 2 DO  
  
LET b=b+5  
  
PRINT "b2=", b, "\n"  
  
IF a = 2 THEN  
  
a = a+7  
  
ELSE  
  
a = 5  
  
ENDIF  
  
PRINT "a=", a, "\n"  
  
FOR I = 1 TO 3
```

```
PRINT "I=", I, "\n"
```

```
NEXT I
```

```
ENDWHILE
```

```
PRINT "b3=", b, "\n"
```

**Ограничения:**

- Выражение WHILE <условие> DO должно быть на отдельной строке. Все операторы на этой же строке после DO будут игнорироваться.
- Выражение ENDWHILE должно быть на отдельной строке. Все операторы на этой же строке после ENDWHILE будут игнорироваться.
- Запрещены вложенные блоки WHILE...DO...ENDWHILE внутри блока WHILE...DO...ENDWHILE. При возникновении вложенного оператора WHILE...DO...ENDWHILE в Обозревателе событий будет выдаваться сообщение об ошибке, и выполнение скрипта будет прервано.
- Запрещены операторы GOTO и GOSUB изнутри блока WHILE...DO...ENDWHILE. Если такой оператор встречается внутри блока WHILE...DO...ENDWHILE, то выдается сообщение об ошибке в Обозревателе событий:

BASIC - Ошибка при выполнении скрипта 'Проверка Whil...	Ошибка Запрещены GOTO изнутри блоков WHILE, IF
---	--

- Запрещены операторы GOTO и GOSUB внутрь блока WHILE...DO...ENDWHILE. Если такой оператор выполняет переход внутрь блока WHILE...DO...ENDWHILE, то выдается сообщение об ошибке в Обозревателе событий:

BASIC - Ошибка при выполнении скрипта 'Проверка Whil...	(строка 40) Ошибка Запрещены GOTO внутрь блоков WH...
---	---

**Пример:**

```
LET b = 1
```

```
LET a = 2
```

```
PRINT "b1=", b, "\n"
```

```
WHILE b<14 DO
```

```
    LET b=b+5
```

```
    PRINT "b2=", b, "\n"
```

```
    IF a = 2 THEN
```

```
        a = a+7
```

```
        PRINT "Тестовый вывод 1, a=", a, "\n"
```

```
    ELSE
```

```
a = 5  
  
PRINT "Тестовый вывод 2, a=",a, "\n"  
  
ENDIF  
  
PRINT "a=", a, "\n"  
  
FOR I = 1 TO 3  
  
    PRINT "I=",I,"\\n"  
  
NEXT I  
  
ENDWHILE
```

PRINT "b3=", b, "\n"

В результате в консоли будет напечатано:

```
b1=1  
b2=6  
a=9  
I=1  
I=2  
I=3  
b2=11  
a=5  
I=1  
I=2  
I=3  
b2=16  
a=5  
I=1  
I=2  
I=3  
b3=16
```

### **REPEAT... UNTIL <условие>**

Оператор REPEAT ... UNTIL используется для выполнения блока команд до тех пор, пока указанное условие не станет истинным. В отличие от других циклов, таких как FOR или WHILE, REPEAT...UNTIL гарантированно выполнится хотя бы один раз, так как проверка условия происходит после выполнения блока команд.

Оператор REPEAT...UNTIL должен быть на нескольких строках. Между REPEAT и UNTIL допускается любое число операторов. Вложенные REPEAT...UNTIL не разрешены внутри блока REPEAT...UNTIL, однако разрешены другие операторы цикла, такие как FOR...NEXT и WHILE...DO...ENDWHILE. Например, такой скрипт является правильным:

```
LET b = 1
```

```
LET a = 2
```

```
PRINT "b1=", b, "\n"
```

**REPEAT**

```
LET b=b+1
```

```
PRINT "b2=", b, "\n"
```

```
REM Вложенный FOR
```

```
FOR I = 1 TO 2
```

```
PRINT "I=",I,"\n"
```

**NEXT I**

```
REM Вложенный WHILE
```

```
LET aa = 2
```

```
LET bb = 1
```

**WHILE bb<14 DO**

```
LET bb=bb+5
```

```
PRINT "bb2=", bb, "\n"
```

```
IF aa = 2 THEN
```

```
aa = a+7
```

```
ELSE
```

```
aa = 5
```

```
ENDIF
```

```
PRINT "aa=", aa, "\n"
```

**ENDWHILE****UNTIL b>14 AND a <> 3**

```
PRINT "b3=", b, "\n"
```

***Ограничения:***

- Выражение REPEAT должно быть на отдельной строке. Все операторы на этой же строке после REPEAT будут игнорироваться.
- Выражение UNTIL <условие> должно быть на отдельной строке. Все операторы на этой же строке после <условие> будут игнорироваться.
- Запрещены вложенные блоки REPEAT...UNTIL. При возникновении вложенного оператора REPEAT...UNTIL в Обозревателе событий будет выдаваться сообщение об ошибке, и выполнение скрипта будет прервано.

- Запрещены операторы GOTO изнутри блока REPEAT...UNTIL . Если такой оператор встречается внутри блока, то выдается сообщение об ошибке в Обозревателе событий:
- Запрещены операторы GOTO внутрь блока REPEAT...UNTIL. Если такой оператор встречается внутрь блока, то выдается сообщение об ошибке в Обозревателе событий:

### Пример:

```
LET b = 1
```

```
LET a = 2
```

```
PRINT "b1=", b, "\n"
```

```
REPEAT
```

```
LET b=b+1
```

```
PRINT "b2=", b, "\n"
```

```
UNTIL b>3 AND a <> 3
```

```
PRINT "b3=", b, "\n"
```

### Пользовательские функции

#### *DEF FN <имя функции> ({аргументы})=...функция*

Оператор DEF FN задает определение пользовательской функции. DEF FN - пара ключевых слов, которые задают формулу вычисления значения. Функции BASIC реализованы на основе односторонних функций (как в FORTRAN). Пользовательские функции представляют собой одно выражение с одним или двумя аргументами с синтаксисом по образцу DEF FN(x)=x\*x или без аргументов. Функции вызываются с помощью оператора FN. Запрещены вложенные блоки DEF FN.

#### *Ограничения:*

- Выражение DEF FN <функция> должно быть на одной строке.
- Пользовательские функции анализируются во время выполнения, так как они могут быть вызваны до того, как они будут определены, например, как в следующем примере:

```
10 RINT SQUARE (3)
20 DEF FN SQUARE (a)=a*a
```

#### *FN <имя функции> ({аргументы})*

Оператор FN выполняет вызов пользовательской функции.

```
DEF FN MY1(x)=x+x
```

```
LET a = FN MY1(115)
```

```
PRINT a,"\\n"
```

Пример 1 (функция с одним аргументом)

```
10 REM
```

```
20 REM This program demonstrates the usage
```

```
30 REM of DEF FN, and FN.
```

```

40 REM

50 DEF FN double(x) = x + x

60 DEF FN square(x) = x * x

70 DEF FN cube(x) = x * x * x

80 DEF FN quad(x) = x * x * x * x

90 PRINT "N\tDoubled\tSquared\tCubed\tQuadded (?)"

100 FOR I = 1 TO 10

110 PRINT I, "\t", FN double(I), "\t", FN square(I), "\t", FN cube(I), "\t", FN quad(I), ""

120 NEXT I

```

Пример 2 (функция с двумя аргументами)

```
10 DEF FN TWOARGS(x,y)=x+y
```

```
20 PRINT "TWOARGS=",FN TWOARGS(5,2),"\n"
```

Пример 3 (функция без аргументов)

```
FN MY1()
```

```
DEF FN MY1()=PRINT "Функция без аргументов\n"
```

## Подпрограммы

### *RETURN*

Выйти из подпрограммы

### *END*

Конец программы

### *GOSUB*

Переходит к пронумерованной или помеченной строке, выполняет найденный там код до тех пор, пока не дойдет до команды RETURN, по которой он переходит обратно к оператору, следующему за GOSUB, либо после двоеточия, либо на следующей строке.

### *Ограничения:*

- Запрещены операторы GOSUB изнутри блока WHILE...DO...ENDWHILE, FOR...NEXT, IF...ENDIF. Если такой оператор встречается внутри блока, то выдается сообщение об ошибке в Обозревателе событий.

BASIC - Ошибка при выполнении скрипта 'Проверка GOS...	Запрещены GOSUB изнутри блоков WHILE,FOR,IF
--	---

- Запрещены операторы GOSUB внутрь блока WHILE...DO...ENDWHILE, FOR...NEXT, IF...ENDIF. Если такой оператор GOSUB выполняет переход внутрь блока, то выдается сообщение об ошибке в Обозревателе событий.

BASIC - Ошибка при выполнении скрипта 'Проверка GOSUB...' (строка 10) Запрещены GOSUB внутри блоков WHILE, FOR...

Пример:

120 FOR I=1 TO 3

150 PRINT "I=",I,"\n"

160 NEXT I

140 GOSUB 8000

170 PRINT "После GOSUB\n"

200 PRINT "Перед END\n"

300 END

8000 PRINT "Перешли по GOSUB\n"

8045 RETURN

В результате в консоли будет напечатано:

I=1

I=2

I=3

Перешли по GOSUB

После GOSUB

Перед END

### **Пользовательские процедуры**

#### ***DEFPROC <имя процедуры>...ENDPROC***

Оператор DEFPROC <имя процедуры>...ENDPROC задает определение пользовательской процедуры.

#### ***CALLPROC <номер строки процедуры>***

Оператор CALLPROC выполняет вызов пользовательской процедуры по имени пользовательской процедуры. Вызов пользовательской процедуры CALLPROC <имя процедуры> должен быть только после определения процедуры и выполняется по имени пользовательской процедуры.

Оператор DEFPROC <имя процедуры> ...ENDPROC должен быть на нескольких строках. Вложенные DEFPROC <имя процедуры> ...ENDPROC не разрешены, однако DEFPROC и ENDPROC допускается любое число других операторов, таких как FOR...NEXT и WHILE...DO...ENDWHILE. Например, такой скрипт является правильным:

```
PRINT "I0=","\n"
```

```
DEFPROC Test
```

```
LET b = 1
```

```
LET a = 2
```

```
PRINT "b1=", b, "\n"
```

```
REPEAT
```

```
    LET b=b+1
```

```
    PRINT "b2=", b, "\n"
```

```
REM Вложенный FOR
```

```
FOR I = 1 TO 2
```

```
    PRINT "I=", I, "\n"
```

```
NEXT I
```

```
REM Вложенный WHILE
```

```
LET aa = 2
```

```
LET bb = 1
```

```
WHILE bb<14 DO
```

```
    LET bb=bb+5
```

```
    PRINT "bb2=", bb, "\n"
```

```
    IF aa = 2 THEN
```

```
        aa = a+7
```

```
    ELSE
```

```
        aa = 5
```

```
    ENDIF
```

```
    PRINT "aa=", aa, "\n"
```

```
ENDWHILE
```

```
UNTIL b>14 AND a <> 3
```

```
40 PRINT "b3=", b, "\n"
```

```
PRINT "Перед END\n"
```

```
ENDPROC
```

```
PRINT "I1=","\n"
```

```
CALLPROC Test
```

```
PRINT "I2=","\n"
```

CALLPROC Test

PRINT "I2=","\n"

CALLPROC Test

PRINT "I3=","\n"

**Ограничения:**

- Выражение DEFPROC <имя процедуры> должно быть на отдельной строке. Все операторы на этой же строке после DEFPROC будут игнорироваться.
- <имя процедуры> после оператора DEFPROC является обязательным, так как по этому имени выполняется вызов пользовательской процедуры оператором CALLPROC <имя процедуры>.
- Выражение ENDPROC должно быть на отдельной строке. Все операторы на этой же строке после ENDPROC будут игнорироваться.
- Вызов пользовательской процедуры CALLPROC <имя процедуры> должен быть только после определения процедуры и выполняется по имени пользовательской процедуры.

PRINT "I0=","\n"

DEFPROC Test

PRINT "Процедура 1\n"

ENDPROC

PRINT "I1=","\n"

CALLPROC Test

PRINT "I2=","\n"

- Запрещены вложенные блоки DEFPROC <имя процедуры> ...ENDPROC внутри блока DEFPROC <имя процедуры> ...ENDPROC. При возникновении вложенного оператора DEFPROC <имя процедуры>...ENDPROC в Обозревателе событий будет выдаваться сообщение об ошибке, и выполнение скрипта будет прервано.
- Запрещены операторы GOTO и GOSUB изнутри блока DEFPROC <имя процедуры> ...ENDPROC. Если такой оператор встречается внутри блока DEFPROC <имя процедуры> ...ENDPROC, то выдается сообщение об ошибке в Обозревателе событий.
- Запрещены операторы GOTO и GOSUB внутри блока <номер строки> DEFPROC...ENDPROC. Если такой оператор выполняет переход внутрь блока DEFPROC <имя процедуры> ...ENDPROC, то выдается сообщение об ошибке в Обозревателе событий.

## Приложение: Комбинации клавиш редактора скриптов

Подробное описание [здесь](#).

## Стандартный набор комбинаций клавиш

Набор привязок клавиш, тесно связанных со стандартными для платформы или широко используемыми привязками.

ArrowLeft: cursorCharLeft (selectCharLeft with Shift)

ArrowRight: cursorCharRight (selectCharRight with Shift)

Ctrl-ArrowLeft (Alt-ArrowLeft on macOS): cursorGroupLeft (selectGroupLeft with Shift)

Ctrl-ArrowRight (Alt-ArrowRight on macOS): cursorGroupRight (selectGroupRight with Shift)

Cmd-ArrowLeft (on macOS): cursorLineStart (selectLineStart with Shift)

Cmd-ArrowRight (on macOS): cursorLineEnd (selectLineEnd with Shift)

ArrowUp: cursorLineUp (selectLineUp with Shift)

ArrowDown: cursorLineDown (selectLineDown with Shift)

Cmd-ArrowUp (on macOS): cursorDocStart (selectDocStart with Shift)

Cmd-ArrowDown (on macOS): cursorDocEnd (selectDocEnd with Shift)

Ctrl-ArrowUp (on macOS): cursorPageUp (selectPageUp with Shift)

Ctrl-ArrowDown (on macOS): cursorPageDown (selectPageDown with Shift)

PageUp: cursorPageUp (selectPageUp with Shift)

PageDown: cursorPageDown (selectPageDown with Shift)

Home: cursorLineBoundaryBackward (selectLineBoundaryBackward with Shift)

End: cursorLineBoundaryForward (selectLineBoundaryForward with Shift)

Ctrl-Home (Cmd-Home on macOS): cursorDocStart (selectDocStart with Shift)

Ctrl-End (Cmd-Home on macOS): cursorDocEnd (selectDocEnd with Shift)

Enter: insertNewlineAndIndent

Ctrl-a (Cmd-a on macOS): selectAll

Backspace: deleteCharBackward

Delete: deleteCharForward

Ctrl-Backspace (Alt-Backspace on macOS): deleteGroupBackward

Ctrl-Delete (Alt-Delete on macOS): deleteGroupForward

Cmd-Backspace (macOS): deleteToLineStart.

Cmd-Delete (macOS): deleteToLineEnd.

### Комбинации клавиш по умолчанию

Alt-ArrowLeft (Ctrl-ArrowLeft on macOS): cursorSyntaxLeft (selectSyntaxLeft with Shift)

Alt-ArrowRight (Ctrl-ArrowRight on macOS): cursorSyntaxRight (selectSyntaxRight with Shift)

Alt-ArrowUp: moveLineUp

Alt-ArrowDown: moveLineDown

Shift-Alt-ArrowUp: copyLineUp

Shift-Alt-ArrowDown: copyLineDown

Escape: simplifySelection

Ctrl-Enter (Comd-Enter on macOS): insertBlankLine

Alt-I (Ctrl-I on macOS): selectLine

Ctrl-i (Cmd-i on macOS): selectParentSyntax

Ctrl-[ (Cmd-[ on macOS): indentLess

Ctrl-] (Cmd-) on macOS): indentMore

Ctrl-Alt-\ (Cmd-Alt-\ on macOS): indentSelection

Shift-Ctrl-k (Shift-Cmd-k on macOS): deleteLine

Shift-Ctrl-\ (Shift-Cmd-\ on macOS): cursorMatchingBracket

Ctrl-/ (Cmd-/ on macOS): включить/отключить комментарий для выбранной строки в виде символа '

```
' LET B = WD_TODAY+"T00:00"
LET E = WD_TODAY+"T23:59"
PRINT "B=",B, " E=", E, "\n"
LET H = STR$(WD_HOUR_OF_DAY)
PRINT "HOUR=", H, "\n"
LET M = STR$(WD_MINUTE_OF_HOUR)
PRINT "MINUTE=", M, "\n"
LET S = STR$(WD_SECOND_OF_MINUTE)
```

Ctrl-/: включить/отключить комментарий для всех строк выделенного блока в виде символа '

```
LET B = WD_TODAY+"T00:00"
' LET E = WD_TODAY+"T23:59"
' PRINT "B=",B, " E=", E, "\n"
' LET H = STR$(WD_HOUR_OF_DAY)
' PRINT "HOUR=", H, "\n"
' LET M = STR$(WD_MINUTE_OF_HOUR)
' PRINT "MINUTE=", M, "\n"
' LET S = STR$(WD_SECOND_OF_MINUTE)
```

## Комбинации клавиш истории действий

Mod-z: отменить правку.

Mod-y (Mod-Shift-z on macOS) + Ctrl-Shift-z on Linux: повторить правку.

Mod-u: отменить Выбор.

Alt-u (Mod-Shift-u on macOS): повторить Выбор.