Утверждаю
Директор
ООО «Сигма-Софт
Автоматизация»
_____М.И. Мальцев
“____”_____2023 г.

Программный комплекс «С-Платформа» (S-Platform)

Руководство программиста
RU.82469608.0001-03 33
Том 3

Описание языка TScript

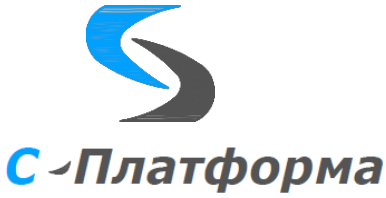
Версия 1.6

Руководитель разработки
Начальник департамента

_____И.О. Урухин
“_____”_____2023 г.

Ответственный исполнитель
Ведущий инженер-программист
_____ М.Ю. Дьяченко
“_____”_____2023 г.

ООО «Сигма Софт»
2023 г.



Утвержден

RU.82469608.0001-03 33

Программный комплекс «С-Платформа» (S-Platform)

Руководство программиста (том 3)

RU.82469608.0001-03 33

Том 3

Описание языка TScript

Версия 1.6

Листов 45

ООО «Сигма Софт»
2023 г.

АННОТАЦИЯ

В настоящем документе приведено описание языка TScript для разработки технологических алгоритмов программного комплекса «С-Платформа» (далее по тексту – ПК), описание элементов языка, поддерживаемых функций.

Данный комплекс служит для создания многоуровневых диспетчерских информационно-управляющих систем реального времени.

СОДЕРЖАНИЕ

Лист

1. ОБЩИЕ СВЕДЕНИЯ

Язык TScript разработан для целей описания технологических алгоритмов, выполняемых программным обеспечением ПК «С-Платформа».

Алгоритмы, написанные с использованием языка TScript, выполняются программным обеспечением интерпретатора языка, встроенным в сервер системы.

Пользователи могут самостоятельно реализовывать различные задачи для расчета и обработки данных и объектов ПК «С-Платформа».

2. ЭЛЕМЕНТЫ ЯЗЫКА

2.1. Общий синтаксис

2.1.1. Разделители в исходном тексте

Один или более пробелов, символов табуляции, переводов на новую строку и переводов страницы используются в языке TScript как разделители исходного текста.

2.1.2. Идентификаторы

Идентификаторы используются как имена констант, параметров, переменных и функций. Допустимыми символами для идентификаторов являются:

- цифры от 0 до 9
- латинские прописные (A-Z) и строчные (a-z) буквы
- символ подчеркивания (_)
- точка (.)

Первый символ идентификатора не может быть цифрой или точкой. Идентификаторы могут быть произвольной длины. Идентификаторы являются регистронезависимыми, то есть последовательности символов var1 и VAR1 обозначают один и тот же идентификатор. Примеры допустимых идентификаторов: Get_TI, TS.Value, Set_Flags.123, _New.Signum.

Следующие слова:
const, var, if, else, elseif, while, break, continue, exit, result
являются зарезервированными и не должны использоваться в качестве идентификаторов.

2.1.3. Типы данных

В языке TScript используются следующие типы данных:

- Integer — знаковое целое значение (8 байт)
- Double — значение с плавающей запятой (8 байт)
- String — последовательность байт произвольной длины
- Array — одномерный массив произвольной длины. Каждый элемент массива может иметь любой допустимый тип.
- Object — объект (механизм расширения)
- Empty — пустой элемент

Производных типов данных в языке TScript нет.

2.1.4. Описание констант, параметров и переменных

Прежде чем имя (идентификатор) может быть использовано в выражении, оно должно быть описано. Идентификатор может представлять собой константу, параметр функции или локальную переменную.

В языке TScript константы могут быть одного из основных типов Integer, Double и String. Значения констант подставляются в выражения на этапе компиляции функций.

Параметры и переменные представляют собой данные, подобные типу VARIANT в языке BASIC, и в

процессе вычисления могут принимать значения любого из основных типов.

Контроль совместимости типов производится на этапе вычисления. Все идентификаторы, не являющиеся константами, параметрами, переменными или именами встроенных функций, считаются внешними функциями и связываются на этапе компоновки исполняемого модуля (калькулятора).

2.1.5. Комментарии

Однострочный комментарий начинается символами // и заканчивается концом строки, в которой находятся эти символы. Многострочный комментарий начинается символами /* и заканчивается символами */. Комментарии разрешены везде, где допустимы пробелы. Примеры комментариев:

```
FUN1 (Prm1, Prm2) // Пример использования комментариев
{
var Loc1 /*перменная1*/, Loc2 /*перменная2*/, Loc3;
/* Тело
функции */
...
}
```

2.2. Операции и выражения

Список операций языка TScript приведен в Приложении А. Выражение состоит из одного или большего числа операндов и символов операций. Значение выражения может быть передано как параметр. Примеры выражений:

```
X = B + 1 + A[10, 12] + B[1][N]
Y = STR(A % 2) + "12345"
Z[2] = ((Y[1] = 100) * Z) / 2
```

2.2.1. Приоритеты и порядок выполнения операций

Приоритеты и порядок выполнения операций приведен в нижеследующей таблице. В одной и той же строке таблицы находятся операции с одинаковым приоритетом, первая строка соответствует наивысшему приоритету. Для операций с одинаковым приоритетом порядок их выполнения указан в столбце «Порядок выполнения».

Операция	*Наименование*	*Порядок выполнения*
()	Вызов функции	Слева направо
!	Логическое отрицание	3*Справа налево
~	Побитовое отрицание	
-	Изменение знака	
*	Умножение	4*Справа налево
/	Деление	
\	Целочисленное деление	
%	Деление по модулю	
+	Сложение	2*Справа налево
-	Вычитание	

<	Меньше	4*Справа налево
<=	Меньше или равно	
>	Больше	
>=	Больше или равно	
==	Равно	2*Справа налево
!=	Не равно	
&	Побитовая операция И	Справа налево
^	Побитовая операция исключающее ИЛИ	Справа налево
|	Побитовая операция ИЛИ	Справа налево
&&	Логическая операция И	Слева направо
||	Логическая операция ИЛИ	Слева направо
=	Присваивание	Справа налево
,	Операция запятая	2*Слева направо
[,]	Операция конструктор массива	

2.3. Операторы

2.3.1. Формат и вложенность

Один оператор может занимать одну или более строк. Два или большее количество операторов могут быть расположены на одной строке.

Операторы, управляющие порядком выполнения (if, while), могут быть вложены друг в друга.

2.3.2. Составной оператор

Составной оператор (блок) состоит из одного или большего числа операторов любого типа, заключенных в фигурные скобки. После закрывающей скобки не должно быть точки с запятой (;). Пример составного оператора:

```
{ X = 1; Y = 2; If (X >= Y) Z = X; else Z = Y; }
```

2.3.3. Оператор выражение

Любое выражение, заканчивающееся точкой с запятой (;), является оператором.

2.3.4. Оператор присваивания

Операция = присваивает значение выражения справа от знака операции переменной, параметру или элементу массива слева от знака операции. В выражении может быть несколько операций присваивания.

2.3.5. Оператор выхода из цикла break

Оператор break прекращает выполнение ближайшего вложенного внешнего оператора while. Управление передается оператору, следующему за заканчиваемым.

2.3.6. Оператор описания локальных констант const

Для описания локальных констант используется следующий формат:

```
const <имя1> = <значение1>[, <имя2> = <значение2>][, ...];
```

Оператор const допускается только первым оператором в составном операторе функции. Примеры описания констант:

```
const Const1 = 1;
const Const2 = 2.3, Const3 = "Строка";
```

2.3.7. Оператор продолжения цикла continue

Оператор continue передает управление в начало ближайшего внешнего оператора while. Этот оператор по действию противоположен оператору break.

2.3.8. Оператор выхода из функции exit

Оператор exit завершает выполнение текущей функции и передает управление в вызывающую. Возвращаемым значением функции является значение переменной Result на момент выхода.

2.3.9. Условный оператор if-elseif-else

Условный оператор if-elseif-else имеет следующий вид:

```
if (выражение1)
  оператор1
elseif (выражение2)
  оператор2
else
  оператор3
```

Если какое либо условное выражение истинно, то выполняется оператор, следующий за соответствующей частью if или elseif. Если все выражения ложны, то выполняется оператор, следующий за частью else. Часть else может опускаться. Частей elseif может быть произвольное количество.

2.3.10. Оператор описания локальных переменных var

Для описания локальных переменных используется следующий формат:

```
var <имя1>, <имя2>, ...;
```

Оператор var допускается только первым или вторым оператором (после оператора const) в составном операторе функции. Пример описания переменных:

```
var Local1, Local2;
```

2.3.11. Оператор цикла while

Оператор цикла while имеет следующий вид:

```
while (выражение)
  оператор
```

Если выражение истинно, то оператор выполняется до тех пор, пока выражение не станет ложным. Если выражение ложно, то управление передается следующему оператору.

2.4. Функции

2.4.1. Определение функции

Функция определяется описанием имени функции, формальных параметров и составного оператора (блока), описывающего выполняемые функцией действия.

Если в теле функции определяются константы, то первым оператором составного оператора функции должен стоять оператор const, описывающий имена и значения этих констант, разделенные запятыми. Если в теле функции используются локальные переменные (кроме Result), то следующим оператором составного оператора функции должен стоять оператор var, описывающий имена этих локальных переменных.

Пример определения функции:

```
FUNMAX(Prm1,Prm2)
{
    const Const1=1, Const2=1.234, Const3="строка";
    var Loc1, Loc2;
    // ...
    if (Prm1 >= Prm2)
      Result = Prm1;
    else
      Result = Prm2;
}
```

2.4.2. Вызов функции

Вызов функции осуществляется по ее имени. Если функция имеет параметры, то значения параметров перечисляются в круглых скобках через запятую. Количество формальных и фактических параметров должно соответствовать друг другу. Параметры передаются по значению. Каждая функция возвращает результат — значение функции.

В исполняющей системе языка TScript имеются встроенные функции, список которых приведен в Приложении Б. Названия, назначение, типы аргументов и результата большинства функций соответствуют аналогичным функциям исполняющей библиотеки языка С.

2.5. Исполняющая подсистема

Для получения результатов функций во время работы программы группы связанных функций компилируются в Р-код и объединяются вместе в так называемом «калькуляторе». Этот программный компонент позволяет хранить сгенерированный Р-код функций, обеспечивает реализацию ссылок между связанными функциями, исполняет Р-код для получения результатов.

В калькуляторе допускается использование не только функций языка и встроенных функций, но и внешних функций, разработанных пользователем. Данные функции вызываются по адресам, указанным в процессе формирования калькулятора.

В программе допускается существование множества экземпляров компонента «калькулятор».

ПРИЛОЖЕНИЕ А. ОПЕРАЦИИ ЯЗЫКА TSCRIPT

А.1 Арифметические операции

=	присваивание
+	сложение
-	вычитание
*	умножение
/	деление
\	целочисленное деление
%	деление по модулю

А.2 Операции отношения

==	равенство
!=	неравенство
>	больше
>=	больше или равно
<	меньше
<=	меньше или равно

Логическое значение ЛОЖЬ представляется целым нулевым значением, а значение ИСТИНА представляется любым ненулевым значением.

А.3 Логические операции

!	отрицание
||	логическое ИЛИ
&&	логическое И

Логическое значение ЛОЖЬ представляется целым нулевым значением, а значение ИСТИНА представляется любым ненулевым значением.

А.4 Побитовые операции

~	дополнение до единицы

&	побитовое И
|	побитовое ИЛИ
^	исключающее ИЛИ

Побитовые операции применимы только к целочисленным операндам.

A.5 Прочие операции

,	Выражения, разделенные запятой, выполняются слева направо. В качестве результата берется значение последнего выражения.
[x1, ..., xn]	Выражения, заключенные в квадратные скобки, вычисляются и из них формируется массив. Пример операции: Arr = [1, “Строка”,2.33];

ПРИЛОЖЕНИЕ Б. ВСТРОЕННЫЕ ФУНКЦИИ ЯЗЫКА TSCRIPT

Б.1 Общие функции

2.5.1. ABS(X)

Абсолютное значение X

2.5.2. IIF(X, Y1, Y2)

Если X равно ИСТИНА, то возвращает Y1, иначе Y2

2.5.3. MAX(X, Y)

Наибольшее из X и Y

2.5.4. MIN(X, Y)

Наименьшее из X и Y

2.5.5. ABORT(X)

Аварийное завершение функции с кодом возврата X

2.5.6. TRY(X)

Код завершения выполнения выражения X. Вариант try..catch блока. Если выражение X выполнено успешно, то функции возвращает значение 0.

Б.2 Функции преобразования типов

2.5.7. TYPE(X)

Получение типа результата выражения X. Возвращаемые значения: 0 — Empty, 1 — Int, 2 — Dbl, 3 — Str, 4 — IntS, 5 — DblS, 6 — StrS, 7 — Rec, 8 — Tbl, 9 — Xml, 10 — Graph)

2.5.8. INT(X)

Преобразование результата выражения X в целочисленное значение

2.5.9. DBL(X)

Преобразование результата выражения X в значение с плавающей запятой

2.5.10. STR(X)

Преобразование результата выражения X в строковое значение

2.5.11. ARR(N)

Формирование массива из N элементов

Б.3 Математические функции

2.5.12. SIN(X)

Синус X

2.5.13. COS(X)

Косинус X

2.5.14. TAN(X)

Тангенс X

2.5.15. ATAN(X)

Арктангенс X

2.5.16. EXP(X)

Экспонента X

2.5.17. LDEXP(X, E)

Экспонента (X-double, мантисса, E-integer, экспонента)

2.5.18. LOG(X)

Логарифм X

2.5.19. LOG10(X)

Десятичный логарифм X

2.5.20. SQRT (X)

Квадратный корень из X

2.5.21. POW(X, Y)

Возведение X в степень Y

2.5.22. ROUND(X, N)

Округление X до N знаков после запятой

Б.4 Функции работы со строками и массивами

2.5.23. LEN(S) => N

2.5.24. LEN(A) => N

Длина строки (без завершающего 0) или массива

2.5.25. MID(S,P,L) => S

2.5.26. MID(A,P,L) => A

Получение подстроки или подмассива (S(A)-источник, P-начальная позиция(0..N), L-длина подстроки)

2.5.27. CMP(S1,S2) => N

Сравнение строк (результат (-,0,+))

2.5.28. CMPI(S1,S2) => N

Сравнение строк без учета регистра (результат (-,0,+))

2.5.29. POS(S1,S2) => N

Поиск подстроки S2 в строке S2 (-1 не найдена)

2.5.30. LEFT(S,N) => S

2.5.31. LEFT(A,N) => A

Получение подстроки (подмассива) слева

2.5.32. RIGHT(S,N) => S

2.5.33. RIGHT(A,N) => A

Получение подстроки (подмассива) справа

2.5.34. CHR(B) => S

Преобразование кода символа в символ

2.5.35. CHR(AB) => S

Преобразование массива кодов символов в строку символов

2.5.36. ORD(S) => B

2.5.37. ORD(S) => AB

Получение кода символа или массива кодов, если строка состоит из нескольких символов

2.5.38. PRINTF(SFmt,AVal) => S

Формирование строки по шаблону и массиву значений. Правила форматирования совпадают с соглашениями, принятыми в языке C для функции printf.

2.5.39. SCANF(SBuf,SFmt) => AVal

Сканирование строки и выделение массива значений по заданному шаблону. Правила сканирования совпадают с соглашениями, принятыми в языке C для функции scanf.

Б.5 Функции работы со временем

В нижеуказанных функциях в качестве значения параметра T может использоваться как целочисленное значение времени в формате Unix timestamp, так и значение времени с плавающей запятой с миллисекундами.

2.5.40. TMMSEC(T) => N

Миллисекунды, mseconds after the seconds - [0,999]

2.5.41. TMSEC(T) => N

Секунды, seconds after the minute - [0,59]

2.5.42. TMMIN(T) => N

Минуты, minutes after the hour - [0,59]

2.5.43. TMHOUR(T) => N

Часы, hours since midnight - [0,23]

2.5.44. TMDAY(T) => N

День месяца, day of the month - [1,31]

2.5.45. TMMON(T) => N

Месяц, months since January - [0,11]

2.5.46. TMYEAR(T) => N

Год, years since 1900

2.5.47. TMISDST(T) => N

1-daylight savings time, 0-зимнее, -1-неизвестно

2.5.48. TMWDAY(T) => N

День недели, days since Sunday - [0,6]

2.5.49. TMYDAY(T) => N

День года, days since January 1 - [0,365]

2.5.50. TMLOCAL(T) => AT

Arr(Year,Mon,Day,Hour,Min,Sec,MSec,IsDst,WDay,YDay)

2.5.51. TMGM(T) => AT

Arr(Year,Mon,Day,Hour,Min,Sec,MSec,IsDst,WDay,YDay)

2.5.52. TMMAKE(Y,M,D,H,M,S,Ms) => T

Формирование времени из составляющих

2.5.53. TMMAKEA(AT) => T

Формирование времени из составляющих, расположенных в массиве [Year, Mon, Day, Hour, Min, Sec, MSec, [IsDST]]

2.5.54. TMSTR(SFmt,T) => Str

Строковое представление времени. Правила сканирования совпадают с соглашениями, принятыми в языке C для функции strftime

ПРИЛОЖЕНИЕ В. ВСТРОЕННЫЕ ОБЪЕКТЫ ЯЗЫКА TSCRIPT

B.1 IntS. Целочисленное значение с флагами

Объект IntS используется для работы с телеметрией. Целочисленному значению сопутствуют архивные флаги.
Имя типа: «IntS».
Номер типа по умолчанию: 5 (cv_Obj).

2.5.55. Функции объекта

Функция	*Результат*	*Описание*
INTS(iVal, iFlag)	Объект IntS	Конструктор объекта IntS
VAL(vIntS)	Integer	Значение
SIGN(vIntS)	Integer	Флаг значения

B.2 DblS. Значение с плавающей запятой с флагами

Объект IntS используется для работы с телеметрией. Значению с плавающей запятой сопутствуют архивные флаги.
Имя типа: «DblS».
Номер типа по умолчанию: 6 (cv_Obj+1).

2.5.56. Функции объекта

Функция	*Результат*	*Описание*
DBLS(dVal, iFlag)	Объект DblS	Конструктор объекта DblS
VAL(vDblS)	Double	Значение
SIGN(vDblS)	Integer	Флаг значения

B.3 StrS. Строковое значение с флагами

Объект StrS используется для работы с телеметрией. Строковому значению сопутствуют архивные флаги.
Имя типа: «StrS».
Номер типа по умолчанию: 7 (cv_Obj+2).

2.5.57. Функции объекта

Функция	*Результат*	*Описание*
STRS(sVal, iFlag)	Объект StrS	Конструктор объекта StrS
VAL(vStrS)	Double	Значение
SIGN(vStrS)	Integer	Флаг значения

B.4 Rec. Запись

Объект Rec представляет собой структуру, состоящую из типизированных полей. Он может использоваться как самостоятельно, так и в составе таблиц. Объект таблицы описаны в Приложении B.
Имя типа: «Rec».
Номер типа по умолчанию: 8 (cv_Obj+3).

Допустимые типы полей записи:

Тип поля	*Мнемоника*	*Описание*
Bool	B1	Логическое значение (1), 0..1
Byte	I1	Байт (1), 0..255
Int8	I1	Целое (1), -128 to 127
Small	I2	Целое (2), -32,768 to 32,767
Int32	I4	Целое (4), -2,147,483,648 to 2,147,483,647
Int64	I8	Целое (8), -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
Float	F4	Вещественное (4), -3.4E +/- 38
Double	F8	Вещественное (8), 1.7E +/- 308
Char	CHn	Строка фиксированной длины
Memo	CVn	Строка переменной длины
Blob	CB	Строка произвольной длины
UnixDT	DT	Время в UNIX-формате, mm-dd-yyyy hh:mm:ss
UnixMDT	DM	Время в UNIX-формате с миллисекундами, mm-dd-yyyy hh:mm:ss.ms
Currency	MN	Денежный тип, -922337203685477.5808..922337203685477.5807

Для регистрации нового типа записи формируется текстовая строка следующего содержания:

```
Имя типа|Количество полей N[|Описание]<Cr><Lf>
Имя поля0|Тип поля[|Описание]<Cr><Lf>
...
Имя поляN-1|Тип поля[|Описание]<Cr><Lf>
```

Имена типа и поля — произвольный текст длиной не больше 20 символов. Тип поля — мнемоника: B1, I1, I2, I4, F4, F8, CHn, CVn, CB, DT, DM, MN. Описание — необязательный текстовый комментарий. Пример строки описания типа:

```
Новый тип|3\r\nНомер\I4\r\nТекст10\CH10\r\nДанные\CB
```

2.5.58. Функции объекта

Функция	*Результат*	*Описание*

CRecTypReg(sTypeDef) ->TypeId	Integer	Регистрация нового типа записи
CRecNew(Rec|iTypeID| sTypeName)->Rec	Запись	Создание экземпляра записи
CRecCopy(Rec)-> Crec	Запись	Копирование (дублирование) записи
CrecTypName(Rec| iTypeId)->sTypeName	Строка	Получение имени типа записи
CRecTypeId(Rec| sTypeName)->iTypeId	Integer	Получение номера регистрации типа записи
CRecFlds(Rec|sTypeName| iTypeId)->iFldNum	Integer	Получение количества полей записи
CRecFld(Rec|sTypeName| iTypeId,iFldIdx|sFldName) ->{sFldName|iFldIdx, iFldType,iFldSize}	Массив	Получение характеристик поля: имя или индекс поля, тип (1..10), размер (в байтах)
CRecFldGet(Rec,iFldIdx| sFldName)->FldValue	Значение поля	Получение значения поля
CRecFldPut(Rec,iFldIdx| sFldName,FldValue)	Нет	Изменение значения поля

B.5 Tbl. Таблица

Объект Tbl представляет собой таблицу, состоящую из упорядоченных записей. Данный тип является скриптовым аналогом таблицы в оперативной памяти.
Имя типа: «Tbl».
Номер типа по умолчанию: 9 (cv_Obj+4).

2.5.59. Функции объекта

Функция	*Результат*	*Описание*
CTblNew()->CTbl	Tbl	Создание таблицы
CTblCopy(CTbl,iOpt) ->CTbl	Tbl	Копирование (дублирование) таблицы
CTblFldAdd(CTbl,sName, iType,iSize)		Добавление поля
CTblFlds(CTbl)->iFldNum	Integer	Количество полей
CTblFld(CTbl,iIdx|sName) ->{sFldName|iFldIdx, iFldType,iFldSize}	Массив	Характеристики поля
CTblKeyAdd(CTbl,sFlds,iOpt)		Добавление индекса (имена полей через «;»)
CTblKeyDel(CTbl,iIdx)		Удаление индекса
CTblKeys(CTbl)-> iCnt	Integer	Количество индексов

CTblKey(CTbl,iIdx) ->{sFlds,iOpt}	Массив	Характеристики индекса
CTblClearRec(CTbl)		Удаление всех записей
CTblClearKey(CTbl)		Удаление индексов
CTblClearAll(CTbl)		Удаление записей, полей и индексов
CTblRecNew(CTbl)->Rec	Rec	Создание новой пустой записи
CTblRecEdit(Rec)->Rec	Rec	Регистрация изменений
CTblRecPost(Rec)		Фиксация изменений
CTblRecDel(Rec)		Удаление записи
CTblRec(CTbl,iKey,iIdx) ->Rec	Rec	Получение записи по индексу
CTblRecIdx(CTbl,iKey, Rec)->iIdx	Integer	Получение индекса записи
CTblRecCmp(CTbl,iKey, Rec1,Rec2)->iCmp[-1,0,1]	Integer	Сравнение записей таблиц
CTblRecSeek(CTbl,iKey, KeyRec|KeyVal|[...]) ->[Rec|Empty, iIdx]	Массив	Поиск по шаблону или значению (значениям) ключевых полей
CTblRecSort(CTbl,iKey)		Сортировка по индексу (например, если idxNonMaintained)
CTblRecCount(CTbl) ->iCnt	Integer	Количество записей в таблице

B.6 Xml

Объект Xml представляет собой «дерево» xml-узлов, связанных между собой отношением предок-потомок. Каждый узел может иметь собственные данные и набор атрибутов.
Имя типа: «Xml».
Номер типа по умолчанию: 10 (cv_Obj+5).

2.5.60. Функции объекта

Функция	*Результат*	*Описание*

CXmlNew(iType)->CXml	Xml	Создание Xml-узла. Тип, одно из значений
		itXml_Element=0 // Узел
		itXml_Declaration=1 // <?xml version="1.0" ?>
		itXml_Comment=2 // Комментарий
		itXml_CData=3 // Секция данных
		itXml_Document=4 // Документ

CXmlType(Xml)->iType	Integer	Тип XML-узла
CXmlOwner(Xml) ->Xml|null	Xml	Получение узла документа

CXmlGetParent(Xml) ->Xml	Xml	Предок
CXmlGetName(Xml) ->sName	String	Имя XML-узла
CXmlSetName(Xml, sName)->Xml	Xml	Изменение имени XML-узла
CXmlGetVal(Xml)->sValue	String	Собственные данные узла
CXmlSetVal(Xml,sVal) ->Xml	Xml	Изменение данных
CXmlGetXML(Xml) ->sXML	String	XML-представление узла (с дочерними узлами)
CXmlSetXML(Xml, sXML) ->iErr	Integer	Изменение XML-представления (не 0 — ошибка)
CXmlErrPos(Xml)->iPos	Integer	Позиция ошибки при разборе
CXmlCount(Xml)->iCount	Integer	Количество дочерних узлов
CXmlNode(Xml,iIdx| sName)->Xml|null	Xml	Дочерний узел
CXmlFind(Xml,sName) ->Xml|null	Xml	Поиск по имени во вложенных узлах
CXmlAdd(Xml,XmlAdd) ->Add	Xml	Добавление дочернего узла
CXmlAddNew(Xml, sName,sVal)->Xml	Xml	Создание дочернего элемента
CXmlDel(Xml)->null		Удаление Xml-документа или узла
CXmlAttrCount(Xml) ->iCount	Integer	Количество атрибутов узла
CXmlAttrName(Xml,iIdx) ->sName|null	String	Имя атрибута
CXmlAttrIdx(Xml,sName) ->iIdx	Integer	Индекс атрибута в списке (начинается с 0)
CXmlAttrGet(Xml,iIdx| sName)->sValue|null	String	Значение атрибута
CXmlAttrSet(Xml,iIdx| sName,Val)->Xml	Xml	Изменение значения атрибута
CXmlAttrAdd(Xml, sName,Val)->Xml	Xml	Добавление атрибута
CXmlAttrDel(Xml,iIdx| sName)->Xml	Xml	Удаление атрибута

B.7 Graph. Граф

Объект Graph представляет собой набор взаимосвязанных узлов. Узел идентифицируется уникальным целым четырехбайтовым числом (int32). Связи между узлами не имеют направлений. С каждым узлом может быть связана группа дополнительных номеров, характеризующих содержание узла. Номера группы не являются узлами графа. Обычно группы образуются при выполнении операции стягивания графов.
Имя типа: «Graph».
Номер типа по умолчанию: 11 (cv_Obj+6).

К графам применимы следующие операции:

- A + B — объединение
- A & B — пересечение
- A - B — разность
- A * B — приращение
- A ^B — замыкание
- A / B — стягивание

2.5.60.1. Объединение

Множество вершин графа A + B есть объединение множеств вершин графов A и B. Множество ребер графа A + B есть объединение множеств ребер графов A и B.

2.5.60.2. Пересечение

Множество вершин графа A & B есть пересечение множеств вершин графов A и B. Множество ребер графа A & B есть пересечение множеств ребер графов A и B.

2.5.60.3. Разность

Множество вершин графа A - B есть разность множеств вершин графов A и B. Эта разность есть множество всех тех вершин графа A, которые не принадлежат графу B. Множество ребер графа A - B есть множество всех тех ребер графа A, которые не инцидентны ни одной вершине графа B. При удалении вершин графа B из графа A удаляется каждое ребро графа A, у которого хотя бы один конец является вершиной графа B.

2.5.60.4. Приращение

Приращение A * B графа A в графе B есть объединение графа A со всеми такими ребрами графа B, у которых хотя бы один конец принадлежит графу A. При этом каждое такое ребро рассматривается как граф, состоящий из одного этого ребра и двух вершин — концов этого ребра. Множество вершин графа A * B есть объединение множества вершин графа A и множества всех тех вершин графа B, которые соединены хотя бы одним ребром графа B с какой-нибудь вершиной графа A. Множество ребер графа A * B есть объединение множества ребер графа A и множества всех тех ребер графа B, у которых хотя бы один конец принадлежит графу A.

2.5.60.5. Замыкание

Замыкание графа A в графе B есть объединение графа A со всеми связными компонентами графа B, имеющими непустое пересечение с графом A.

2.5.60.6. Стягивание

По определению, при стягивании графа A посредством графа B (или по графу B) «стягивающее действие» на граф A оказывает только та часть графа B, которая принадлежит графу A, т.е. граф A & B. То есть, по определению операции стягивания графа A по графу B, принимается

A / B = A / (A & B)

Операцию A / B проще всего представить как последовательность стягиваний в графе A всех ребер графа A & B. Стягивания ребер могут производиться в любом порядке. Стягивание одного ребра состоит в том, что это ребро удаляется, а два его конца отождествляются. Параллельные ребра в случае их появления отбрасываются. Достаточно произвести стягивание по всем ребрам любого остова графа A & B.

2.5.61. Функции объекта

Функция	*Результат*	*Описание*
CGraphNew()	Graph	Создание графа
CGrapNodeCount(graph) ->iCount	Integer	Количество узлов в графе
CGrapNodes(graph) ->arrInt	Array	Получение массив номеров узлов графа
CGrapHasNode(graph, iNum)->iBool	Integer	Проверка наличия номера узла
CGrapAddNode(graph, iNum)->iBool	Integer	Добавление узла
CGrapDelNode(graph, iNum)->iBool	Integer	Удаление узла
CGrapRenameNode(graph, iNum,iNumNew)->iBool	Integer	Переименование (изменение номера) узла
CGrapRenameGroup(graph, iNum,iNumNew)->iBool	Integer	Переименование узла и добавление старого номера в группу
CGrapLinkCount(graph) ->iCount	Integer	Количество связей в графе
CGrapHasLink(graph, iNum1,iNum2)->iBool	Integer	Проверка наличия связи между 2 узлами
CGrapAddLink(graph, iNum1,iNum2)->iBool	Integer	Добавление связи между 2 узлами
CGrapDelLink(graph, iNum1,iNum2)->iBool	Integer	Удаление связи между 2 узлами
CGrapNodeLinkCount(graph,iNum)->iCount	Integer	Количество связей узла
CGrapNodeLinks(graph, iNum)->arrInt	Array	Получение массива номеров связанных узлов
CGrapHasGroup(graph, iNum)->iBool	Integer	Gроверка наличия гуппы узлов
CGrapGroupNodes(graph, iNum)->arrInt	Array	Получение массива номеров узлов группы
CGrapGroupHasNode(graph, iNum,iNumGr)->iBool	Integer	Проверка наличия узла в группе

CGrapGroupAddNode(graph, iNum,iNumGr)->iBool	Integer	Добавление узла в группу
CGrapGroupDelNode(graph, iNum,iNumGr)->iBool	Integer	Удаление узла из группы
CGrapBuildPath(graph, iNum1,iNum2)->graph	Graph	Получение графа кратчайшего пути между двумя узлами

ПРИЛОЖЕНИЕ Г. ВНЕШНИЕ ФУНКЦИИ ЯЗЫКА TSCRIPT

_AnsiToUtf8(Str)

Преобразование строки из кодировки ANSI в формат кодировки UTF-8

Параметры

- *Str* Строка для преобразования

Возвращаемое значение

Преобразованная строка, если *Str* не является пустой строкой
Пустая строка, если *Str* является пустой строкой или произошла ошибка преобразования

_CalcExec(Query)

Выполнение SQL-запроса

Параметры

- *Query* Строка с SQL запрсом

Возвращаемое значение

В случае ошибки функция возвращает значение *-1*.

Изменения в БД комплекса, выполненные с помощью этой функции, не фиксируются в событиях изменения БД. Также не возвращаются результаты выполнения SQL-запроса, кроме кода ошибки.

Параметры

- *Query* SQL-запрос для выполнения

Возвращаемое значение

0, если SQL-запрос выполнен успешно
-1, если выполнение SQL-запроса завершилось ошибкой

_CalcSql(Query)

Выполнение SQL-запроса SELECT

Параметры

- *Query* SQL-запрос для выполнения

Возвращаемое значение

0, если SQL-запрос выполнен успешно
-1, если выполнение SQL-запроса завершилось ошибкой

_CloseHandle(Handle)

Закрытие описателя объекта (процесса, потока, события)

Параметры

- *Handle* Целочисленный описатель объекта

Возвращаемое значение

0, если закрытие описателя выполнено успешно
-1, если закрытие описателя завершилось ошибкой

_CreateEvent(EventName)

Создание флага события

Параметры

- *EventName* Строковое имя флага события. Если *EventName* равен пустой строке, то создается флаг без имени.

Возвращаемое значение

Описатель флага (значение > 0), если флаг создан успешно
-1, если создание флага завершилось ошибкой

_CreateGuid()

Создание статического уникального 128-битного идентификатора

Возвращаемое значение

Созданный идентификатор

_CreateProcess(ComStr, DirStr, WinHide)

Создание процесса (запуск внешней программы)

Параметры

- *ComStr* Командная строка запуска процесса
- *DirStr* Рабочая папка процесса
- *WinHide* Состояние окна процесса при запуске: 0 — скрывать, не 0 — минимизировать

Возвращаемое значение

Целочисленный описатель (handle) процесса (значение > 0), если процесс создан успешно
-1, если создание процесса завершилось ошибкой

_DosToWinStr(Str)

Преобразование строки из кодировки MS-DOS (CP 866) в кодировку Windows-1251

Параметры

- *Str* Строка для преобразования

Возвращаемое значение

Преобразованная строка

_EcsPlan(Time, Id, Str)

Экстраполяция плановых данных сигнала комплекса в пределах часа

Параметры

- *Time* Время расчета в формате Unix timestamp
- *Id* Номер сигнала
- *Str* Имя таблицы архива

Возвращаемое значение

Экстраполированное значение сигнала в виде значения с плавающей запятой

_EcsPlanHalfHour(Time, Id)

Экстраполяция плановых данных сигнала комплекса в пределах получаса

Параметры

- *Time* Время расчета в формате Unix timestamp
- *Id* Номер сигнала комплекса

Возвращаемое значение

Экстраполированное значение сигнала в виде значения с плавающей запятой

_FileClose(FILE)

Закрытие файла

Параметры

- *FILE* Описатель файла

Возвращаемое значение

0, если закрытие файла выполнено успешно
-1, если закрытие файла завершилось ошибкой

_FileDelete(FilePath)

Удаление файла

Параметры

- *FilePath* Путь к удаляемому файлу

Возвращаемое значение

0, если удаление файла выполнено успешно
-1, если удаление файла завершилось ошибкой

_FileLen(FILE)

Определение размера файла

Параметры

- *FILE* Описатель файла

Возвращаемое значение

Размер файла в байтах (значение >= 0), если определение размера файла выполнено успешно
-1, если определение размера файла завершилось ошибкой

_FileOpen(FileName, TypeAcces)

Открытие файла

Параметры

- *FileName* Имя файла, включая путь

- *TypeAcces* Тип доступа к файлу. Значения этого параметра совпадают со значениями параметра mode функции fopen языка C.

Возвращаемое значение

Описатель файла, совпадающий с возвращаемым значением функции fopen языка C, если открытие файла выполнено успешно
0, если открытие файла завершилось ошибкой

_FileRead(FILE, Byte)

Чтение из файла заданного количества байт

Параметры

- *FILE* Описатель файла

- *Byte* Количество байт для чтения

Возвращаемое значение

Поледовательность прочитанных байт, если чтение выполнено успешно. Проверка того, что чтение выполнено успешно, может быть выполнена с помощью выражения TYPE(Result) == 3.
Числовое значение -1, если чтение завершилось ошибкой. Проверка того, что значение является числовым, а не строковым, может быть выполнена с помощью выражения TYPE(Result) != 3.

_FileSeek(FILE, Offset)

Установка указателя на заданную позицию в файле относительно его начала

Параметры

- *FILE* Описатель файла

- *Offset* Позиция указателя в байтах, на которую он должен быть установлен

Возвращаемое значение

0, если установка указателя выполнено успешно

-1, если установка указателя завершилось ошибкой

_FileWrite(FILE, Buff)

Запись последовательности байт в файл

Параметры

- *FILE* Описатель файла
- *Buff* Последовательности байт для записи

Возвращаемое значение

0 , если запись выполнена успешно
-1, если запись завершилась ошибкой

_GblVarDel(GblName)

Удаление глобальной переменной

Параметры

- *GblName* Имя глобальной переменной

Возвращаемое значение

0, если удаление глобальной переменной выполнено успешно
-1, если удаление глобальной переменной завершилось ошибкой

_GblVarGet(GblName)

Получение значения глобальной переменной

Параметры

- *GblName* Имя глобальной переменной

Возвращаемое значение

Значение переменной, если тип переменной не равен EMPTY
0, если тип переменной равен EMPTY

_GblVarSet(GblName, Value)

Установка значения глобальной переменной. Глобальная переменная остается доступной даже после завершения создавшей ее функции языка TScript. Если переменной с таким именем нет, то она будет создана, если есть, то ее значение будет заменено.

Параметры

- *GblName* Имя глобальной переменной
- *Value* Значение глобальной переменной. Значение может иметь любой тип значения языка TScript.

Возвращаемое значение

0, если установка значения выполнена успешно
-1, если установка значения завершилась ошибкой

_GblServiceVarSet (GblName, Sql, TblName)

Установка значения глобальной переменной для служебных НСИ. В этой переменной будит храниться результат SQL запроса, переданный во втором параметре. Глобальная переменная остается доступной даже после завершения создавшей ее функции языка TScript. Если переменной с таким именем нет, то она будет создана, если есть, то ее значение будет заменено. Если в списке таблиц, переданных в третьем параметре, присутствует таблица, значения которой меняются в процессе работы сервера приложений, то данная глобальная переменная будит автоматически обновлена, в соответствии с изменениями в таблицах.

Параметры

- *GblName* строка с именем глобальной переменной
- *Sql* строка с SQL запросом
- *TblName* строка со списком таблиц, разделенных запятой, которые должны участвовать в реконфигурации серверных таблиц НСИ

Возвращаемое значение

в случае успеха, результат SQL запроса
-1, если установка значения завершилась ошибкой

_GblServiceVarGet (GblName)

Получение значения глобальной переменной для служебных НСИ.

Параметры

- *GblName* Имя глобальной переменной

Возвращаемое значение

Значение переменной, если тип переменной не равен EMPTY
0, если тип переменной равен EMPTY

_GblServiceVarDel (GblName)

Удаление глобальной переменной для служебных НСИ.

Параметры

- *GblName* Имя глобальной переменной

Возвращаемое значение

0, если удаление глобальной переменной выполнено успешно
-1, если удаление глобальной переменной завершилось ошибкой

_NameSrv ()

Получение строки заголовка консольного окна сервера.

Параметры

- *Нет*

Возвращаемое значение

Заголовок окна сервера, в случае успеха

Пустая строка, в случае ошибки

_GetExitCodeProcess(Handle)

Определение кода завершения ранее запущенного процесса

Параметры

- *Handle* Целочисленный описатель (handle) процесса

Возвращаемое значение

Код завершения процесса

_IntgrVal(Time, Id, Str, Dim)

Суммирование значений сигнала комплекса за заданные интервалы с циклом 1 час

Параметры

- *Time* Время расчета в формате Unix timestamp
- *Id* Номер сигнала
- *Str* Имя таблицы архива
- *Dim* Массив целочисленных значений с начальными и конечными часами интервалов сумммирования в виде {<час начала интервала 1>, <час конца интервала 1>, <час начала интервала 2>, <час конца интервала 2>, ...}. Количество элементов масива должно быть кратно 2.

Возвращаемое значение

Суммированное значение сигнала в виде значения с плавающей запятой

_IntgrValHalfHour(Time, Id, Str, Dim)

Суммирование значения сигнала комплекса за заданные интервалы с циклом 30 минут

Параметры

- *Time* Время расчета в формате Unix timestamp
- *Id* Номер сигнала
- *Str* Имя таблицы архива
- *Dim* Массив целочисленных значений с начальными и конечными часами и минутами интервалов сумммирования в виде {<час начала интервала 1>, <минуты начала интервала 1>, <час конца интервала 1>, <минуты конца интервала 1>, <час начала интервала 2>, <минуты начала интервала 2>, <час конца интервала 2>, <минуты конца интервала 2>, ...}. Количество элементов масива должно быть кратно 4. Значение элементов минут округляется до ближайшего меньшего получаса.

Возвращаемое значение

Суммированное значение сигнала в виде значения с плавающей запятой

_MaxHalfHour(Time, Id, Str)

Определение часа и получаса максимума значения сигнала комплекса

Параметры

- *Time* Время расчета в формате Unix timestamp
- *Id* Номер сигнала
- *Str* Имя таблицы архива

Возвращаемое значение

Массив значений с плавающей запятой, состоящий из 2 элементов: час максимума и минуты максимума

_MaxHour(Time, Id, Str)

Определение часа максимума значения сигнала комплекса

Параметры

- *Time* Время расчета в формате Unix timestamp
- *Id* Номер сигнала
- *Str* Имя таблицы архива

Возвращаемое значение

Значение часа максимума в виде значения с плавающей запятой

_MiddlPlan(Time, Id)

Усреднение плановых данных сигнала комплекса за час по формуле (значение сигнала в предыдущем часе + значение сигнала в текущем часе) / 2

Параметры

- *Time* Время расчета в формате Unix timestamp
- *Id* Номер сигнала

Возвращаемое значение

Среднее значение в виде значения с плавающей запятой

_MiddlPlanHalfHour(Time, Id)

Расчет средних получасовых значений диспетчерского графика из часовых значений сигнала комплекса

Параметры

- *Time* Время расчета в формате Unix timestamp
- *Id* Номер сигнала

Возвращаемое значение

Среднее значение в виде значения с плавающей запятой

_MiddlPlanHalfHourNext(Time, Id)

Усреднение плановых данных ведомости за получас

Параметры

- *Time* Время расчета в формате Unix timestamp
- *Id* Номер сигнала комплекса

Возвращаемое значение

Среднее значение в виде значения с плавающей запятой

_MiddlPlanNext(Time, Id)

Усреднение плановых данных сигнала комплекса за час по формуле (значение сигнала в текущем часе + значение сигнала в следующем часе) / 2

Параметры

- *Time* Время расчета в формате Unix timestamp
- *Id* Номер сигнала

Возвращаемое значение

Среднее значение в виде значения с плавающей запятой

_MiddlVal(Time, Id, CyclCalc)

Расчет среднего значения величины за заданные интервалы

Параметры

- *Time* Время расчета в формате Unix timestamp
- *Id* Номер сигнала
- *CyclCalc* интервал для интегрирования (в минутах)

Возвращаемое значение

Среднее значение в виде значения с плавающей запятой

_MiddlValCycle (Time, Id, Array, iCycle)

Расчет среднего значения сигнала среднего значения величины за заданные интервалы с заданным циклом. Все недостоверные данные игнорируются.

Параметры

- *Time* Время расчета в формате Unix timestamp
- *Id* Номер сигнала
- *Array* Массив целочисленных значений с начальными и конечными часами и минутами интервалов усреднения в виде {<час начала интервала 1>, <минуты начала интервала 1>, <час конца интервала 1>, <минуты конца интервала 1>, <час начала интервала 2>, <минуты начала интервала 2>, <час конца интервала 2>,

<минуты конца интервала 2>, ...}. Количество элементов масива должно быть кратно 4. Значение элементов минут округляется до ближайшего меньшего получаса.

- *iCycle*. цикл интегрирования (минуты)

Возвращаемое значение

Среднее значение в виде значения с плавающей запятой

_IntegrVal (Time, Id, Cycle)

П/программа подсчета интегрального значения величины за интервал, равный цикличности архива (архив по срезам).

Параметры

- *Time* Время расчета в формате Unix timestamp
- *Id* идентификатор данного
- *Cycle* цикличность расчета (в минутах)

Возвращаемое значение

Интегральное значение величины данного, в виде значения с плавающей запятой и флагами.

В случае ошибки 0.0 с флагом недостоверности.

_QuanDayMon(Time)

Определение количества дней в месяце

Параметры

- *Time* Время расчета в формате Unix timestamp

Возвращаемое значение

Количество дней в месяце

_ReadData(Id, Time, Type)

Чтение данных сигнала комплекса из любых архивов

Параметры

- *Id* Номер сигнала
- *Time* Время расчета в формате Unix timestamp, с миллисекундами
- *Type* Выбор возвращаемых данных: 0 — значение сигнала, 1 — метка времени верхнего уровня значения сигнала, 2 — метка времени нижнего уровня значения сигнала

Возвращаемое значение

Значение в соответствии с параметром *Type*

В случае ошибки 0.0 с признаком недостоверности, или код ошибки

_ReadDataInterval (Id, Time, Time2)

Чтение данных сигнала комплекса из архивов, за интервал времени

Параметры

- *Id* Номер сигнала
- *Time* Время начала расчета в формате Unix timestamp, с миллисекундами
- *Time2* время конца расчета в формате Unix timestamp, с миллисекундами

Возвращаемое значение

Таблица, состоящая из 5 следующих полей: "ID" (mdr_TInt32), "DT" (mdr_TDouble), "VAL" (Тип архива), "FLG" (mdr_TInt32) и "DTCP" (mdr_TDouble). Таблица заполнена полученными из архива значениями сигнала.

_ReadDataTime(Id, Time)

Чтение метки времени значения сигнала из архива. Эта функция равнозначна функции _ReadData со значением параметра bTime, равным 1.

Параметры

- *Id* Номер сигнала
- *Time* Время расчета в формате Unix timestamp

Возвращаемое значение

Метка времени значения сигнала

_SetEvent(Handle)

Установка флага события

Параметры

- *Handle* Описатель флага события

Возвращаемое значение

0, если установка флага события выполнена успешно
-1, если установка флага события завершилась ошибкой

_Sleep(Time)

Приостановка выполнения (задержка)

Параметры

- *Time* Целочисленное время приостановки в миллисекундах

Возвращаемое значение

Нет

_Template(Str)

Получение объекта таблицы по имени таблицы

Параметры

- *Str* Имя таблицы

Возвращаемое значение

Объект таблицы (значение >= 0), если получение объекта выполнено успешно
-1, если получение объекта завершилось ошибкой

_TerminateProcess(Handle)

Принудительное завершение процесса

Параметры

- *Handle* Описатель процесса

Возвращаемое значение

0, если завершение процесса выполнено успешно
-1, если завершение процесса завершилось ошибкой

_TerminateThread(Handle)

Принудительное завершение потока

Параметры

- *Handle* Описатель потока

Возвращаемое значение

0, если завершение потока выполнено успешно
-1, если завершение потока завершилось ошибкой

_UnixTime()

Получение текущего времени сервера в формате Unix timestamp (количество секунд с 00:00:00 01.01.1970 UTC)

Возвращаемое значение

Время в формате Unix timestamp (без миллисекунд)

_Update(Tbl, Obj, bEvn)

Функция удаления, создания и редактирования записей в таблице БД НСИ

Параметры

- *Tbl* Имя таблицы
- *Obj* Объект таблицы
- *bEvn* Включение и выключние создания соответствующего события изменения таблицы: пустая

строка — создать событие, NONCIEVN — не создавать событие

Возвращаемое значение

0, если изменение таблицы выполнено успешно
-1, если изменение таблицы завершилось ошибкой

_Utf8ToAnsi(Str)

Преобразование строки из формата кодировки UTF-8 в кодировку ANSI

Параметры

- *Str* Строка для преобразования

Возвращаемое значение

Преобразованная строка, если *Str* не является пустой строкой
Пустая строка, если *Str* является пустой строкой или произошла ошибка преобразования

_ValHour(Time, Hour, Id)

Чтение данных сигнала комплекса из любого архива за заданный час

Параметры

- *Time* Время расчета в формате Unix timestamp
- *Hour* Целочисленное значение часа, за который выполняется чтение
- *Id* Номер сигнала

Возвращаемое значение

Значение сигнала

_ValRead(Time, Hour, Min, Sec, Id)

Чтение данных сигнала комплекса за заданное время в пределах суток

Параметры

- *Time* Время расчета в формате Unix timestamp
- *Hour, Min, Sec* Целочисленные значения часа, минуты и секунды, за которые выполняется чтение
- *Id* Номер сигнала

Возвращаемое значение

Значение сигнала

_WaitForMultipleObjects(HandleName, TimeOut)

Ожидание срабатывания одного из синхронизирующих объектов (событие, процесс, поток)

Параметры

- *HandleName* Массив описателей синхронизирующих объектов, срабатывание одного из которых

ожидается

- *TimeOut* Целочисленное время ожидания срабатывания в миллисекундах

Возвращаемое значение

Индекс сработавшего объекта (значение > 0), если сработал один из синхронизирующих объектов
255, если превышено время ожидания
-1, если выполнение синхронизации завершилось ошибкой

_WaitForSingleObject(Handle, TimeOut)

Ожидание срабатывания синхронизирующего объекта (событие, процесс, поток)

Параметры

- *HandleName* Описатель синхронизирующего объекта, срабатывание которого ожидается

- *TimeOut* Целочисленное время ожидания срабатывания в миллисекундах

Возвращаемое значение

0, если сработал синхронизирующий объект
255, если превышено время ожидания
-1, если выполнение синхронизации завершилось ошибкой

_WinToDosStr(Str)

Преобразование строки из кодировки Windows-1251 в кодировку MS-DOS (CP 866)

Параметры

- *Str* Строка для преобразования

Возвращаемое значение

Преобразованная строка

_WriteData(Id, Time, Val)

Запись данных сигнала комплекса в любые архивы

Параметры

- *Id* Номер сигнала

- *Time* Время расчета в формате Unix timestamp

- *Val* Записываемое значение одного из следующих типов: целое значение, значение с плавающей запятой, массив целых значений, массив значений с плавающей запятой

Возвращаемое значение

Нет

_WriteLogFile(Str, Name)

Запись сообщения в лог-файл с произвольным именем

Параметры

- *Str* Строка для записи в лог-файл
- *Name* Имя лог-файла

Возвращаемое значение

Нет

_WriteSrvLogFile(Str)

Запись сообщения в лог-файл сервера

Параметры

- *Str* Строка для записи в лог-файл

Возвращаемое значение

Нет

_GetCliSessId ()

Функция возвращает индекс текущей клиентской сессии, в контексте которой произошел данный вызов.

Параметры

Возвращаемое значение

Целочисленный индекс клиентской сессии.

_GetAllSessId ()

Функция возвращает массив индексов клиентских сессий, открытых в сервера.

Параметры

Возвращаемое значение

Целочисленный массив индексов клиентских сессий.

_GetUsrInfoByUsrId (sesId)

По идентификатору сессии возвращает информацию о пользователеа.

Параметры

- *sesId* целочисленный идентификатор сессии пользователя

Возвращаемое значение

Строка, состоящая из двух, разделенными запятой, частями с логином пользователя и его фамилией, именем и отчеством.

_qDebug (Str)

Выводит сообщение на консоль сервера (но не в канал Telnet).

Параметры

- *Str* строка для вывода. Выводится в формате: "Калькулятор> *Str* "

Возвращаемое значение

Нет.

_ConMessage (Str, Type)

Вывод сообщение с заданным типом на консоль сервера и в канал Telnet.

Параметры

- *Str* строка для вывода. Выводится в формате: "Калькулятор> *Str* "

- *Type* целочисленный тип сообщения: 0 – отладочное сообщение, 1 – предупредительное сообщение, 2 – сообщение об ошибке, 3 – без типа, 4 – информационное сообщение, 5 – сообщение выводимое только в случае включенного отладочного режима сервера приложений

Возвращаемое значение

Нет.

_EventNew (Code, time, dtcp, obj, capt, comment, data)

Генерирует новое событие сервера и записывает его в БД событий.

Параметры

- *Code* целочисленный код события

- *time* время верхнего уровня события в формате Unix timestamp

- *dtcp* время нижнего уровня события в формате Unix timestamp

- *obj* строковое значение поля obj события

- *capt* строковое значение поля capt события

- *comment* строковое значение поля comment события

- *data* строковое значение поля data события

Возвращаемое значение

В случае успешной генерации события - 0.

Код ошибки в случае неудачи.

_EventKwit (KeyLink, Code, time)

Квитирует событие в БД событий сервера приложений.

Параметры

- *Code* целочисленный код события

- *KeyLink* строковый уникальный идентификатор события

- *time* время квитации события в формате Unix timestamp, если передан 0 – берется текущее время сервера.

Возвращаемое значение

В случае успешной генерации события - 0.

Код ошибки в случае неудачи.

_ManualInput (Id, Flag, Value)

Постановка или снятие измерения с ручного ввода.

Параметры

- Id целочисленный идентификатор измерения

- Flag целочисленный флаг установки или снятия с ручного ввода(0-снятие, 1-установка блокирующего ручника, 2-установка НЕ блокирующего ручника)

- Value значение ручного ввода, тип значения зависит от типа измерения.

Возвращаемое значение

В случае успеха возвращает 0.

-1 в случае ошибки.

_GetArchCycle (Id)

Определение цикличности архива (в секундах) по идентификатору измерения.

Параметры

- Id целочисленный идентификатор измерения

Возвращаемое значение

В случае успеха возвращает:

> 0 - цикличность архива в секундах;

0 - ошибка;

-1 - цикличность год;

-2 - цикличность квартал;

-3 - цикличность месяц;

-4 - цикличность неделя;

-5 - цикличность год;

-1 в случае ошибки.

ПЕРЕЧЕНЬ ПРИНЯТЫХ СОКРАЩЕНИЙ

БД - база данных

ОС - операционная система

ОЗУ - оперативное запоминающее устройство

ЛИСТ РЕГИСТРАЦИИ ИЗМЕНЕНИЙ									
Изм	Номера листов (страниц)				Всего листов (страниц) в докум.	№ документа	Входящий № сопровод-ительного докум. и дата	Подп.	Дата
	изменен-ных	заменен-ных	новых	аннулиро-ванных					