ООО "МНПП "САТУРН"

Управляющая программа домового регистратора

Встроенный скриптовой язык обработки данных

Руководство программиста

Версия документа:	*190*
Дата последних изменений:	*05.04.2023 13:41*
Количество страниц:	*79*

Москва 2008

Оглавление

АННОТАЦИЯ....7

1. НАЗНАЧЕНИЕ....7

1.1 Применение....7
1.2 Ограничения....7

2. ПОДКЛЮЧЕНИЕ СКРИПТА....7

3. ОСНОВНЫЕ ВОЗМОЖНОСТИ....8

3.1 Реализованные возможности....8
3.2 Нереализованные возможности....8
3.3 Особенности....8
3.3.1 Общие правила....8
3.3.2 Особенности «C++Script»....8
3.4 Оценка быстродействия....9

4. СИНТАКСИС СКРИПТОВОГО ЯЗЫКА....9

4.1 Синтаксис программы на языке «PascalScript»....9
4.2 Синтаксис программы на языке «C++Script»....11
4.3 Синтаксис программы на языке «BasicScript»....12

5. СТРУКТУРА ПРОГРАММЫ....14

5.1 Структура программы на языке «PascalScript»....14
5.1.1 Состав программы....14
5.1.2 Пример....14
5.2 Структура программы на языке «C++Script»....14
5.2.1 Состав программы....14
5.2.2 Пример....14
5.3 Структура программы на языке «BasicScript»....15
5.3.1 Состав программы....15
5.3.2 Пример....15

6. ТИПЫ ДАННЫХ....15

6.1 Целочисленные....15
6.2 Числа с плавающей точкой....16
6.3 Логический....16
6.4 Символьный....16
6.5 Прочие....16
6.6 Совместимость типов....17
6.7 Особенности применения типов в «BasicScript»....17

7. МАССИВЫ....17

8. КЛАССЫ....17

8.1 Стандартные классы....18
8.2 Классы доступа к данным....18
8.2.1 Класс TDevice....18
8.2.1.1 Описание класса....18
8.2.1.2 Свойство DeviceID....19
8.2.1.3 Свойство ProtocolID....19
8.2.1.4 Свойство SOSVersion....19
8.2.1.5 Свойство LastExchange....20
8.2.1.6 Свойство State....20
8.2.1.7 Свойство SOSExchangeCount....20
8.2.1.8 Свойство SOSNotAnswerCount....20
8.2.1.9 Свойство Period....20
8.2.1.10 Свойство SOSVcc....20
8.2.1.11 Свойство Comment....20
8.2.1.12 Метод ImmediatelyDoIO....21
8.2.2 Класс TChannel....21
8.2.2.1 Описание класса....21
8.2.2.2 Свойства A1, A2, A3, A4....22
8.2.2.3 Свойство Name....22
8.2.2.4 Свойство DType....22
8.2.2.5 Свойство State....22
8.2.2.6 Свойство SendCount....23
8.2.2.7 Свойство ExpressionOk....23
8.2.2.8 Свойство Expression....23
8.2.2.9 Свойство Time....23
8.2.2.10 Свойство DataChange....23
8.2.2.10 Свойство DataReady....24
8.2.2.10 Свойство Value....24
8.2.2.11 Свойство ValCount....24
8.2.2.12 Индексное свойство Values....24
8.2.2.13 Свойство Control....25
8.2.2.14 Свойство Owner....26
8.2.2.15 Метод AddrIs....26
8.2.3 Класс TDeviceList....26
8.2.3.1 Описание класса....26
8.2.3.2 Свойство Count....27
8.2.3.3 Индексное свойство Items....27
8.2.3.4 Метод CountOnState....27
8.2.3.5 Метод AverageQuality....28
8.2.4 Класс TChannelList....28
8.2.4.1 Описание класса....28
8.2.4.2 Свойство Count....29
8.2.4.3 Индексное свойство Items....29
8.2.4.4 Метод CountOnState....29
8.2.4.5 Метод SendAll....29
8.2.4.6 Метод Get....29
8.2.4.7 Метод Get2....30
8.2.5 Класс TChannel2....30
8.2.5.1 Описание класса....30
8.2.5.2 Свойства Addr, ID....31
8.2.5.3 Свойство Active....31
8.2.5.4 Свойство Changed....31
8.2.5.5 Свойство Type....32
8.2.5.6 Свойство TypeText....32
8.2.5.7 Свойство Time, DT....32
8.2.5.8 Свойство SaveServerValue....32
8.2.5.9 Свойство Quality, State....32
8.2.5.10 Свойство Value....32
8.2.5.11 Свойство ValueText....33
8.2.5.12 Свойство Control....33
8.2.5.13 Свойство WriteEnable....33

8.2.5.14 Метод AddrIs....33
8.2.6 Класс TChannel2List....34
8.2.6.1 Описание класса....34
8.2.6.2 Свойство Count....34
8.2.6.3 Индексное свойство Items....34
8.2.6.4 Метод Get....35
8.2.7 Класс TLocalStorage....35
8.2.7.1 Описание класса....35
8.2.7.2 Метод SetItem....36
8.2.7.3 Метод GetItem....36
8.2.7.4 Метод AsInt....37
8.2.7.5 Метод AsDouble....37
8.2.7.6 Метод AsBool....37
8.2.7.7 Метод HasItem....37
8.2.7.8 Метод RemoveItem....38
8.2.7.9 Метод Clear....38

9. ВСТРОЕННЫЕ ПРОЦЕДУРЫ И ФУНКЦИИ....38

9.1 Преобразование типов....38
9.1.1 IntToStr....38
9.1.2 FloatToStr....38
9.1.3 DateToStr....38
9.1.4 TimeToStr....39
9.1.5 DateTimeToStr....39
9.1.6 VarToStr....39
9.1.7 StrToInt....39
9.1.8 StrToFloat....40
9.1.9 StrToDate....40
9.1.10 StrToTime....40
9.1.11 StrToDateTime....40
9.2 Форматирование....41
9.2.1 Format....41
9.2.2 FormatFloat....41
9.2.3 FormatDateTime....42
9.3 Работа с датой и временем....43
9.3.1 EncodeDate....43
9.3.2 EncodeTime....43
9.3.3 DecodeDate....43
9.3.4 DecodeTime....44
9.3.5 Date....44
9.3.6 Time....44
9.3.6 Now....44
9.3.7 DayOfWeek....44
9.3.8 IsLeapYear....45
9.3.9 DaysInMonth....45
9.3.10 GetTickCount....45
9.3.11 TickDelta....45
9.4 Строковые функции....46
9.4.1 Length....46
9.4.2 Copy....46
9.4.3 Pos....46

9.4.4 Delete....46
9.4.5 Insert....47
9.4.6 UpperCase....47
9.4.7 LowerCase....47
9.4.8 Trim....47
9.4.9 NameCase....47
9.4.10 CompareText....48
9.4.11 Chr....48
9.4.12 Ord....48
9.4.13 SetLength....48
9.5 Математические функции....48
9.5.1 Round....48
9.5.2 Trunc....49
9.5.3 Int....49
9.5.4 Frac....49
9.5.5 Sqrt....49
9.5.6 Abs....50
9.5.7 Sin....50
9.5.8 Cos....50
9.5.9 Tan....50
9.5.10 ArcTan....50
9.5.11 Exp....51
9.5.12 Ln....51
9.5.13 Pi....51
9.6 Функции управления таймером....51
9.6.1 SetTimer....51
9.7 Запись в системный журнал....53
9.7.1 AddLog....53
9.7.2 Writeln....53
9.7.3 ShowMessage....53
9.8 Функции работы с каналами....54
9.8.1 AddChannel2....54
9.8.2 cq....55
9.8.3 cv....56
9.8.4 cvDef....56
9.8.5 SendControl....57
9.9 Прочие....57
9.9.1 IIF....57
9.9.2 EnableException....58
9.9.3 Inc....58
9.9.4 Dec....59
9.9.5 RaiseException....59
9.9.6 Randomize....59
9.9.7 Random....59
9.9.8 ValidInt....60
9.9.9 ValidFloat....60
9.9.10 ValidDate....60
9.9.11 LMServerStatus....60

9.9.12 TimeOfSunrise.....61
9.9.13 TimeOfSunset.....61
9.9.14 Min.....61
9.9.15 Max.....62

10. ПРЕДОПРЕДЕЛЕННЫЕ КОНСТАНТЫ.....62

10.1 Работа с журналом SYSLOG.....62
10.2 Состояние данных.....62
10.3 Тип данных каналов тип 1.....63
10.4 Время.....64
10.5 Операции с файлами.....64
10.6 Конфигурация системы.....64
10.7 Тип данных каналов тип 2.....65

11. ПРЕДОПРЕДЕЛЕННЫЕ ПЕРЕМЕННЫЕ.....65

12. СОБЫТИЯ.....65

12.1 Введение.....65
12.2 Запуск программы.....65
12.3 Окончание работы программы.....66
12.4 Изменение состояния канала тип 1.....66
12.5 Изменение даты и времени.....67
12.6 Событие таймера.....67
12.7 Получение команды управления для канала тип 2.....68
12.8 Изменение состояния канала тип 2.....69

13. ПРИМЕРЫ ПРИМЕНЕНИЯ.....70

13.1 Управление системой вентиляции.....70
13.1.1 Описание.....70
13.1.2 Схема подключения оборудования.....70
13.1.3 Пример конфигурации.....71
13.2 Управление светофором.....72
13.2.1 Описание.....72
13.2.2 Пример конфигурации.....72
13.3 Управление насосной станцией.....73
13.3.1 Описание.....73
13.3.2 Пример конфигурации.....74
13.4 Изменение настроек регулятора CAT-500 в зависимости от расхода теплоносителя (каналы тип 2).....75
13.4.1 Описание.....75
13.4.2 Пример конфигурации.....76
13.5 Автоматический терморегулятор (каналы тип 2).....77
13.5.1 Описание.....77
13.5.2 Пример конфигурации.....77

Аннотация

Данный документ является описанием встроенных программных ресурсов управляющей программы домового регистратора: синтаксиса языков программирования, объектов и функций. Он предназначен изучения программистами достаточной степени квалификации, владеющими одним из языков: «ObjectPascal», объектный бэйсик, или «C++». Этот документ не является руководством по изучению программирования. Для изучения программирования на одном из вышеназванных языков используйте соответствующую литературу.

1. Назначение

1.1 Применение

Встроенный скриптовой язык программирования предназначен для реализации автоматических алгоритмов управления оборудованием. Программа на скрипте является частью конфигурации управляющей программы домового регистратора настройка и управление которой описано в документе «Описание OPROS.PDF».

Областью возможного применения является использование в качестве языка управления программируемым контроллером в системах коммерческого учета потребления энергоресурсов, управления доступом, регулирования климата, вентиляции и кондиционирования, управления электроснабжением, производственным оборудованием и прочими системами автоматики различных отраслей народного хозяйства.

Системы построенные на основе контроллеров работающих под управлением управляющей программы с использованием скриптового языка нельзя отнести к системам «жесткого» реального времени.

1.2 Ограничения

Система функционирования управляющей программы не гарантирует четкого соблюдения временных циклов, времени отклика на событие и времени выполнения управляющего воздействия, которые могут зависеть от качества обмена данными с оборудованием, типа и количества опрашиваемых устройств, функционирования внешних программно-аппаратных средств. Тем не менее, в управляющей программе приняты все меры для минимизации фактора случайности временных интервалов. Программное обеспечение предназначено для построение систем «мягкого» реального времени, режимы работы которых допускают некоторое нарушение временных ограничений которые, в свою очередь, не приводят к серьезным негативным последствиям.

2. Подключение скрипта

Программа на на скриптовом языке является частью конфигурации управляющей программы домового регистратора. Текст скриптовой программы является частью файла конфигурации списка опрашиваемых устройств (device.ini). Исходный текст скрипта начинается с указания секции [CODE] файла конфигурации. Подробное описание смотри в пункте 4.5.1.2 документа «Описание OPROS.PDF».

3. Основные возможности

3.1 Реализованные возможности

К реализованным возможностям скриптового языка относятся:

- многоязычная структура скрипта, позволяющая использовать при написании программ три языка программирования: «PascalScript», «C++Script», «BasicScript»;
- относительно высокая скорость исполнения: при вызове кода на скрипте выполняется предварительно скомпилированный из исходного текста «псевдокод», компиляция выполняется на этапе загрузки и инициализации управляющей программы;
- стандартный языковой набор: переменные, константы, процедуры и функции (с возможностью вложенности) с переменными/постоянными/умалчиваемыми параметрами;
- все стандартные операторы и объявления (включая case, try/finally/except, with), циклы (for, while, repeat), предопределенные типы (целый, дробный, логический, символьный, строковый, многомерные массивы, variant);
- классы (с методами, событиями, свойствами, индексами и свойствами по умолчанию);
- проверка совместимости типов;
- доступ к предопределенным стандартным классам;
- доступ к свойствам и методам основных объектов управляющей программы.

3.2 Нереализованные возможности

К нереализованным возможностям относятся:

- отсутствуют возможность объявления новых типов, записей и классов;
- нет указателей (pointers) и множеств (sets) (однако возможно использование оператора 'IN' - "a in ['a'..'c','d']");
- нет типа «ShortStrings», все строки относятся к типу «AnsiString»;
- нет оператора безусловного перехода (GOTO);
- в «C++Script» нет восьмеричных констант.

3.3 Особенности

3.3.1 Общие правила

При написании скриптов обработки данных следует обращать внимание на следующее:

- выполнение скриптов производится в общем контексте потока опроса оборудования;
- необходимо по возможности минимизировать время выполнения процедур, функций и обработчиков событий, избегать больших циклов и вызова блокирующих функций;
- необходимо тщательно анализировать написанный код с целью избежания зацикливания так как это приведет к остановке функционирования основной управляющей программы;
- управляющая программа не содержит средств ограничения времени функционирования скрипта.

3.3.2 Особенности «C++Script»

При написании скриптов на языке «C++Script» существуют следующие особенности:

- нет оператора «break» в конструкции «switch», который работает подобно конструкции «case» языка «Pascal»;

- операторы «++» и «--» возможны только после переменных ("--i" работать не будет);
- операторы «++» и «--» не возвращают значения ("if(i++)" работать не будет);
- все идентификаторы не чувствительны к регистру;
- вместо константы «NULL» необходимо использовать «Nil».

3.4 Оценка быстродействия

Для оценки быстродействия скриптового языка были использованы три задачи:

Задача 1:

```
procedure Test1;
var I,J: Integer;
begin
 For I:= 0 to 100000 do J:= I;
end;
```

Задача 2:

```
procedure Test2;
var I: Integer;
    D: Double;
begin
 For I:= 0 to 100000 do D:= sin(I);
end;
```

Задача 3:

```
procedure Test3;
var I: Integer;
    S: String;
begin
 For I:= 0 to 100000 do S:= IntToStr(I);
end;
```

Ориентировочные результаты эксперимента по замеру времени выполнения трех опытных задач на трех компьютерах разной конфигурации приведены в таблице. Для сравнения скобках приведены результаты измерения времени выполнения указанных задач скомпилированных компилятором «Free Pascal Compiler 2.2.0». Результаты измерения указаны в секундах.

	Задача 1	**Задача 2**	**Задача 3**
Intel Pentium 4 2.67 ГГц Microsoft Windows XP	0.011 (0)	0.2 (0.008)	0.32 (0.049)
NS Geode SBC 300 МГц Linux (домовой регистратор)	0.84 (0.004)	15.3 (0.084)	20 (0.4)
ARM920T 180 МГц (без FPU) Linux (блок БКД-ПК)	2.2 (0.021)	127.5 (56.3)	86.3 (7.8)

4. Синтаксис скриптового языка

4.1 Синтаксис программы на языке «PascalScript»

```
Program -> [PROGRAM Ident ';']
           [UsesClause]
           Block '.'
UsesClause -> USES (String/,)... ';'
Block ->   [DeclSection]...
           CompoundStmt
DeclSection          -> ConstSection
                     -> VarSection
```

```
                     -> ProcedureDeclSection
ConstSection -> CONST (ConstantDecl)...
ConstantDecl -> Ident '=' Expression ';'
VarSection -> VAR (VarList ';')...
VarList -> Ident/','... ':' TypeIdent [InitValue]
TypeIdent  -> Ident
           -> Array
Array -> ARRAY '[' ArrayDim/','... ']' OF Ident
ArrayDim   -> Expression..Expression
           -> Expression
InitValue -> '=' Expression
Expression -> SimpleExpression [RelOp SimpleExpression]...
SimpleExpression -> ['-'] Term [AddOp Term]...
Term -> Factor [MulOp Factor]...
Factor     -> Designator
           -> UnsignedNumber
           -> String
           -> '(' Expression ')'
           -> NOT Factor
           -> '[' SetConstructor ']'
SetConstructor -> SetNode/','...
SetNode -> Expression ['..' Expression]
RelOp      -> '>'
           -> '<'
           -> '<='
           -> '>='
           -> '<>'
           -> '='
           -> IN
           -> IS
AddOp      -> '+'
           -> '-'
           -> OR
           -> XOR
MulOp      -> '*'
           -> '/'
           -> DIV
           -> MOD
           -> AND
           -> SHL
           -> SHR
Designator -> ['@'] Ident ['.' Ident | '[' ExprList ']' | '(' ExprList ')']...
ExprList -> Expression/','...
Statement -> [SimpleStatement | StructStmt]
StmtList -> Statement/';'...
SimpleStatement      -> Designator
                     -> Designator ':=' Expression
                     -> BREAK | CONTINUE | EXIT
StructStmt -> CompoundStmt
           -> ConditionalStmt
           -> LoopStmt
           -> TryStmt
           -> WithStmt
CompoundStmt -> BEGIN StmtList END
ConditionalStmt      -> IfStmt
                     -> CaseStmt
IfStmt -> IF Expression THEN Statement [ELSE Statement]
CaseStmt -> CASE Expression OF CaseSelector/';'... [ELSE Statement] [';'] END
CaseSelector -> SetConstructor ':' Statement
LoopStmt   -> RepeatStmt
           -> WhileStmt
           -> ForStmt
RepeatStmt -> REPEAT StmtList UNTIL Expression
WhileStmt -> WHILE Expression DO Statement
ForStmt -> FOR Ident ':=' Expression ToDownto Expression DO Statement
```

```
ToDownto -> (TO | DOWNTO)
TryStmt -> TRY StmtList (FINALLY | EXCEPT) StmtList END
WithStmt -> WITH (Designator/,..) DO Statement
ProcedureDeclSection -> ProcedureDecl
                     -> FunctionDecl
ProcedureDecl -> ProcedureHeading ';'
                     Block ';'
ProcedureHeading -> PROCEDURE Ident [FormalParameters]
FunctionDecl -> FunctionHeading ';'
                     Block ';'
FunctionHeading -> FUNCTION Ident [FormalParameters] ':' Ident
FormalParameters -> '(' FormalParam/';'... ')'
FormalParm -> [VAR | CONST] VarList
```

4.2 Синтаксис программы на языке «C++Script»

```
Program -> [UsesClause]
           [DeclSection]...
           CompoundStmt
UsesClause -> '#' INCLUDE (String/,)...
DeclSection          -> ConstSection
                     -> ProcedureDeclSection
                     -> VarStmt ';'
ConstSection -> '#' DEFINE ConstantDecl
ConstantDecl -> Ident Expression
VarStmt -> Ident Ident [Array] [InitValue] /','...
ArrayDef -> '[' ArrayDim/','... ']'
ArrayDim -> Expression
InitValue -> '=' Expression
Expression -> SimpleExpression [RelOp SimpleExpression]...
SimpleExpression -> ['-'] Term [AddOp Term]...
Term -> Factor [MulOp Factor]...
Factor     -> Designator
           -> UnsignedNumber
           -> String
           -> '(' Expression ')'
           -> '!' Factor
           -> '[' SetConstructor ']'
           -> NewOperator
SetConstructor -> SetNode/','...
SetNode -> Expression ['..' Expression]
NewOperator -> NEW Designator
RelOp      -> '>'
           -> '<'
           -> '<='
           -> '>='
           -> '!='
           -> '=='
           -> IN
           -> IS
AddOp      -> '+'
           -> '-'
           -> '||'
           -> '^'
MulOp      -> '*'
           -> '/'
           -> '%'
           -> '&&'
           -> '<<'
           -> '>>'
Designator -> ['&'] Ident ['.' Ident | '[' ExprList ']' | '(' ExprList ')']...
ExprList   -> Expression/','...
Statement  -> [SimpleStatement ';' | StructStmt | EmptyStmt]
EmptyStmt  -> ';'
StmtList   -> (Statement...)
```

```
SimpleStatement      -> DeleteStmt
                     -> AssignStmt
                     -> VarStmt
                     -> CallStmt
                     -> ReturnStmt
                     -> (BREAK | CONTINUE | EXIT)
DeleteStmt -> DELETE Designator
AssignStmt -> Designator ['+'|'-'|'*'|'/']'=' Expression
CallStmt   -> Designator ['+''+'|'-''-']
ReturnStmt -> RETURN [Expression]
StructStmt -> CompoundStmt
           -> ConditionalStmt
           -> LoopStmt
           -> TryStmt
CompoundStmt -> '{' [StmtList] '}'
ConditionalStmt -> IfStmt
                -> CaseStmt
IfStmt     -> IF '(' Expression ')' Statement [ELSE Statement]
CaseStmt   -> SWITCH'('Expression')' '{'(CaseSelector)...[DEFAULT ':' Statement]'}'
CaseSelector -> CASE SetConstructor ':' Statement
LoopStmt   -> RepeatStmt
           -> WhileStmt
           -> ForStmt
RepeatStmt -> DO Statement [';'] WHILE '(' Expression ')' ';'
WhileStmt  -> WHILE '(' Expression ')' Statement
ForStmt -> FOR '(' ForStmtItem ';' Expression ';' ForStmtItem ')' Statement
ForStmtItem          -> AssignStmt
                     -> VarStmt
                     -> CallStmt
                     -> Empty
TryStmt -> TRY CompoundStmt (FINALLY | EXCEPT) CompoundStmt
FunctionDecl         -> FunctionHeading CompoundStmt
FunctionHeading      -> Ident Ident [FormalParameters]
FormalParameters -> '(' [FormalParam/';'...] ')'
FormalParam -> TypeIdent (['&'] Ident [InitValue]/',')...
```

4.3 Синтаксис программы на языке «BasicScript»

```
Program -> Statements
Statements -> (EOL | StatementList EOL)...
StatementList -> Statement/':'...
ImportStmt -> IMPORTS (String/,)...
DimStmt    -> DIM (VarDecl/',')...
VarDecl    -> Ident [Array] [AsClause] [InitValue]
AsClause   -> AS Ident
Array      -> '[' ArrayDim/','... ']'
ArrayDim   -> Expression
InitValue  -> '=' Expression
Expression -> SimpleExpression [RelOp SimpleExpression]...
SimpleExpression -> ['-'] Term [AddOp Term]...
Term       -> Factor [MulOp Factor]...
Factor     -> Designator
           -> UnsignedNumber
           -> String
           -> '(' Expression ')'
           -> NOT Factor
           -> NewOperator
           -> '<' FRString '>'
SetConstructor -> SetNode/','...
SetNode -> Expression ['..' Expression]
NewOperator -> NEW Designator
RelOp      -> '>'
           -> '<'
           -> '<='
           -> '>='
```

```
           -> '<>'
           -> '='
           -> IN
           -> IS
AddOp      -> '+'
           -> '-'
           -> '&'
           -> OR
           -> XOR
MulOp      -> '*'
           -> '/'
           -> '\'
           -> MOD
           -> AND
Designator -> [ADDRESSOF] Ident ['.' Ident | '[' ExprList ']' | '(' [ExprList] ')']...
ExprList -> Expression/','...
Statement  -> BreakStmt
           -> CaseStmt
           -> ContinueStmt
           -> DeleteStmt
           -> DimStmt
           -> DoStmt
           -> ExitStmt
           -> ForStmt
           -> FuncStmt
           -> IfStmt
           -> ImportStmt
           -> ProcStmt
           -> ReturnStmt
           -> SetStmt
           -> TryStmt
           -> WhileStmt
           -> WithStmt
           -> AssignStmt
           -> CallStmt
BreakStmt  -> BREAK
ContinueStmt -> CONTINUE
ExitStmt   -> EXIT
DeleteStmt -> DELETE Designator
SetStmt -> SET AssignStmt
AssignStmt -> Designator ['+'|'-'|'*'|'/']'=' Expression
CallStmt -> Designator ['+''+'|'-''-']
ReturnStmt -> RETURN [Expression]
IfStmt -> IF Expression THEN ThenStmt
ThenStmt   -> EOL [Statements] [ElseIfStmt | ElseStmt] END IF
           -> StatementList
ElseIfStmt -> ELSEIF Expression THEN
               (EOL [Statements] [ElseIfStmt | ElseStmt] | Statement)
ElseStmt   -> ELSE (EOL [Statements] | Statement)
CaseStmt   -> SELECT CASE Expression EOL
             (CaseSelector...) [CASE ELSE ':' Statements] END SELECT
CaseSelector -> CASE SetConstructor ':' Statements
DoStmt -> DO [Statements] LOOP (UNTIL | WHILE) Expression
WhileStmt  -> WHILE Expression [Statements] WEND
ForStmt    -> FOR Ident '=' Expression TO Expression [STEP Expression] EOL
            [Statements] NEXT
TryStmt -> TRY Statements (FINALLY | CATCH) [Statements] END TRY

WithStmt -> WITH Designator EOL Statements END WITH
ProcStmt -> SUB Ident [FormalParameters] EOL [Statements] END SUB
FuncStmt -> FUNCTION Ident [FormalParameters] [AsClause] EOL
             [Statements] END FUNCTION
FormalParameters -> '(' (FormalParam/',')... ')'
FormalParm -> [BYREF | BYVAL] VarList
```

5. Структура программы

5.1 Структура программы на языке «PascalScript»

5.1.1 Состав программы

Структура программы на языке «PascalScript» аналогична структуре языка «ObjectPascal» и должна состоять из следующих разделов:
- объявление языка программирования (не обязательно);
- объявление имени программы (не обязательно);
- раздел подключаемых модулей (не обязательно);
- объявление констант (не обязательно);
- объявление переменных (не обязательно);
- объявление процедур и функций (не обязательно);
- объявление главного исполнительного модуля.

5.1.2 Пример

```
#language PascalScript            // опционально
program MyProgram;                // опционально
// раздел uses должен быть перед любыми другими разделами
uses 'unit1.pas', 'unit2.pas';
const    pi = 3.14159;
var      i, j: Integer;
procedure p1;                     // процедуры и функции
var i: Integer;
 procedure p2;                    // вложенная процедура
 begin
 end;
begin
end;

begin // главный исполняемый модуль.
end.
```

5.2 Структура программы на языке «C++Script»

5.2.1 Состав программы

Структура программы на языке «C++Script» должна состоять из следующих разделов:
- объявление языка программирования (не обязательно);
- раздел подключаемых модулей (не обязательно);
- объявление констант (не обязательно);
- объявление переменных (не обязательно);
- объявление процедур и функций (не обязательно);
- объявление главной исполняемой функции.

5.2.2 Пример

```
#language C++Script               // опционально
#include "unit1.cpp", "unit2.cpp"
// раздел include - должен быть перед любым другим разделом
```

```
int i, j = 0;              // раздел переменных - может быть в любом месте
#DEFINE pi = 3.14159       // раздел констант
void p1()                  // функции
{                          // вложенных процедур нет
}

{                          // главная исполняемая функция
}
```

5.3 Структура программы на языке «BasicScript»

5.3.1 Состав программы

Структура программы на языке «BasicScript» должна состоять из следующих разделов:
- объявление языка программирования (не обязательно);
- раздел подключаемых модулей (не обязательно);
- объявление переменных (не обязательно);
- объявление процедур и функций (не обязательно);
- объявление главной исполняемой функции.

5.3.2 Пример

```
#language BasicScript              // опционально
// раздел imports - должен быть перед любым другим разделом
imports "unit1.vb", "unit2.vb"
dim i, j = 0               // раздел переменных - может быть в любом месте
Sub p1()                           // объявление функции

End Sub
// главная исполняемая функция
for i = 0 to 10
 p1()
next
```

6. Типы данных

При написании программ можно использовать ряд предопределенных типов данных. Объявление новых типов данных (записи, структуры, классы) не поддерживается.

6.1 Целочисленные

Тип	Диапазон	Формат
pascc++bas Byte	0..255	Без знака, 8 бит
pascc++bas Word	0..65535	Без знака, 16 бит
pascc++bas Integer LongInt c++ int	-2147483648..2147483647	С знаком, 32 бит

long void		
pascc++bas Cardinal LongWord	0..4294967295	Без знака, 32 бит

6.2 Числа с плавающей точкой

Тип	Диапазон	Значащие цифры	Размер, байт
pascc++bas real double TDate TTime TDateTime	5.0×10^{-324} .. 1.7×10^{308}	15-16	8
pascc++bas single	1.5×10^{-45} .. 3.4×10^{38}	7-9	4
pascc++bas extended c++ float	3.6×10^{-4951} .. 1.1×10^{4932}	19-20	10
pascc++bas currency	-922337203685477.5808.. 922337203685477.5807	19-20	8

6.3 Логический

Тип	Значение
pascc++bas Boolean c++ bool	True, False

6.4 Символьный

Тип	Максимальное количество символов	Затраты памяти	Использование
pascc++bas char	1 символ	16 байт	Хранение одного символа
pascc++bas string	2^{31} символов	от 16 байт до 2 ГБайт	Хранение строки символов

6.5 Прочие

Тип	Назначение

pascc++bas Variant	Вариантный тип (хранение данных любого типа)
pascc++bas pointer	Указатель
pas array	Массив

6.6 Совместимость типов

Совместимость при присвоении переменных или свойств различных типов не является симметричной. Значение выражение типа T2 может быть присвоено переменной или свойству с типом T1 если по крайней мере одно из следующих условий удовлетворяется:
- T1 и T2 одного и того же типа;
- T1 и T2 относятся к совместимым целочисленным типам;
- T1 и T2 относятся к типам числа с плавающей точкой;
- T1 — тип числа с плавающей точкой, T2 — целочисленный тип;
- T1 и T2 относятся к типу «class», класс T2 наследуется от класса T1;
- T1 — тип Variant, T2 — любой поддерживаемый тип данных;
- T1 — любой поддерживаемый тип данных, T2 — тип Variant.

6.7 Особенности применения типов в «BasicScript»

Программа написанная на языке «BasicScript» может использовать описание типов переменных и массивов (например dim i as Integer), а может опускать описание типа или объявление переменной. В этом случае тип переменная считается типом Variant.

7. Массивы

Система выполнения скриптов поддерживает следующие типы массивов:
- статические (одномерные, многомерные);
- динамические;
- вариантные.

Пример программы, использующей три массива целых чисел, объявленных разным способом:

```
pas
var ar1: array[0..2] of Integer;
    ar2: array of Integer;
    ar3: Variant;
begin
 SetLength(ar2, 3);
 ar3 := VarArrayCreate([0, 2], varInteger);
 ar1[0]:= 1;
 ar2[0]:= 1;
 ar3[0]:= 1;
end;
```

8. Классы

При использовании скриптового языка программист не может описывать новые типы классов. Тем не менее ему доступны для использования ряд предопределенных внешних

классов объявленных в управляющей программе.

8.1 Стандартные классы

Из приложения на скрипте доступны следующие предопределенные стандартные классы:

Наименование	**Описание**
TObject	Базовый класс
TPersistent	Базовый класс
TList	Список элементов
TStrings	Базовый список строк
TStringList	Список строк
TStream	Базовый поток
TFileStream	Поток в файле
TMemoryStream	Поток в памяти
TfsXMLItem	Элемент XML
TfsXMLDocument	Документ XML

Описание свойств и методов данных классов в объем настоящего документа не входит.

8.2 Классы доступа к данным

8.2.1 Класс TDevice

8.2.1.1 Описание класса

Резюме: Класс — опрашиваемое устройство, источник данных. Скрипт не позволяет динамически создавать экземпляры класса **TDevice**. Указатели на экземпляры класса могут быть возвращены некоторыми свойствами и методами (список **DeviceList**).

Описание:

```
pas
TDevice = class(TObject)
 public
  property DeviceID: String;
  property ProtocolID: String;
  property SOSVersion: String;
  property LastExchange: TDateTime;
  property State: Integer;
  property SOSExchangeCount: Integer;
  property SOSNotAnswerCount: Integer;
  property Period: TDateTime;
  property SOSVcc: Double;
  property Comment: String;
  procedure ImmediatelyDoIO;
```

```
end;
c++
class TDevice : TObject
{
public:
 __property string DeviceID;
 __property string ProtocolID;
 __property string SOSVersion;
 __property TDateTime LastExchange;
 __property int State;
 __property int SOSExchangeCount;
 __property int SOSNotAnswerCount;
 __property TDateTime Period;
 __property Double SOSVcc;
 __property string Comment;
 void ImmediatelyDoIO(void);
}
```

Иерархия: TObject
|
TDevice

8.2.1.2 Свойство DeviceID

Доступ: Только чтение

Описание: Свойство возвращает текстовый идентификатор устройства, идентификатор совпадает с идентификатором, указанным в файле описания устройств, контролируемых управляющей программой.

8.2.1.3 Свойство ProtocolID

Доступ: Только чтение

Описание: Свойство возвращает текстовый идентификатор протокола используемого при обмене данными с устройствами. Возможные значения:
'n\a' — не определено;
'SOS' — протокол «СОС95»;
'SOS(FST)' — протокол «СОС95 Fast»;
'SOS(CRC)' — протокол «СОС95 CRC»
'UDP' — протокол «UDP»;
'TCP' — протокол «TCP»;
'RS232' — протокол «RS232».

8.2.1.4 Свойство SOSVersion

Доступ: Только чтение

Описание: Свойство возвращает текстовый идентификатор версии встроенного программного обеспечения блока сети «СОС95».

8.2.1.5 Свойство LastExchange

Доступ: Только чтение

Описание: Свойство возвращает дату и время последнего обмена данными с устройством в формате **TDateTime**. Если устройство еще не опрашивалось, то возвращается значение 0.

8.2.1.6 Свойство State

Доступ: Только чтение

Описание: Свойство возвращает результат последнего обмена с устройством. При анализе значения свойства возможно использовать константы описанные в пункте 10.2. Если устройство еще не опрашивалось, то возвращается значение 0 (stOk).

8.2.1.7 Свойство SOSExchangeCount

Доступ: Только чтение

Описание: Свойство возвращает общее количество обменов с устройством сети «СОС95».

8.2.1.8 Свойство SOSNotAnswerCount

Доступ: Только чтение

Описание: Свойство возвращает общее количество неудачных обменов с устройством сети «СОС95».

8.2.1.9 Свойство Period

Доступ: Чтение и запись

Описание: При чтении свойства возвращается значение периода опроса устройства в формате **TDateTime**.
При записи устанавливается новый период опроса устройства. При установке значения 0 опрос устройства запрещается.

Пример:

```pas
Device.Period:= OneSecond; // период опроса 1 секунда
Device.Period:= 0;         // запрет опроса
```

8.2.1.10 Свойство SOSVcc

Доступ: Только чтение

Описание: Свойство возвращает напряжение в линии сети «СОС95» в точке подключения устройства. Значение свойства — число с плавающей точкой.

8.2.1.11 Свойство Comment

Доступ: Только чтение

Описание: Свойство возвращает указанный в файле конфигурации текстовый комментарий к устройству.

8.2.1.12 Метод ImmediatelyDoIO

Описание: Вызов метода приводит к выполнению процедуры обмена данными с устройством в минимально возможные сроки.

8.2.2 Класс TChannel

8.2.2.1 Описание класса

Резюме: Класс — канал данных. Скрипт не позволяет динамически создавать экземпляры класса **TChannel**. Указатели на экземпляры класса могут быть возвращены некоторыми свойствами и методами (список каналов **ChannelList**)

Описание:

pas
```
TChannel = class(TObject)
 public
  property A1: Integer;
  property A2: Integer;
  property A3: Integer;
  property A4: Integer;
  property Name: String;
  property DType: Integer;
  property State: Integer;
  property SendCount: Integer;
  property ExpressionOk: Boolean;
  property Expression: String;
  property Time: TDateTime;
  property DataChange: Boolean;
  property DataReady: Boolean;
  property Value: Variant;
  property ValCount: Integer;
  property Values[Index: Integer]: Variant;
  property Control: Variant;
  property Owner: TDevice;
  function AddrIs(A1, A2, A3, A4: Integer): Boolean;
end;
```

c++
```
class TChannel: TObject
{
public:
 __property int A1;
 __property int A2;
 __property int A3;
 __property int A4;
 __property string Name;
```

```
__property int DType;
__property int State;
__property int SendCount;
__property bool ExpressionOk;
__property string Expression;
__property TDateTime Time;
__property bool DataChange;
__property bool DataReady;
__property variant Value;
__property int ValCount;
__property variant Values[int Index];
__property variant Control;
__property TDevice Owner;
bool AddrIs(int A1, int A2, int A3, int A4);
}
```

Иерархия: TObject
|
TChannel

8.2.2.2 Свойства A1, A2, A3, A4

Доступ: Только чтение

Описание: Свойства возвращают адрес информационного канала.

8.2.2.3 Свойство Name

Доступ: Только чтение

Описание: Свойство возвращает строку — текстовый идентификатор канала.

8.2.2.4 Свойство DType

Доступ: Только чтение

Описание: Свойство возвращает числовой идентификатор типа данных канала. При анализе значения можно использовать константы описанные в пункте 10.3.

Пример:

```
pas
var ch: TChannel;
begin
 ...
 If ch.DType = dtFloat then ...
 ...
end;
```

8.2.2.5 Свойство State

Доступ: Чтение и запись

Описание: При чтении свойство возвращает числовой идентификатор качества

данных канала. При записи значения, каналу присваивается значение качества данных в канале.
При анализе и установке значения можно использовать константы описанные в пункте 10.2.

Пример:

```
pas
var ch: TChannel;
begin
 ...
 Case ch.State of
  stOk:      Writeln('Ok');
  stError:   Writeln('Error');
  stFailure: Writeln('Failure');
  else Writeln('other value');
 end;
 ...
end;
```

8.2.2.6 Свойство SendCount

Доступ: Только чтение

Описание: Свойство возвращает значение количества изменений значения и качества данных канала.

8.2.2.7 Свойство ExpressionOk

Доступ: Только чтение

Описание: Свойство возвращает значение «True» если последнее вычисление значения канала выполнено успешно.

8.2.2.8 Свойство Expression

Доступ: Только чтение

Описание: Свойство возвращает значение текстовую строку с выражением используемым для вычисления значения канала. Если выражение отсутствует, то возвращается пустая строка.

8.2.2.9 Свойство Time

Доступ: Только чтение

Описание: Свойство возвращает дату и время последнего изменения состояния канала (изменение данных или качества).

8.2.2.10 Свойство DataChange

Доступ: Чтение и запись

Описание: Свойство возвращает признак того, что данные или качество изменилось и значение канала будет отправлено на сервер .

8.2.2.10 Свойство DataReady

Доступ: Только чтение

Описание: Свойство возвращает признак того, что значение канала прочитано из источника и полю качества данных присвоено корректное значение.

8.2.2.10 Свойство Value

Доступ: Чтение и запись

Описание: При чтении свойство возвращает значение основного поля данных канала. Тип возвращаемого значения определяется значением свойства **DType**. В случае когда свойство **State** <> **stOk** значение поля данных не определено. Если при инициализации скрипта была вызвана процедура установки режима формирования исключений **EnableException**(**True**), то обращение к свойству при **State** <> **stOk** вызывает исключение.
При записи производится присвоение значения основному полю данных канала. Тип присваиваемого значения должен совпадать с типом канала. При несовпадении типа вызывает исключение.

Пример:

```
pas
var C1,C2: TChannel;
    D: Double;
begin
 EnableException(True);
 C1:= ChannelList.Items[0];
 C2:= ChannelList.Items[1];
 try
  D:= C1.Value + C2.Value;    // чтение значения
  C1.Value:= D;               // запись значения
 except
  Writeln('error');
 end;
 ...
end;
```

8.2.2.11 Свойство ValCount

Доступ: Только чтение

Описание: Свойство возвращает количество полей данных для каналов с сложным типом данных.
Некоторые типы каналов несут более одного значения. Например канал с типом **DType** = **dtBDKL** (24) имеет пять значений. Для большинства типов каналов равно 1.

8.2.2.12 Индексное свойство Values

Доступ: Чтение и запись

Описание: При чтении свойство возвращает значение поля данных канала с индексом

Index. Значение **Index** должно быть меньше значения свойства **ValCount**. Обращение по несуществующему индексу вызывает формирование исключения. Обращение к значению с индексом равным 0 аналогично обращению к свойству **Value**. Тип возвращаемого значения определяется значением свойства **DType**. В случае когда свойство **State** <> **stOk** значение поля данных не определено.
Если при инициализации скрипта была вызвана процедура установки режима формирования исключений **EnableException**(**True**), то обращение к свойству при **State** <> **stOk** вызовет исключение.

Пример:

```
pas
var C1,C2: TChannel;
    D: Double;
begin
 EnableException(True);
 C1:= ChannelList.Items[0];
 C2:= ChannelList.Items[1];
 try
  D:= C1.Values[0] + C2.Values[1]; // чтение
  C1.Values[1]:= 1;            // запись 1
  C1.Values[0]:= 3.14;         // запись 2
  C1.Value:= 3.14;             // то же, что запись 2
 except
  Writeln('error');
 end;
 ...
end;
```

8.2.2.13 Свойство Control

Доступ: Чтение и запись

Описание: Чтение свойства полностью аналогично чтению свойства **Value**.
Запись свойства передает команду управления устройству, являющемуся источником данных для канала.
В настоящее время поддерживается передача управления только для каналов с типом данных **dtKey** (**DType** = **dtKey**). Запись значения для каналов других типов игнорируется.
Записываемое значение зависит от типа оборудования. Чаще всего используются следующие значения:
0 — выключить; 1 — включить.
Следует обратить внимание на то, что запись значения ставит команду управления устройством в очередь, и физически управление будет выполнено через некоторый промежуток времени.

Пример:

```
pas
var C: TChannel;
begin
 C:= ChannelList.Items[0];
 C.Control:= 1;               // включить
```

```
 C.Control:= 0;                // выключить
end;
```

8.2.2.14 Свойство Owner

Доступ: Только чтение

Описание: Свойство возвращает объект класса **TDevice** — источника данных для информационного канала.

Пример:

```
pas
var C: TChannel;
    D: TDevice;
begin
// запись в журнал идентификатора устройства,
// являющегося источником данных для канала
 C:= ChannelList.Items[0];
 D:= C.Owner;
 Writeln(D.DeviceID);
end;
```

8.2.2.15 Метод AddrIs

Описание: Метод возвращает «True» если адрес указанный в параметрах метода совпадает с адресом канала.

Пример:

```
pas
var C: TChannel;
begin
 C:= ChannelList.Items[0];
 If C.AddrIs(1,1,1,1) then Writeln('this 1.1.1.1');
// альтернативный вариант
 If C.A1=1 and C.A2=1 and C.A3=1 and C.A4=1 then
  Writeln('this 1.1.1.1');
end;
```

8.2.3 Класс TDeviceList

8.2.3.1 Описание класса

Резюме: Класс — список опрашиваемых устройств. Скрипт не позволяет динамически создавать экземпляры класса **TDeviceList**. В программе предопределена переменная **DeviceList** являющаяся экземпляром класса **TDeviceList**, содержащая список всех опрашиваемых устройств текущим потоком управляющей программы.

Описание:

```
pas
TDeviceList = class(TObjectList)
 public
  property Count: Integer;
  function CountOnState(St: Integer): Integer;
```

```
  function AverageQuality: Integer;
  property Items[Index: Integer]: TDevice;
end;
c++
class TDeviceList: TObjectList
{
public:
 __property int Count;
 int CountOnState(int St);
 int AverageQuality;
 __property TDevice Items[int Index];
}
```

Иерархия:

```
TObject
   |
TObjectList
   |
TDeviceList
```

8.2.3.2 Свойство Count

Доступ: Только чтение

Описание: Свойство возвращает количество объектов в списке.

8.2.3.3 Индексное свойство Items

Доступ: Только чтение

Описание: Возвращает объект класса **TDevice** с индексом **Index** из списка. Значение индекса должно быть меньше значения свойства **Count**. При обращении к несуществующему элементу списка формируется исключение.

Пример:

```
pas
var D: TDevice;
begin
 D:= DeviceList.Items[0];
end;
c++
TDevice D = DeviceList.Items[0];
```

8.2.3.4 Метод CountOnState

Описание: Метод возвращает количество устройств из списка, свойство **State** которых равно значению параметра St. При указании параметра можно использовать константы описанные в пункте 10.2.

Пример:

```
pas
Var I: Integer;
begin
 I:= DeviceList.CountOnState(stOk);
 // переменной I присвоено значение количества
 // устройств, находящихся в состоянии stOk
end;
```

8.2.3.5 Метод AverageQuality

Описание: Метод возвращает среднее качество связи со всеми устройствами списка. Только для устройств сети «СОС95».

8.2.4 Класс TChannelList

8.2.4.1 Описание класса

Резюме: Класс — список информационных каналов. Скрипт не позволяет динамически создавать экземпляры класса **TChannelList**. В программе предопределена переменная **ChannelList** являющаяся экземпляром класса **TChannelList**, содержащая список всех информационных каналов текущего потока управляющей программы.

Описание:

```
pas
TChannelList = class(TObjectList)
 public
  property Count: Integer;
  function CountOnState(St: Integer): Integer;
  property Items[Index: Integer]: TChannel;
  procedure SendAll;
  function Get(A1, A2, A3, A4: Integer): TChannel;
  function Get2(Addr: String): TChannel;
end;
c++
class TChannelList: TObjectList
{
public:
 __property int Count;
 int CountOnState(int St);
 __property TChannel Items[int Index];
 void SendAll(void);
 TChannel Get(int A1, int A2, int A3, int A4);
 TChannel Get2(string Addr);
}
```

Иерархия:

```
TObject
   |
TObjectList
   |
```

TChannelList

8.2.4.2 Свойство Count

Доступ: Только чтение

Описание: Свойство возвращает количество объектов в списке.

8.2.4.3 Индексное свойство Items

Доступ: Только чтение

Описание: Возвращает объект класса **TChannel** с индексом **Index** из списка. Значение индекса должно быть меньше значения свойства **Count**. При обращении к несуществующему элементу списка формируется исключение.

Пример:

```
pas
var C: TChannel;
begin
 C:= ChannelList.Items[0];
end;
c++
TChannel C = ChannelList.Items[0];
```

8.2.4.4 Метод CountOnState

Описание: Метод возвращает количество каналов данных из списка, свойство **State** которых равно значению параметра St. При указании параметра можно использовать константы описанные в пункте 10.2.

Пример:

```
pas
var I: Integer;
begin
 I:= ChannelList.CountOnState(stOk);
 // переменной I присвоено значение количества
 // каналов, находящихся в состоянии stOk
end;
```

8.2.4.5 Метод SendAll

Описание: Присваивает свойству **DataChange** всех каналов списка значение **True**.

8.2.4.6 Метод Get

Описание: Метод возвращает канал из списка адрес которого указан в параметрах **A1**, **A2**, **A3**, **A4**. При отсутствии канала с указанным адресом возвращается **Nil**.

8.2.4.7 Метод Get2

Описание: Метод похож на метод **Get**, отличие заключается в том, что адрес необходимого канала указан в виде строки Addr. Параметр Addr должен иметь вид: 'A1.A2.A3.A4' — строка из четырех чисел разделенных символом точки. При отсутствии канала с указанным адресом возвращается Nil.

Пример:

```
pas
var C: TChannel;
begin
 C:= ChannelList.Get(1, 1, 1, 1);
 // аналогично
 C:= ChannelList.Get2('1.1.1.1');
 If C <> Nil then Writeln('channel found');
end;
```

8.2.5 Класс TChannel2

8.2.5.1 Описание класса

Резюме: Класс — канал данных тип 2.
Экземпляры класса создаются автоматически управляющей программой в процессе опроса подключенного оборудования. Кроме того, новый экземпляр класса может быть создан путём вызова метода **AddChannel2** (см. 9.8.1). Указатели на существующие экземпляры класса могут быть возвращены некоторыми свойствами и методами (список каналов **Channel2List**)

Описание:

```
pas
TChannel2 = class(TObject)
 public
  property Addr: String;
  property ID: String;
  property Active: Boolean;
  property Changed: Boolean;
  property Type: Integer;
  property TypeText: String;
  property Time: TdateTime;
  property Quality: Integer;
  property SaveServerValue: Boolean;
  property State: Integer;
  property Value: Variant;
  property ValueText: String;
  property WriteEnable: Boolean;
  function AddrIs(Addr: String): Boolean;
end;
c++
class TChannel2: TObject
{
public:
```

```
__property string Addr;
__property string ID;
__property bool Active;
__property bool Changed;
__property int Type;
__property string TypeText;
__property TDateTime Time;
__property int Quality;
__property bool SaveServerValue;
__property int State;
__property variant Value;
__property string ValueText;
__property bool WriteEnable;
bool AddrIs(string Addr);
}
```

Иерархия: TObject
|
TChannel2

8.2.5.2 Свойства Addr, ID

Доступ: Только чтение

Описание: Свойства **Addr** и **ID** являются синонимами.
Свойства возвращают текстовое значение адреса канала.

8.2.5.3 Свойство Active

Доступ: Чтение и запись

Описание: Свойство возвращает текущее значение признака активности канала. При записи свойства устанавливается признак активности, используемый при регистрации канала на сервере.
Текущая реализация не позволяет изменять активность существующих каналов в реальном времени и имеет практический смысл только для новых каналов создаваемых из скрипта.

Пример:

```
pas
Tchannel2 newch;
newch = AddChannel2("test", varByte, 0, "", "", True);
newch.Active = True;
```

8.2.5.4 Свойство Changed

Доступ: Чтение и запись

Описание: Свойство возвращает признак изменения значения канала со стороны сервера (команда управления). Свойство может принимать значение **True** только для каналов доступных для записи (свойство **WriteEnable** имеет значение **True**). Значение свойства может быть установлено в **False** в процедуре обработки управления (см. 12.7 Получение команды управления

для канала тип 2), в этом случае команда не будет передана соответствующему драйверу устройства.

8.2.5.5 Свойство Type

Доступ: Только чтение

Описание: Свойство возвращает числовое значение типа данных канала.
Для анализа могут быть использованы значения констант описанных в пункте 10.7.

8.2.5.6 Свойство TypeText

Доступ: Только чтение

Описание: Свойство возвращает текстовое наименование типа данных канала.

8.2.5.7 Свойство Time, DT

Доступ: Только чтение

Описание: Свойства **Time** и **DT** являются синонимами.
Свойство возвращает значение даты и времени последнего изменения канала (значения или качества).

8.2.5.8 Свойство SaveServerValue

Доступ: Чтение и запись

Описание: Чтение или установка признака необходимости сохранения значения канала на сервере при отключении или обрыве связи с управляющей программой.

8.2.5.9 Свойство Quality, State

Доступ: Чтение и запись

Описание: Качество значения канала, содержит информацию о его достоверности.
Свойства **Quality** и **State** являются синонимами.
Свойство может принимать значения описанные в пункте 10.2.
При установке значение свойства его новое состояние будет передано на сервер. Практический смысл имеет установка только значений отличных от **stOk**, значение **stOk** устанавливается автоматически при установке значения свойства **Value**. При анализе и установке значения можно использовать константы описанные в пункте 10.2.

8.2.5.10 Свойство Value

Доступ: Чтение и запись

Описание: При чтении свойство возвращает значение данных канала. Тип возвращаемого значения определяется значением свойства Type. В случае когда свойство **Quality** <> **stOk** значение данных не определено.

Если при инициализации скрипта была вызвана процедура установки режима формирования исключений **EnableException**(**True**), то обращение к свойству при **State** <> **stOk** вызывает исключение.
При установке свойства необходимо следить за тем, что тип присваиваемого значения должен соответствовать типу канала. При установке значения свойства новое состояние канала будет передано на сервер. При этом значению свойства **Quality** будет автоматически присвоено значение **stOk**. Если тип канала является числом с плавающей точкой, то перед отправкой нового значения используется механизм фильтрации.

8.2.5.11 Свойство ValueText

Доступ: Только чтение

Описание: Значение канала преобразованное в строку. Если значение свойства **Quality** не равно **stOk** (значение 0), то свойство возвращает пустую строку.

8.2.5.12 Свойство Control

Доступ: Чтение и запись

Описание: Чтение свойства полностью аналогично чтению свойства **Value**.
Запись свойства передает команду управления устройству, являющемуся источником данных для канала. Запись поддерживается только для каналов созданных автоматически драйвером опроса оборудования, значение свойства **WriteEnable** которых равно **True**. Для каналов созданных методом **AddChannel2**() запись свойства **Control** не поддерживается, т.к. они не связаны с реальным оборудованием.
Следует обратить внимание на то, что запись значения ставит команду управления устройством в очередь, и физически управление будет выполнено через некоторый промежуток времени.

Пример:

```
pas
var C: TChannel2;
begin
 // изменение точки регулирования CAT-500
 C:= Channel2List.Get['CAT500_2202080_SP0'];
 If C <> Nil then C.Control:= 55;
end;
```

8.2.5.13 Свойство WriteEnable

Доступ: Только чтение

Описание: Свойство возвращает признак доступности записи значений в канал со стороны сервера системы.

8.2.5.14 Метод AddrIs

Описание: Метод возвращает True если значение его параметра совпадает с адресом

канала.

8.2.6 Класс TChannel2List

8.2.6.1 Описание класса

Резюме: Класс — список информационных каналов типа 2. Скрипт не позволяет динамически создавать экземпляры класса **TChannel2List**. В программе предопределена переменная **Channel2List** являющаяся экземпляром класса **TChannel2List**, содержащая список всех информационных каналов типа 2 текущего потока управляющей программы.

Описание:

```
pas
TChannel2List = class(TObjectList)
 public
  property Count: Integer;
  property Items[Index: Integer]: TChannel2;
  function Get(Addr: String): TChannel2;
end;
c++
class TChannel2List: TObjectList
{
public:
 __property int Count;
 __property TChannel2 Items[int Index];
 TChannel2 Get(string Addr);
}
```

Иерархия:

```
TObject
   |
TObjectList
   |
TChannelList
```

8.2.6.2 Свойство Count

Доступ: Только чтение

Описание: Свойство возвращает количество объектов (каналов тип 2) в списке. Каналы тип 2 формируются автоматически по мере опроса оборудования, поэтому значение свойства может изменяться (увеличиваться) по мере работы управляющей программы.

8.2.6.3 Индексное свойство Items

Доступ: Только чтение

Описание: Возвращает объект класса **TChannel2** с индексом **Index** из списка. Значение индекса должно быть меньше значения свойства **Count**. При обращении к несуществующему элементу списка формируется исключение.

Пример:

```
pas
var C: TChannel2;
begin
 C:= Channel2List.Items[0];
end;
c++
TChannel2 C = Channel2List.Items[0];
```

8.2.6.4 Метод Get

Описание: Метод возвращает объект класса **TChannel2** из списка, адрес канала указывается в параметре **Addr**.
При отсутствии требуемого канала возвращается значение **Nil**. Каналы типа 2 формируются автоматически по мере опроса оборудования, первоначально они отсутствуют. Кроме того, отсутствие необходимого канала может быть вызвано тем, что подключенное устройство не отвечает и формируемый им канал не создан. В реальных условиях может потребоваться несколько повторных периодических попыток получения объекта класса **TChannel2**.

Пример:

```
pas
var C: TChannel2;
begin
 C:= Channel2List.Get('vist_12345_T1');
 If C <> Nil then Writeln('channel found');
end;
```

8.2.7 Класс TLocalStorage

8.2.7.1 Описание класса

Резюме: Класс — хранилище данных в формате «Ключ — Значение».
Класс позволяет сохранять и получать ранее сохранённые значения в постоянном хранилище, значения в котором сохраняются, в том числе, после отключения питания устройства. Хранилище представляет собой набор элементов данных. Каждому элементу данных соответствует уникальный текстовый ключ.
Скрипт не позволяет динамически создавать экземпляры класса **TLocalStorage**.
В программе предопределена переменная **LocalStorage** являющаяся экземпляром класса **TLocalStorage**, предоставляющая доступ к методам класса.
Для каждого потока управляющей программы создаётся свой отдельный экземпляр класса **TLocalStorage.**

Описание:

```
pas
TLocalStorage = class(TObject)
 public
  procedure SetItem(Key: String; Value: Variant);
  function GetItem(Key, DefVal:String): String;
  function AsInt(Key:String; DefVal:Integer): Integer;
```

```
  function AsDouble(Key: String; DefVal: Double):Double;
  function AsBool(Key: String; DefVal: Boolean):Boolean;
  function HasItem(Key: String): Boolean;
  procedure RemoveItem(Key: String);
  procedure Clear;
end;
```

c++

```
class TLocalStorage: TObject
{
public:
 void SetItem(String Key, Variant Value);
 String GetItem(String Key, String DefVal);
 int AsInt(String Key, int DefVal);
 double AsDouble(String Key, double DefVal);
 bool AsBool(String Key, ble DefVal);
 bool HasItem(String Key);
 void RemoveItem(String Key);
 void Clear(void)
}
```

Иерархия: TObject
|
TLocalStorage

8.2.7.2 Метод SetItem

Описание: Устанавливает значение элемента хранилища. Если хранилище уже содержит элемент с указанным ключом, то его значение будет заменено на новое. Для предотвращения преждевременного исчерпания ресурса Flash-накопителя фактическая запись изменённых значений на диск производится не чаще чем один раз в две минуты. Если питание устройства будет отключено до записи изменений на диск, то они могут быть утеряны.

Параметры: **Key** – ключ значения. Ключ должен представлять собой строку символов, Допустимыми символами ключа являются «A..Z,a..z,0..9,_».
Value – значение для помещения в хранилище. Значение должно представлять собой произвольную строку символов. Если тип **Value** не является строкой, то будет выполнено автоматическое преобразование к типу **String**.

Пример:

pas

```
begin
 LocalStorage.SetItem('key1', '1.2345');
 LocalStorage.SetItem('key1', 123);
end;
```

8.2.7.3 Метод GetItem

Описание: Получение элемента данных из хранилища.

Параметры: **Key** – ключ значения. Ключ должен представлять собой строку символов, Допустимыми символами ключа являются «A..Z,a..z,0..9,_».
DefValue – значение «по умолчанию», которое будет возвращено в случае, когда элемент данных с ключом **Key** отсутствует.

Пример:

```
pas
begin
 A:= LocalStorage.GetItem('key1', '123');
end;
```

8.2.7.4 Метод AsInt

Описание: Метод возвращает значение элемента хранилища с ключом **Key** преобразованное к типу **Integer**. При отсутствии элемента или невозможности преобразования возвращается значение **DefVal**.

Параметры: **Key** – ключ значения.
DefVal – значение по умолчанию.

8.2.7.5 Метод AsDouble

Описание: Метод возвращает значение элемента хранилища с ключом **Key** преобразованное к типу **Double**. При отсутствии элемента или невозможности преобразования возвращается значение **DefVal**.

Параметры: **Key** – ключ значения.
DefVal – значение по умолчанию.

8.2.7.6 Метод AsBool

Описание: Метод возвращает значение элемента хранилища с ключом **Key** преобразованное к типу **Boolean**. При преобразовании используются следующие правила:
- если содержимое элемента может быть преобразовано к целому числу, то результатом является **False** если число равно 0 или **True** в иных случаях;
- если содержимое элемента не может быть преобразовано к целому числу, то результатом является **False** если значение элемента равно строке «False» или **True** если значение элемента равно строке «True».
При отсутствии элемента или невозможности преобразования возвращается значение **DefVal**.

Параметры: **Key** – ключ значения.
DefVal – значение по умолчанию.

8.2.7.7 Метод HasItem

Описание: Метод возвращает значение **True** если элемент данных с указанным ключом присутствует в хранилище.

Параметры: **Key** – ключ значения.

8.2.7.8 Метод RemoveItem

Описание: Метод удаляет из хранилища элемент данных с указанным ключом. Если хранилище не содержит такой элемент данных, то никаких действий не производится.

Параметры: **Key** – ключ значения.

8.2.7.9 Метод Clear

Описание: Метод удаляет все имеющиеся элементы данных хранилища.

9. Встроенные процедуры и функции

Скрипт содержит ряд встроенных стандартных процедур и функций, которые можно использовать из пользовательских программ.

9.1 Преобразование типов

9.1.1 IntToStr

Резюме: Преобразование целого числа в строку

Описание:

pas
```
function IntToStr(i: Integer): String
```
c++
```
string IntToStr(int I)
```

Назначение: Функция предназначена для преобразования целого числа в строку с его десятичным представлением

9.1.2 FloatToStr

Резюме: Преобразование числа с плавающей точкой в строку

Описание:

pas
```
function FloatToStr(e: Extended): String
```
c++
```
string FloatToStr(Extended e)
```

Назначение: Функция предназначена для преобразования числа с плавающей точкой в строку с его десятичным представлением. При преобразовании используется «общий» формат с 15 значащими цифрами.

9.1.3 DateToStr

Резюме: Преобразование даты в формате TDateTime в строку

Описание:

pas
```
function DateToStr(e: Extended): String
```
c++
```
string DateToStr(Extended e)
```

Назначение: Функция возвращает строковое представление даты, при форматировании

строки используются текущие настройки операционной системы.

9.1.4 TimeToStr

Резюме: Преобразование времени в формате TDateTime в строку

Описание:

pas
```
function TimeToStr(e: Extended): String
```
c++
```
string TimeToStr(Extended e)
```

Назначение: Функция возвращает строковое представление времени, при форматировании строки используются текущие настройки операционной системы.

9.1.5 DateTimeToStr

Резюме: Преобразование даты и времени в формате TDateTime в строку

Описание:

pas
```
function DateTimeToStr(e: Extended): String
```
c++
```
string DateTimeToStr(Extended e)
```

Назначение: Функция возвращает строковое представление даты и времени, при форматировании строки используются текущие настройки операционной системы.

9.1.6 VarToStr

Резюме: Преобразование значения переменной типа “Variant” в строку

Описание:

pas
```
function VarToStr(e: Variant): String
```
c++
```
string VarToStr(variant e)
```

Назначение: Функция возвращает строковое представление значения переменной типа “**Variant**”. Если значение переменной равно **Nil**, то результат – пустая строка. При невозможности преобразования формируется исключение.

9.1.7 StrToInt

Резюме: Преобразование строки в целое число

Описание:

pas
```
function StrToInt(s: String): Integer
```
c++
```
int StrToInt(string s)
```

Назначение: Функция возвращает целое число – результат преобразования строковой переменной.
При невозможности преобразования формируется исключение.

9.1.8 StrToFloat

Резюме: Преобразование строки в число с плавающей точкой

Описание:

pas
```
function StrToFloat(s: String): Extended
```
c++
```
extended StrToFloat(string S)
```

Назначение: Функция возвращает число с плавающей точкой – результат преобразования строковой переменной.
При невозможности преобразования формируется исключение.

9.1.9 StrToDate

Резюме: Преобразование строки в значение даты в формате TDateTime

Описание:

pas
```
function StrToDate(s: String): Extended
```
c++
```
extended StrToDate(string s)
```

Назначение: Функция возвращает число с плавающей точкой, дату в формате **TDateTime** – результат преобразования строковой переменной.
При невозможности преобразования формируется исключение.

9.1.10 StrToTime

Резюме: Преобразование строки в значение времени в формате **TDateTime**

Описание:

pas
```
function StrToTime(s: String): Extended
```
c++
```
extended StrToTime(string s)
```

Назначение: Функция возвращает число с плавающей точкой, время в формате **TDateTime** – результат преобразования строковой переменной.
При невозможности преобразования формируется исключение.

9.1.11 StrToDateTime

Резюме: Преобразование строки в значение дата и времени в формате **TDateTime**

Описание:

pas
```
function StrToDateTime(s: String): Extended
```
c++
```
extended StrToDateTime(string s)
```

Назначение: Функция возвращает число с плавающей точкой, дата и время в формате **TDateTime** – результат преобразования строковой переменной.
При невозможности преобразования формируется исключение.

9.2 Форматирование

9.2.1 Format

Резюме: Формирование строки в соответствии с спецификацией формата и перечнем параметров

Описание:

pas
```
function Format(Fmt: String; Args: array): String
```
c++
```
string Format(string Fmt, array Args)
```

Назначение: Функция преобразует несколько аргументов Args в строку в соответствии со спецификацией форматирования Fmt.
В простом случае спецификация форматирования включает в себя элементы, начинающиеся с символа **'%'** и заканчиваются индикатором типа данных:
d = десятичное (целое число);
e = плавающая точка в “научном” формате;
f = плавающая точка в “фиксированном” формате;
g = плавающая точка в “общем” формате;
m = “денежный” формат;
n = плавающая точка;
p = значение указателя;
s = строка символов;
u = десятичное число без знака;
x = число в шестнадцатеричном представлении.
Общий формат форматирования каждой подстроки следующий:
%[Index:][-][Width][.Precision]Type
Подробное описание спецификаций форматирование в настоящий документ не входит и может быть получено в интернете по адресу:
http://community.freepascal.org:10000/docs-html/rtl/sysutils/format.html
При неверных аргументах и невозможности преобразования формируется исключение.

Пример:

pas
```
Var I: Integer;
    S, RES: String;
    D: Double;
begin
 I:= 1;
 S:= 'ура';
 D:= 3.14;
 RES:= Format('I=%d S=%s D=%f', [I, S, D]);
 // результат: RES= 'I=1 S=ура D=3.14';
end;
```

9.2.2 FormatFloat

Резюме: Преобразование числа с плавающей точкой в строку в соответствии с спецификацией формата

Описание:

pas
```
function FormatFloat(Fmt: String; Value: Extended):
```

```
String
c++
string FormatFloat(string Fmt, extended Value)
```

Назначение: Функция преобразует число с плавающей точкой **Value** в строку в соответствии со спецификацией форматирования **Fmt**.
Описание спецификаций форматирование в настоящий документ не входит и может быть получено в интернете по адресу:
http://community.freepascal.org:10000/docs-html/rtl/sysutils/formatfloat.html
При неверных аргументах и невозможности преобразования формируется исключение.

9.2.3 FormatDateTime

Резюме: Преобразование даты и времени в формате **TDateTime** в строку в соответствии с спецификацией формата

Описание:

```
pas
function FormatDateTime(Fmt: String; DT: TDateTime):
String
c++
string FormatDateTime(string Fmt, TDateTime DT)
```

Назначение: Функция преобразует дату и время **DT** в строку в соответствии со спецификацией форматирования **Fmt**. В спецификации форматирования могут быть использованы следующие символы:
c – дата и время в короткой записи;
d – день месяца;
dd – день месяца с лидирующим нулем;
ddd – день недели (сокращенное наименование);
dddd – день недели (полное наименование);
ddddd – короткая запись даты;
dddddd – длинная запись даты;
m – месяц;
mm – месяц с лидирующим нулем;
mmm – месяц (сокращение);
mmmm – месяц (полное наименование);
y – год (две цифры);
yy – год (две цифры);
yyyy – год (с столетием);
h – час;
hh – час с лидирующим нулем;
n – минута;
nn – минута с лидирующим нулем;
s – секунда;
ss – секунда с лидирующим нулем;
t – короткий формат времени;
tt – длинный формат времени;
/ - разделитель даты;

: - разделитель времени.
При неверных аргументах и невозможности преобразования формируется исключение.

Пример:

pas
```
Writeln(FormatDateTime('dd-mm-yyyy hh:nn:ss', Now));
// результат (для тек.даты и времени): 20-01-2008 13:15:17
```

9.3 Работа с датой и временем

9.3.1 EncodeDate

Резюме: Преобразование числовых значений года, месяца и дня в значение типа TDateTime

Описание:

pas
```
function EncodeDate(Year, Month, Day: Word): TDateTime
```
c++
```
TDateTime EncodeDate(Word Year, Month, Day)
```

Назначение: Функция возвращает значение даты в формате **TDateTime** – результат преобразования числовых значений года (**Year**), месяца (**Month**) и дня (**Day**).
При невозможности преобразования формируется исключение.

9.3.2 EncodeTime

Резюме: Преобразование числовых значений часа, минуты, секунды и миллисекунды в значение типа **TDateTime**

Описание:

pas
```
function EncodeTime(Hour, Min, Sec, MSec: Word):
TDateTime
```
c++
```
TDateTime EncodeTime(Word Hour, Min, Sec, MSec)
```

Назначение: Функция возвращает значение времени в формате **TDateTime** – результат преобразования числовых значений часа (**Hour**), минуты (**Min**), секунды (**Sec**) и миллисекунды (**MSec**).
При невозможности преобразования формируется исключение.

9.3.3 DecodeDate

Резюме: Преобразование значения типа **TDateTime** в числовые значения даты

Описание:

pas
```
procedure DecodeDate(Date: TDateTime; var Year, Month,
Day: Word);
```
c++
```
void DecodeDate(TDateTime Date, Word &Year, Word &Month,
Word &Day)
```

Назначение: Процедура преобразует значение даты (**Date**) в числовые значения года (**Year**), месяца (**Month**) и дня (**Day**).

При невозможности преобразования формируется исключение.

9.3.4 DecodeTime

Резюме: Преобразование значения типа **TDateTime** в числовые значения

Описание:

pas
```
procedure DecodeTime(Date: TDateTime; var Hour, Min,
Sec, MSec: Word);
```
c++
```
void DecodeTime(TDateTime Date, Word &Hour, Word &Min,
Word &Sec, Word &MSec)
```

Назначение: Процедура преобразует значение времени (**Time**) в числовые значения часа (**Hour**), минуты (**Min**), секунды (**Sec**) и миллисекунды (**MSec**).
При невозможности преобразования формируется исключение.

9.3.5 Date

Резюме: Возвращает текущую дату

Описание:

pas
```
Function Date: TDateTime;
```
c++
```
TDateTime Date(void)
```

Назначение: Функция возвращает значение текущей даты в формате **TDateTime**.

9.3.6 Time

Резюме: Возвращает текущее время

Описание:

pas
```
Function Time: TDateTime;
```
c++
```
TDateTime Time(void)
```

Назначение: Функция возвращает значение текущего времени в формате **TDateTime**.

9.3.6 Now

Резюме: Возвращает текущее время и дату

Описание:

pas
```
Function Now: TDateTime;
```
c++
```
TDateTime Now(void)
```

Назначение: Функция возвращает значение текущего времени и даты в формате **TDateTime**.

9.3.7 DayOfWeek

Резюме: Возвращает номер дня недели для указанной даты и времени

Описание:

pas
```
Function DayOfWeek(Date: TDateTime): Integer;
```
c++
```
int DayOfWeek(TDateTime Date)
```

Назначение: Функция возвращает номер дня недели для указанной даты. Воскресенье – 1, понедельник – 2, вторник – 3, среда – 4, четверг – 5, пятница – 6, суббота – 7.

9.3.8 IsLeapYear

Резюме: Определение високосного года.

Описание:

pas
```
Function IsLeapYear(Year: Word): Boolean
```
c++
```
bool IsLeapYear(Word Year)
```

Назначение: Функция возвращает TRUE для високосных номеров года.

9.3.9 DaysInMonth

Резюме: Определение количества дней в месяце.

Описание:

pas
```
Function DaysInMonth(Year, Month: Word): Integer
```
c++
```
int DaysInMonth(Word Year, Month)
```

Назначение: Функция возвращает количество дней в месяце для указанного года (Year) и месяца (Month).

9.3.10 GetTickCount

Резюме: Возвращает количество миллисекунд, прошедшее с момента запуска операционной системы.

Описание:

pas
```
Function GetTickCount: LongWord;
```
c++
```
LongWord GetTickCount(void)
```

Назначение: Функция возвращает количество миллисекунд, прошедшее с момента запуска операционной системы.

9.3.11 TickDelta

Резюме: Возвращает разницу между значениями количества миллисекунд.

Описание:

pas
```
function TickDelta(TickOld, TickNew:LongWord): LongWord
```
c++
```
LongWord TickDelta(LongWord TickOld, LongWord TickNew)
```

Назначение: Функция возвращает разницу между значениями количества миллисекунд **TickOld** и **TickNew**.

9.4 Строковые функции

9.4.1 Length

Резюме: Длина строки

Описание:

pas
```
Function Length(S: String): Integer;
```
c++
```
int Length(String S)
```

Назначение: Функция возвращает количество символов в строке.

9.4.2 Copy

Резюме: Возвращает подстроку из строки

Описание:

pas
```
Function Copy(S: String; From, Count: Integer): String;
```
c++
```
String Copy(String S, int From, int Count)
```

Назначение: Функция возвращает подстроку из строки S, начиная с позиции **From**, состоящую из **Count** символов. Если значение **From** больше чем длина строки S, то результат — пустая строка.

9.4.3 Pos

Резюме: Поиск подстроки в строке

Описание:

pas
```
Function Pos(SubS, S: String): Integer;
```
c++
```
int Pos(String SubS, String S)
```

Назначение: Поиск подстроки **SubS** в строке **S**. Если подстрока найдена, то результат функции — позиция подстроки в строке, если не найдена, то результат 0.

9.4.4 Delete

Резюме: Удаление части строки

Описание:

pas
```
procedure Delete(var S: String; From, Count: Integer)
```
c++
```
void Delete(String &S, int From, int Count)
```

Назначение: Процедура удаляет **Count** символов из строки **S** начиная с позиции **From**.

9.4.5 Insert

Резюме: Вставка одной строки в другую

Описание:

pas
```
procedure Insert(S: String;var S2:String;Pos:Integer)
```
c++
```
void Insert(String S, String &S2, int Pos)
```

Назначение: Процедура вставляет содержимое строки **S** в позицию **Pos** строки **S2**.

9.4.6 UpperCase

Резюме: Перевод символов строки в верхний регистр

Описание:

pas
```
function UpperCase(S: String): String
```
c++
```
string UpperCase(String S)
```

Назначение: Функция возвращает строку состоящую из символов строки **S** переведенных в верхний регистр.

9.4.7 LowerCase

Резюме: Перевод символов строки в нижний регистр

Описание:

pas
```
function LowerCase(S: String): String
```
c++
```
string LowerCase(String S)
```

Назначение: Функция возвращает строку состоящую из символов строки **S** переведенных в нижний регистр.

9.4.8 Trim

Резюме: Удаление пробелов в начале и конце строки

Описание:

pas
```
function Trim(S: String): String
```
c++
```
string Trim(String S)
```

Назначение: Функция возвращает строку **S** без пробелов в ее начале и конце.

9.4.9 NameCase

Резюме: Перевод первого символа строки в верхний регистр

Описание:

pas
```
function NameCase(S: String): String
```
c++
```
string NameCase(String S)
```

Назначение: Функция переводит первый символ строки **S** в верхний регистр.

9.4.10 CompareText

Резюме: Сравнение строк

Описание:

pas
```
function CompareText(S1, S2: String): Integer
```
c++
```
int CompareText(String S1, String S2)
```

Назначение: Функция сравнивает содержимое строк **S1** и **S2**. Результат:
0 — строки одинаковые;
>0 — строка S1 больше чем S2;
<0 — строка S1 меньше чем S2.

9.4.11 Chr

Резюме: Возвращает символ с заданным номером

Описание:

pas
```
function Chr(I: Integer): Char
```
c++
```
char Chr(int I)
```

Назначение: Функция возвращает символ с кодом **I**.

9.4.12 Ord

Резюме: Возвращает код символа

Описание:

pas
```
function Ord(C: Char): Integer
```
c++
```
int Ord(char C)
```

Назначение: Функция возвращает код символа **C**.

9.4.13 SetLength

Резюме: Установка длины строки

Описание:

pas
```
procedure SetLength(var S: String; L: Integer)
```
c++
```
void SetLength(String &S, int L)
```

Назначение: Процедура устанавливает длину строки **S** в значение **L.**

9.5 Математические функции

9.5.1 Round

Резюме: Округление до ближайшего целого значения

Описание:

pas
```
function Round(E: Extended): Integer
```
c++
```
int Round(float E)
```

Назначение: Функция возвращает результат округления аргумента **E** до ближайшего целого значения

9.5.2 Trunc

Резюме: Округление до меньшего целого значения

Описание:

pas
```
function Trunc(E: Extended): Integer
```
c++
```
int Trunc(float E)
```

Назначение: Функция возвращает результат округления аргумента **E** до меньшего целого значения

9.5.3 Int

Резюме: Целая часть

Описание:

pas
```
function Int(E: Extended): Integer
```
c++
```
int Int(float E)
```

Назначение: Функция возвращает целую часть аргумента **E**

9.5.4 Frac

Резюме: Дробная часть

Описание:

pas
```
function Frac(E: Extended): Extended
```
c++
```
float frac(float E)
```

Назначение: Функция возвращает дробную часть аргумента **E**

9.5.5 Sqrt

Резюме: Извлечение квадратного корня

Описание:

pas
```
function Sqrt(E: Extended): Extended
```
c++
```
float sqrt(float E)
```

Назначение: Функция возвращает значение квадратного корня аргумента **E**

9.5.6 Abs

Резюме: Абсолютное значение

Описание:

pas
```
function Abs(E: Extended): Extended
```
c++
```
float abs(float E)
```

Назначение: Функция возвращает модуль аргумента **E**

9.5.7 Sin

Резюме: Значение синуса

Описание:

pas
```
function Sin(E: Extended): Extended
```
c++
```
float sin(float E)
```

Назначение: Функция возвращает значение синуса угла **E**, значение аргумента в радианах

9.5.8 Cos

Резюме: Значение косинуса

Описание:

pas
```
function Cos(E: Extended): Extended
```
c++
```
float cos(float E)
```

Назначение: Функция возвращает значение косинуса угла **E**, значение аргумента в радианах

9.5.9 Tan

Резюме: Значение тангенса

Описание:

pas
```
function Tan(E: Extended): Extended
```
c++
```
float tan(float E)
```

Назначение: Функция возвращает значение тангенса угла **E**, значение аргумента в радианах

9.5.10 ArcTan

Резюме: Значение арктангенса

Описание:

pas
```
function ArcTan(E: Extended): Extended
```
c++
```
float arctan(float E)
```

Назначение: Функция возвращает значение арктангенса **E**

9.5.11 Exp

Резюме: Значение экспоненты

Описание:

pas
```
function Exp(E: Extended): Extended
```
c++
```
float exp(float E)
```

Назначение: Функция возвращает значение экспоненты аргумента **E**

9.5.12 Ln

Резюме: Натуральный логарифм

Описание:

pas
```
function Ln(E: Extended): Extended
```
c++
```
float ln(float E)
```

Назначение: Функция возвращает значение натурального логарифма аргумента **E**

9.5.13 Pi

Резюме: Число «Пи»

Описание:

pas
```
function Pi: Extended
```
c++
```
float pi(void)
```

Назначение: Функция возвращает значение числа «Пи». Более подробно узнать про число «Пи» можно на сайте: http://ru.wikipedia.org/wiki/%D0%9F%D0%B8

9.6 Функции управления таймером

9.6.1 SetTimer

Резюме: Установить или удалить таймер

Описание:

pas
```
function SetTimer(ProcName: String; Period: TDateTime):
Integer
```
c++
```
int SetTimer(string ProcName; TDateTime Period)
```

Назначение: Функция позволяет установить обработчик периодических событий — таймер. В одной программе может быть организовано произвольное количество таймеров. Аргументы функции:
ProcName — строка с именем процедуры обработчика события таймера;
Period — период таймера, значение в формате **TDatTime**.
Если значение параметра **Period** равно 0, то таймер будет удален.
Обработчиком события таймера может быть любая процедура без параметров.
Одна процедура может служить обработчиком нескольких таймеров.
Система не гарантирует вызов процедур таймера точно через интервал **Period**.
Единственное что можно утверждать — процедура будет вызвана не ранее чем через **Period** после предыдущего вызова. Система не требует удаления организованных таймеров по окончании работы программы.
При удачном выполнении функция **SetTimer** возвращает результат > 0.

Пример:

pas
```
procedure OnTimer1;    // обработчик таймера 1
begin
 Writeln('one second timer');
end;
procedure OnTimer2;    // обработчик таймера 2
begin
// обработчик будет вызван только 1 раз
 Writeln('one minute timer');
 SetTimer('OnTimer2', 0);        // удаление таймера 2
end;

begin
 SetTimer('OnTimer1', OneSecond);// установка таймера 1
 SetTimer('OnTimer2', OneMinute);// установка таймера 2
end.
```
c++
```
void OnTimer1()    // обработчик таймера 1
{
 Writeln("one second timer");
}
void OnTimer2()    // обработчик таймера 2
{
// обработчик будет вызван только 1 раз
 Writeln("one minute timer");
 SetTimer("OnTimer2", 0);         // удаление таймера 2
}

{
 SetTimer("OnTimer1", OneSecond); // установка таймера 1
 SetTimer("OnTimer2", OneSecond); // установка таймера 2
}
```

9.7 Запись в системный журнал

Система содержит ряд функций записи сообщений в системный журнал. Данные функции могут быть использованы при отладке программ.

9.7.1 AddLog

Резюме: Запись в системный журнал

Описание:

pas
```
procedure AddLog(Level: Integer; Message: String)
```
c++
```
void AddLog(int Level, string Message)
```

Назначение: Процедура производит запись текстового сообщения **Message** в системный журнал. Сообщение записывается с уровнем (признаком) **Level**. При указании значения **Level** можно использовать константы описанные в пункте 10.1.

Пример:

pas
```
begin
 AddLog(LOG_INFO, 'message 1');
 AddLog(LOG_ERROR, 'error message');
end;
```

9.7.2 Writeln

Резюме: Запись в системный журнал

Описание:

pas
```
procedure Writeln(S: Variant)
```
c++
```
void Writeln(variant S)
```

Назначение: Процедура преобразует значение аргумента **S** в строку и производит ее запись в системный журнал.
Сообщение записывается с уровнем **LOG_NOTICE**.

Пример:

pas
```
begin
 Writeln('message 1');  // запись строки 'message 1'
 Writeln(1);            // запись строки '1'
end;
```

9.7.3 ShowMessage

Резюме: Запись в системный журнал

Описание:

pas
```
procedure ShowMessage(S: Variant)
```
c++
```
void ShowMessage(variant S)
```

Назначение: Процедура полностью аналогична процедуре **Writeln**

9.8 Функции работы с каналами

9.8.1 AddChannel2

Резюме: Добавление нового канала тип 2

Описание:

pas
```
function AddChannel2(
TagName: String;
Type, Signification: Integer;
Units, Comment: String;
WriteEnable: Boolean): TChannel2;
```
c++
```
TChannel2 AddChannel2(
String TagName,
int Type,
int Signification,
String Units,
String Comment,
bool WriteEnable);
```

Назначение: Функция добавляет новый канал тип 2, результат функции - экземпляр класса **TChannel2**. Если в рамках текущего потока уже существует канал с именем **TagName**, то функция возвращает **Nil** или, если разрешено формирование исключений (см. функцию **EnableException**), то вызывает исключение.

Параметры: **TagName** – имя канала. Имя канала должно быть уникально в рамках всех каналов сервера системы.
Type — тип создаваемого канала. Для указания типа необходимо использовать константы varXXXX, описанные в разделе 10.7.
Signification — код назначения канала:
0 — не определено (используется в общем случае);
1 — интегратор в импульсах;
2 — интегратор в физических единицах;
3 — канал управления;
4 — канал контроля;
5 — охранный извещатель или шлейф;
6 — пожарный извещатель или шлейф;
7 — состояние охраны зоны;
8 — сигнализатор загазованности;
9 — датчик температуры;
10 — извещатель пожарный ручной;
11 — датчик затопления;
12 — лифт.
Units — строка: единицы измерения значения канала.
Comment — строка: текстовый комментарий.
WriteEnable — логический признак возможности записи значения канала сервером системы.
Если значения параметров Units и Comment пустая строка, то значение соответствующего атрибута канала не устанавливается (сохраняется без изменения).

Пример:

```pas
var ch2: TChannel2;
ch2:= AddChannel2('new_channel', varInteger, 0, 'кг',
'масса', False);
```

9.8.2 cq

Резюме: Возвращает значение качества для каналов 1-го или 2-го типа

Описание:

```pas
function cq(Addr: String): Integer;
```

```c++
int cq(String Addr);
```

Назначение: Функция выполняет поиск канала, адрес которого указан в параметре **Addr** и возвращает значение его свойства **State** (см. 10.2). Поиск выполняется для каналов первого и второго типа. Для каналов первого типа адрес должен представлять собой строку в которой числовые компоненты адреса разделены символом точка. Если канал не найден, то функция возвращает значение «-1» (если формирование исключений запрещено, см. функцию **EnableException**) или формирует исключение (если формирование исключений разрешено).
Функция позволяет простым способом получить значение качества канала без получения экземпляра его класса. Следует учесть, что работа с каналами через обращение к свойствам предварительно полученного экземпляра класса является более эффективной.

Параметры: **Addr** – строка, имя канала 1-го или 2-го типа.

9.8.3 cv

Резюме: Возвращает значение для каналов 1-го или 2-го типа

Описание:

pas
```
function cv(Addr: String): Variant;
```
c++
```
Variant cv(String Addr);
```

Назначение: Функция выполняет поиск канала, адрес которого указан в параметре **Addr**. Если канал найден и значение его свойства **State** равно stOk (0), то функция возвращает значение его свойства **Value**. Поиск выполняется для каналов первого и второго типа. Для каналов первого типа адрес должен представлять собой строку в которой числовые компоненты адреса разделены символом точка. Если канал не найден или значение свойства **State** не равно **stOk**, то функция возвращает значение «0» (если формирование исключений запрещено, см. функцию **EnableException**) или формирует исключение (если формирование исключений разрешено).
Функция позволяет простым способом получить значение канала без получения экземпляра его класса. Следует учесть, что работа с каналами через обращение к свойствам предварительно полученного экземпляра класса является более эффективной.

Параметры: **Addr** – строка, имя канала 1-го или 2-го типа.

9.8.4 cvDef

Резюме: Возвращает значение для каналов 1-го или 2-го типа

Описание:

pas
```
function cvDef(Addr: String; DefVal: Variant): Variant;
```
c++
```
Variant cvDef(String Addr, Variant DefVal);
```

Назначение: Назначение функции аналогично назначению описанной выше функции **cv**. Отличие заключается в возможности указать значение, которое будет возвращено функцией при отсутствии указанного канала или в случае, когда значение его свойства **State** не равно **stOk**.
Функция не формирует исключений.

Параметры: **Addr** – строка, имя канала 1-го или 2-го типа.
DefVal – значение возвращаемое по умолчанию.

9.8.5 SendControl

Резюме: Передача команды управления драйверу устройства по имени сформированного им канала.

Описание:

pas
```
function SendControl(Addr: String; Value:
Variant):Boolean;
```
c++
```
bool SendControl(String Addr, Variant Value);
```

Назначение: Функция передает команду управления устройству, являющемуся источником данных для канала с указанным адресом. Адрес может содержать имя канала 1-го или 2-го типа. Для 1-го типа поддерживается передача управления только для каналов с типом данных **dtKey** (**DType** = **dtKey**). Запись значения для каналов других типов игнорируется. Для каналов 2-го типа запись возможна только для каналов с установленным свойством **WriteEnable**.
Записываемое значение зависит от типа оборудования.
При отсутствии ошибок (канал найден и доступен для записи) функция возвращает значение **True**, иначе — **False**.
Следует обратить внимание на то, что запись значения ставит команду управления устройством в очередь, и физически управление будет выполнено через некоторый промежуток времени.

Параметры: **Addr** – строка, имя канала 1-го или 2-го типа.
Value – значение команды управления.

9.9 Прочие

9.9.1 IIF

Резюме: Анализ значения

Описание:

pas
```
function IIF(Cond: Boolean; Res1, Res2: Variant):
Variant
```
c++
```
variant IIF(bool Cond, variant Res1, variant Res2)
```

Назначение: Функция возвращает значение **Res1** если выражение **Cond** истина или выражение **Res2** если выражение **Cond** ложь.

Пример:

pas
```
IIF(true, 'истина', 'ложь'); // результат: 'истина'
IIF(false, 1, 2); // результат: 2
```

9.9.2 EnableException

Резюме: Установка режима формирования исключений

Описание:

pas
```
procedure EnableException(Enable: Boolean)
```
c++
```
void EnableException(bool Enable)
```

Назначение: Процедура включает или выключает режим формирования исключений при обращении к свойствам **Value** и **Values** класса **TChannel**.
В случае если режим формирования исключений включен (**Enable** = **True**), то при обращении к свойству **Value** или **Values** класса **TChannel** в случае если свойство **DType** <> **stOk** будет сформировано исключение. Если режим формирования исключений выключен (**Enable** = **True**), то обращение к свойствам в указанной ситуации приведет к неопределенному результату. По умолчанию режим формирования исключений выключен.

Пример:

pas
```
var res: double;
begin
// в случае, если состояние канала c1 или c2 <> stOk
// при обращении к свойству Value будет сформировано
// исключение
 EnableException(true);
 c1:= ChannelList.Get(1,1,1,1);
 c2:= ChannelList.Get(1,1,1,2);
 try
  res:= c1.value + c2.value;
 except
  Writeln('ошибка: c1 или c2 неисправно');
 end;
end;
```

9.9.3 Inc

Резюме: Инкремент

Описание:

pas
```
procedure Inc(var I: Integer; Incr: Integer = 1)
```
c++
```
void Inc(int &I, int Incr = 1)
```

Назначение: Функция увеличивает значение переменной **I** на величину **Incr** (по умолчанию 1)

9.9.4 Dec

Резюме: Декремент

Описание:

pas
```
procedure Dec(var I: Integer; Decr: Integer = 1)
```
c++
```
void Dec(int &I, int Decr = 1)
```

Назначение: Функция уменьшает значение переменной **I** на величину **Decr** (по умолчанию 1)

9.9.5 RaiseException

Резюме: Генерация исключения

Описание:

pas
```
procedure RaiseException(P: String)
```
c++
```
void RaiseException(string P)
```

Назначение: Функция формирует исключение с параметром **P**. Содержимое **P** из кода скрипта не доступно. В случае если исключение не будет обработано в коде скрипта, его выполнение будет прервано и в системный журнал будет помещено сообщение **P** с уровнем **LOG_ERR**.

Пример:

pas
```
begin
 try
  // ...
  RaiseException('error');
  // ...
 except
  Writeln('exception');
 end;
end;
```

9.9.6 Randomize

Резюме: Инициализация генератора псевдослучайных чисел

Описание:

pas
```
procedure Randomize
```
c++
```
void Randomize(void)
```

Назначение: Инициализация генератора псевдослучайных чисел

9.9.7 Random

Резюме: Формирование псевдослучайного числа

Описание:

pas
```
function Random: Extended
```
c++
```
float Random(void)
```

Назначение: Функция возвращает псевдослучайное число. Значение результата лежит в диапазоне 0 <= **Random** < 1

9.9.8 ValidInt

Резюме: Проверка строки

Описание:

pas
```
function ValidInt(Arg: String): Boolean
```
c++
```
bool ValidInt(string Arg)
```

Назначение: Функция возвращает значение истина, если аргумент **Arg** есть строка с корректной записью значения целого числа

9.9.9 ValidFloat

Резюме: Проверка строки

Описание:

pas
```
function ValidFloat(Arg: String): Boolean
```
c++
```
bool ValidFloat(string Arg)
```

Назначение: Функция возвращает значение истина, если аргумент **Arg** есть строка с корректной записью значения числа с плавающей точкой

9.9.10 ValidDate

Резюме: Проверка строки

Описание:

pas
```
function ValidDate(Arg: String): Boolean
```
c++
```
bool ValidDate(string Arg)
```

Назначение: Функция возвращает значение истина, если аргумент **Arg** есть строка с корректной записью даты

9.9.11 LMServerStatus

Резюме: Проверка статуса соединения с сервером "LanMon"

Описание:

pas
```
function LMServerStatus: Integer
```
c++
```
int LMServerStatus(void)
```

Назначение: Функция возвращает значение 1, если соединение с сервером "LanMon" установлено, 0 — если соединения с сервером не установлено.
Версия управляющей программы для "Windows" всегда возвращает значение 1.

9.9.12 TimeOfSunrise

Резюме: Время восхода солнца для указанной даты и координат

Описание:

pas
```
function TimeOfSunrise(Latitude, Longitude: Double;
ADate, Correction: TDateTime): TDateTime
```
c++
```
TDateTime TimeOfSunrise(float Latitude, float Longitude,
TDateTime Adate, TDateTime Correction)
```

Назначение: Функция возвращает время восхода солнца для указанной даты (**ADate**) и координат (**Latitude**, **Longitude**). Величина **Correction** добавляется к вычисленному значению. Если в указанную дату восхода солнца не было (полярный день или полярная ночь) функция возвращает значение 0.

9.9.13 TimeOfSunset

Резюме: Время заката солнца для указанной даты и координат

Описание:

pas
```
function TimeOfSunset(Latitude, Longitude: Double;
ADate, Correction: TDateTime): TDateTime
```
c++
```
TDateTime TimeOfSunset(float Latitude, float Longitude,
TDateTime Adate, TDateTime Correction)
```

Назначение: Функция возвращает время заката солнца для указанной даты (**ADate**) и координат (**Latitude**, **Longitude**). Величина **Correction** добавляется к вычисленному значению. Если в указанную дату заката солнца не было (полярный день или полярная ночь) функция возвращает значение 0.

9.9.14 Min

Резюме: Возвращает минимальное значение

Описание:

pas
```
function Min(A, B: Variant): Variant
```
c++
```
Variant Min(Variant A, Variant B)
```

Назначение: Функция сравнивает значения параметров A и B и возвращает их минимальное значение. Если значения параметров невозможно сравнить (несовместимые типы), то возвращается значение параметра A.

9.9.15 Max

Резюме: Возвращает максимальное значение

Описание:

```
pas
function Max(A, B: Variant): Variant
c++
Variant Max(Variant A, Variant B)
```

Назначение: Функция сравнивает значения параметров A и B и возвращает их максимальное значение. Если значения параметров невозможно сравнить (несовместимые типы), то возвращается значение параметра A.

10. Предопределенные константы

10.1 Работа с журналом SYSLOG

Значения констант используются при помещении сообщений в системный журнал при помощи функции «**AddLog**».

Имя		**Значение**	**Примечание**
LOG_EMERG	=	0	Опасность
LOG_ALERT	=	1	
LOG_CRIT	=	2	Сообщение о критической ошибке
LOG_ERR	=	3	Сообщение об ошибке
LOG_WARNING	=	4	Предупредительное сообщение
LOG_NOTICE	=	5	Извещение
LOG_INFO	=	6	Информационное сообщение
LOG_DEBUG	=	7	Отладочное сообщение

10.2 Состояние данных

Значения констант используются при анализе значения свойства «**State**» классов **TChannel, TChannel2** и **TDevice**.

Имя		**Значение**	**Примечание**
stOk	=	0	Канал или устройство исправны
stTempOff	=	1	Канал или устройство временно выключены
stOff	=	2	Канал или устройство выключены
stError	=	3	Канал или устройство в состоянии ошибки (устройство не отвечает по информационной сети)
stFailure	=	4	Канал или устройство в состоянии ошибки (нет связи с подключенным прибором)
stWrong	=	5	Некорректные данные или данные вне допустимого диапазона

stNotConnected	=	6	Датчик не подключен

10.3 Тип данных каналов тип 1

Значения констант используются при анализе значения свойства «**DType**» класса «**TChannel**».

Имя		**Значение**	**Примечание**
dtBit	=	1	Тип данных «бит»
dtByte	=	2	Тип данных «байт»
dtShort	=	3	Тип данных «короткое целое» (-128..127)
dtInt	=	4	Тип данных «целое» (-32768..32767)
dtLong	=	5	Тип данных «целое» (-2147483648..2147483647)
dtFloat	=	6	Тип данных «плавающая точка» (1.5 x 10^-45 .. 3.4 x 10^38)
dtString	=	7	Тип данных «строка» (до 16-ти символов)
dtWord	=	8	Тип данных «целое» (0..65535)
dtDouble	=	9	Тип данных «плавающая точка» (5.0 x 10^-324 .. 1.7 x 10^308)
dtTemperature	=	10	Тип данных «целая температура» (-128..127)
dtKD	=	11	Тип данных «контактный датчик»
dtOD	=	12	Тип данных «охранный датчик»
dtDD	=	13	Тип данных «дымовой датчик»
dtFase	=	14	Тип данных «сигнал напряжения»
dtGD	=	15	Тип данных «датчик газа»
dtPump	=	16	Тип данных «насос»
dtCooler	=	17	Тип данных «вентилятор»
dtFan	=	17	Тип данных «вентилятор»
dtKey	=	18	Тип данных «канал управления»
dtLift	=	19	Тип данных «лифт» (устаревший)
dtERU	=	20	Тип данных «уровень жидкости»
dtLocalOhr	=	21	Тип данных «охрана»
dtQual	=	22	Тип данных «качество связи»
dtMOLNIA	=	23	Тип данных «Молния» (устаревший)
dtBDKL	=	24	Тип данных «лифт»
dtBGS	=	25	Тип данных «переговорное устройство»
dtUIR	=	26	Тип данных «извещатель ручной»

10.4 Время

Значения констант используются при работе с временем и датой в формате **TDateTime**.

Имя		Значение	Примечание
OneSecond	=	1/86400	Одна секунда
OneMinute	=	1/1440	Одна минута
OneHour	=	1/24	Один час

10.5 Операции с файлами

Константы используются при работе с классами **TStream** и **TFileStream**.

Имя		Значение	Примечание
fmCreate	=	???[1]	Создание файла
fmOpenWrite	=	???	Открыть файл для записи
fmOpenReadWrite	=	???	Открыть файл для чтения и записи
fmShareExclusive	=	???	Эксклюзивный доступ к файлу
fmShareDenyWrite	=	???	Запрет записи для других
fmShareDenyNone	=	???	Нет запрета записи и чтения для других
soFromBegining	=	???	Смещения начиная с начала
soFromCurrent	=	???	Смещения начиная с текущей позиции
soFromEnd	=	???	Смещения начиная с конца файла

10.6 Конфигурация системы

Значения констант содержат параметры операционной системы, типа процессора и версии управляющей программы.

Имя		Значение	Примечание
OS	=	'WINDOWS' 'LINUX'	Для ОС семейства «MS Windows». Для ОС «LINUX».
CPU	=	x86 ARM ARM EABI	Для процессоров с системой команд «x86». Для процессоров «ARM». Для процессоров «ARM EABI».
VERSION	=	'2.224'	Версия управляющей программы (строка)
nVERSION	=	2.224	Версия управляющей программы (число с плавающей точкой)
Channel2Prefix	=	'2200234'	Префикс, используемый для формирования каналов тип 2 (строка). Обычно совпадает с логином к серверу LM, который, в свою очередь, обычно совпадает с серийным номером блока БКД-ПК.

1 Значение константы зависит от операционной системы

10.7 Тип данных каналов тип 2

Значения констант используются при анализе значения свойства «**Type**» класса **TChannel2**.

Имя		**Значение**	**Примечание**
varEmpty	=	0	Пустое значение
varSmallint	=	2	Целое число -32768..32767
varInteger	=	3	Целое число -2147483648..2147483647
varSingle	=	4	Число с плавающей точкой одинарной точности
varDouble	=	5	Число с плавающей точкой двойной точности
varDate	=	7	Дата и время в формате TDateTime
varBoolean	=	11	Логический тип: True/False
varShortInt	=	16	Целое число -128..127
varByte	=	17	Целое число 0..255
varWord	=	18	Целое число 0..65535
varLongWord	=	19	Целое число 0..255
varString	=	256	Строка символов

11. Предопределенные переменные

Предопределенные переменные позволяют получить доступ к методам и свойствам классов устройств и информационных каналов в ходе выполнения программы.

Имя	**Тип**	**Примечание**
DeviceList	TDeviceList	Объект — список опрашиваемых устройств
ChannelList	TChannelList	Объект — список информационных каналов
Channel2List	TChannel2List	Объект — список информационных каналов тип 2
LocalStorage	TLocalStorage	Объект — хранилище данных «Ключ - Значение»

12. События

12.1 Введение

Система выполнения скриптовой программы содержит средства вызова пользовательских процедур в случае происхождения некоторых событий связанных с ее функционированием.

12.2 Запуск программы

При запуске программы производятся следующие действия:
- инициализация констант и инициализируемых переменных;
- запуск на выполнение главной исполняемой функции.

Выполнение главной исполняемой функции производится только один раз при запуске управляющей программы. Входные и выходные параметры отсутствуют.

12.3 Окончание работы программы

Перед окончанием выполнения управляющей программы, делается попытка вызова процедуры с фиксированным именем **onTerminate**. При отсутствии процедуры с таким именем никаких действий не производится.

Резюме: Событие окончания работы программы

Описание:

pas
```
procedure OnTerminate
```
c++
```
void OnTerminate(void)
```

Назначение: Выполнение действий пользователя связанных с завершением функционирования управляющей программы

Пример:

pas
```
procedure OnTerminate;
begin
// некоторые действия
end;
```
c++
```
void OnTerminate(void)
{
// некоторые действия
}
```

12.4 Изменение состояния канала тип 1

При изменении значения данных или качества любого контролируемого канала тип 1 производится попытка вызова процедуры с фиксированным именем **onChannelChange**. При отсутствии процедуры с таким именем никаких действий не производится.

Резюме: Событие изменения состояния канала

Описание:

pas
```
procedure onChannelChange(Channel: TChannel)
```
c++
```
void onChannelChange(TChannel Channel)
```

Назначение: Выполнение действий пользователя связанных с анализом изменения состояния канала. В качестве параметра процедуре передается указатель на экземпляр класса изменившегося канала.

Пример:

```
pas
procedure onChannelChange(Channel: TChannel);
begin
 If Channel.AddrIs(1,1,1,1) then ...
// некоторые действия
end;
c++
void onChannelChange(TChannel Channel)
{
 if(Channel.AddrIs(1,1,1,1)) { // некоторые действия
 };
}
```

12.5 Изменение даты и времени

Резюме: Событие изменения даты или времени

Описание:

```
pas
procedure onTimeChange(Channel: TChannel)
c++
void onTimeChange(TChannel Channel)
```

Назначение: Выполнение действий пользователя связанных с изменением системной даты или времени. В качестве параметра процедуре величина изменения времени (разница между новым и старым значением системного времени) в формате **TDateTime**. Вызов процедуры производится в следующих случаях:
- для ОС Windows: изменение системного времени по команде оператора или выполнения процедуры синхронизации с источником точного времени;
- для ОС Linux: синхронизация времени управляющей программы с сервером «LanMon».
При изменении системного времени оператором системы для ОС Linux вызова события не производится.

Пример:

```
pas
procedure onTimeChange(Delta: TDateTime);
begin
 // некоторые действия
end;
c++
void onTimeChange(TDateTime Delta)
{
 // некоторые действия
}
```

12.6 Событие таймера

Система выполнения скриптовой программы позволяет программисту организовать произвольное количество периодически вызываемых процедур — таймеров. Для создания и удаления таймера служит системный вызов **SetTimer**. В качестве его параметра указывается имя процедуры, выполняющей обработку события. Системный вызов **SetTimer** описан в пункте 9.6.1.

Резюме: Процедура обработки события таймера

Описание:

```
pas
procedure SomeProcName
c++
void SomeProcName(void)
```

Назначение: Процедура обработки события таймера. Входные и выходные параметры отсутствуют. Одна процедура может служить обработчиком событий нескольких таймеров.

Пример:

```
pas
procedure Timer1;
begin
 // некоторые действия
end;
procedure Timer2;
begin
 // некоторые действия
end;
begin
 SetTimer('Timer1', OneSecond);
 SetTimer('Timer2', OneSecond);
end.
c++
void Timer1()
{
 // некоторые действия
}
void Timer2()
{
 // некоторые действия
}
{
 SetTimer("Timer1", OneSecond);
 SetTimer("Timer2", OneSecond);
}
```

12.7 Получение команды управления для канала тип 2

При получении команды управления от сервера для канала тип 2 (изменения состояния канала со стороны сервера) производится попытка вызова процедуры с фиксированным именем **onChannel2Control**. При отсутствии процедуры с таким именем никаких действий не производится. Вызов процедуры производится до передачи команды соответствующему драйверу. Если процедура обработки установит свойству **Changed** канала значение **False**, то команда драйверу устройства передана не будет.
Следует учесть, что получение команды ещё не означает изменение значения самого канала. Для того, что-бы канал изменил своё значение необходимо его явно присвоить. Для каналов, созданных драйверами опроса оборудования это обычно производится автоматически, для каналов созданных в скрипте вызовом метода **AddChannel2** это необходимо выполнить

явно.

Резюме: Событие получения команды управления

Описание:

```
pas
procedure onChannel2Control(Channel: TChannel2)
c++
void onChannel2Control(TChannel2 Channel)
```

Назначение: Выполнение действий пользователя связанных с анализом изменения состояния канала. В качестве параметра процедуре передается указатель на экземпляр класса изменившегося канала.

Пример:

```
pas
procedure onChannel2Control(Channel: TChannel2);
begin
 If Channel.AddrIs('system123') then begin
   // некоторые действия
   // ...
   // присваиваем значение каналу
   Channel.Value:= Channel.Value;
 end;
end;
c++
void onChannel2Control(TChannel2 Channel)
{
 if(Channel.AddrIs("system123")) {
  // некоторые действия
  // ...
  // присваиваем значение каналу
  Channel.Value = Channel.Value;
 };
}
```

12.8 Изменение состояния канала тип 2

При изменении значения данных или качества любого контролируемого канала тип 2 производится попытка вызова процедуры с фиксированным именем **onChannel2Change**. При отсутствии процедуры с таким именем никаких действий не производится.
Вызов процедуры производится до передачи значения канала на сервер системы. Если процедура обработки установит свойству **Changed** канала значение **False**, то передача нового значения канала на сервер выполнена не будет.

Резюме: Событие изменения состояния канала

Описание:

```
pas
procedure onChannel2Change(Channel: TChannel2)
c++
void onChannel2Change(TChannel2 Channel)
```

Назначение: Выполнение действий пользователя связанных с анализом изменения состояния канала. В качестве параметра процедуре передается указатель на экземпляр класса изменившегося канала.

Пример:

```
pas
procedure onChannel2Change(Channel: TChannel2);
begin
 If Channel.AddrIs('system123') then ...
// некоторые действия
end;
c++
void onChannel2Change(TChannel2 Channel)
{
 if(Channel.AddrIs("system32")) { // некоторые действия
 };
}
```

13. Примеры применения

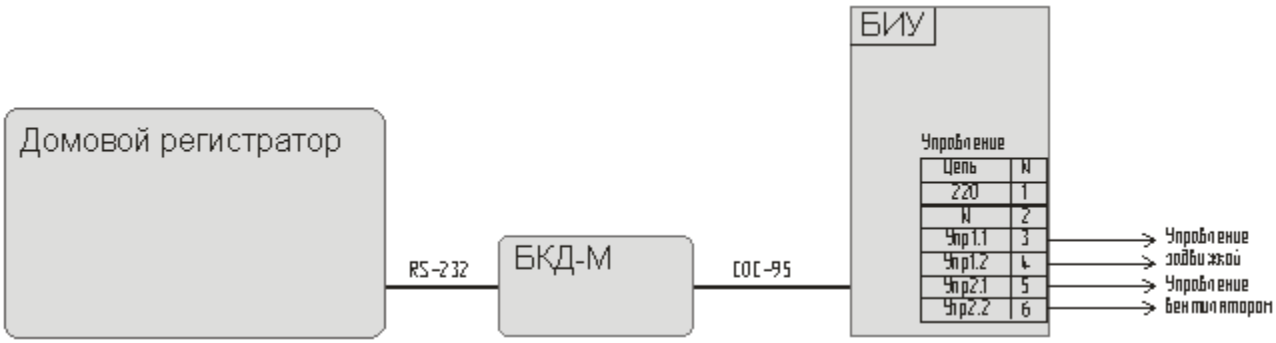
Исходные тексты приведенных в примерах файлов конфигурации входят в состав дистрибутива управляющей программы для ОС Windows (Драйвер OprosLib.DLL).

13.1 Управление системой вентиляции

13.1.1 Описание

Необходимо организовать автоматическое управление вентиляцией производственного помещения. Воздух в помещение подается через воздуховод с управляемой задвижкой и приточным вентилятором. При температуре воздуха меньше или равной 20 °С задвижка должна быть закрыта, вентилятор выключен. При температуре больше 20 °С задвижка должна быть открыта, при температуре больше 25 °С должен быть включен вентилятор.
Решение: управление вентилятором, задвижкой и измерение температуры производится при помощи блока БИУ. Данные блока анализируются программой, которая на основе значения температуры производит управление в соответствии с описанным выше алгоритмом. При написании скрипта используется язык «PascalScript».

13.1.2 Схема подключения оборудования



13.1.3 Пример конфигурации

```
; Файл demo4.ini
;
; Пример программы автоматического управления на скриптовом языке.
;
; Постановка задачи.
; Необходимо организовать автоматическое управление вентиляцией
; производственного помещения. Воздух в помещение подается через
; воздуховод с управляемой задвижкой и приточным вентилятором.
; При температуре воздуха <= 20С задвижка должна быть закрыта, вентилятор
; выключен. При температуре > 20С задвижка должна быть открыта, при
; температуре > 25С должен быть включен вентилятор.
; Решение.
; Управление вентилятором, задвижкой и измерение температуры производится
; при помощи блока БИУ. Данные блока анализируются программой, которая
; на основе значения температуры производит управление в соответствии с
; установленным алгоритмом.
;
; файл defines.inc содержит описания стандартных типов данных

<INCLUDE defines.inc>
;
[SOS]
; тут описание устройств сети СОС-95

#BIU:1
>1.1.1.1: dtTemperature, noUse         ; канал измерения температуры
>1.1.1.5: dtKey, 0                     ; канал управления задвижкой
>1.1.1.6: dtKey, 1                     ; канал управления вентилятором

[CODE:PascalScript]

var Valve, Fan: TChannel;

// обработчик события изменения канала
procedure OnChannelChange(C: TChannel);
begin
 try
  if C.AddrIs(1,1,1,1) then begin      // если это канал температуры (1.1.1.1)
   if C.Value <= 20 then begin         // управление в соответствии с алгоритмом
    Valve.Control:= 0;
    Fan.Control:= 0;
   end else begin
    Valve.Control:= 1;
    If C.Value > 25 then Fan.Control:= 1 else Fan.Control:= 0;
   end;
  end;
 except
  // исключение в случае неисправности оборудования, даем команду
  // включить вентилятор и закрыть задвижку
  Valve.Control:= 0;
  Fan.Control:= 0;
 end;
end;

// запуск программы
begin
 EnableException(True);              // разрешение формирования исключений
 Valve:= ChannelList.Get(1,1,1,5);   // получение класса канала управления задвижкой
 Fan:= ChannelList.Get(1,1,1,6);     // получение класса канала управления вентилятором
end.
```

13.2 Управление светофором

13.2.1 Описание

Необходимо организовать автоматическое управление автомобильным светофором (три сигнала: красный, желтый, зеленый). Переключение осуществлять в последовательности: красный (10 сек), зеленый (10 сек), желтый (2 секунды), затем последовательность повторяется. В ночное время (с 22:00 до 6:00) светофор должен работать в режиме желтого мигающего сигнала. Решение: управление включением сигналов светофора производится при помощи двух блоков БИУ, один блок управляет красным и желтым сигналом, второй - зеленым. Переключение сигналов производится по сигналу программного таймера. При написании скрипта используется язык «PascalScript».

13.2.2 Пример конфигурации

```
; Файл demo5.ini
;
; Пример программы автоматического управления на скриптовом языке.
; Тема: управление автомобильным светофором
;
; Постановка задачи.
; Необходимо организовать автоматическое управление автомобильным
; светофором (три сигнала: красный, желтый, зеленый).
; Переключение осуществлять в последовательности: красный (10 сек),
; зеленый (10 сек), желтый (2 секунды), затем последовательность
; повторяется. В ночное время (с 22:00 до 6:00) светофор должен
; работать в режиме желтого мигающего сигнала.
; Решение.
; Управление включением сигналов светофора производится при помощи
; двух блоков БИУ, один блок управляет красным и желтым сигналом,
; второй - зеленым. Переключение сигналов производится по сигналу
; программного таймера.
;
; файл defines.inc содержит описания стандартных типов данных

<INCLUDE defines.inc>
;
[SOS]

#BIU:1
>1.1.1.1: dtKey, 0                      ; управление красным
>1.1.1.2: dtKey, 1                      ; управление желтым

#BIU:2
>1.1.1.3: dtKey, 0                      ; управление зеленым


[CODE:PascalScript]

var Red, Yellow, Green: TChannel;
    Stage: Integer;

function Night: Boolean;                // возврвщет True если ночь
var Hour, Min, Sec, MSec: Word;
begin
 DecodeTime(Time, Hour, Min, Sec, MSec);
 result:= (Hour>=22) or (Hour<6);
end;

// обработчик события таймера
procedure TimerProc;
```

```
begin
 try
  Case Stage of
   0: begin // горение красного
    Red.Control:= 1;
    Yellow.Control:= 0;
    Green.Control:= 0;
    If not Night then Stage:= 2 else Stage:= 3;
    SetTimer('TimerProc', OneSecond*10);   // красный горит 10 сек
    Writeln('включен красный');
   end;
   1: begin // горение желтого
    Red.Control:= 0;
    Yellow.Control:= 1;
    Green.Control:= 0;
    If not Night then Stage:= 0 else Stage:= 3;
    SetTimer('TimerProc', OneSecond*2);   // желтый горит 2 сек
    Writeln('включен желтый');
   end;
   2: begin // горение зеленого
    Red.Control:= 0;
    Yellow.Control:= 0;
    Green.Control:= 1;
    If not Night then Stage:= 1 else Stage:= 3;
    SetTimer('TimerProc', OneSecond*10);   // зеленый горит 10 сек
    Writeln('включен зеленый');
   end;
   3: begin // мигание желтого
    Red.Control:= 0;
    Green.Control:= 0;
    If Yellow.Control = 0 then Yellow.Control:= 1 else Yellow.Control:= 0;
    SetTimer('TimerProc', OneSecond);      // период мигания 1 секунда
    If not Night then Stage:= 0;
   end;
  end;
 except
  // исключение в случае неисправности оборудования, даем команду выключить все
  Red.Control:= 0;
  Yellow.Control:= 0;
  Green.Control:= 0;
 end;
end;

// запуск программы
begin
 EnableException(True);              // разрешение формирования исключений
 Red:= ChannelList.Get(1,1,1,1);     // получение классов каналов управления лампами
 Yellow:= ChannelList.Get(1,1,1,2);
 Green:= ChannelList.Get(1,1,1,3);
 Stage:= 0;
 SetTimer('TimerProc', OneSecond);   // установка таймера
end.
```

13.3 Управление насосной станцией

13.3.1 Описание

Необходимо организовать автоматическое управление насосной станцией (насосы холодного водоснабжения) центрального теплового пункта (ЦТП) состоящей из двух взаиморезервируемых насосов. Давление воды на выходе насосной станции измеряется при помощи измерительного преобразователя давления подключенного к тепловычислителю "ВИСТ". Кроме того, тепловычислитель используется для учета потребления тепла

потребителями ЦТП.
Управление насосами должно производится по следующему алгоритму:
- в случае нормального функционирования насосов должно производится их попеременное переключение с целью достижения равномерного износа оборудования с периодом один раз в час;
- в случае неисправности насоса (давление на выходе < 0.5 бар) должно произойти переключение на другой (резервный) насос.
Решение: управление включением насосов производится при помощи блока БИУ. Контроль давления на выходе насосной производится путем чтения данных из тепловычислителя "ВИСТ", подключенного непосредственно к последовательному порту COM2 компьютера.
При написании скрипта используется язык «C++Script».

13.3.2 Пример конфигурации

```
; Файл demo6.ini
;
; Пример программы автоматического управления на скриптовом языке.
; Тема: управление резервируемыми насосами
;
; Постановка задачи.
; Необходимо организовать автоматическое управление насосной
; станцией (насосы холодного водоснабжения) состоящей из
; двух взаиморезервирующих насосов. Давление воды на выходе насосной
; станции измеряется при помощи измерительного преобразователя давления
; подключенного к тепловычислителю "ВИСТ".
; Управление насосами должно производится по следующему алгоритму:
; - в случае нормального функционирования насосов должно производится
; их попеременное переключение с целью достижения равномерного износа
; с периодом один раз в час;
; - в случае неисправности насоса (давление на выходе < 0.5 бар) должно
; произойти переключение на другой насос.
; Решение.
; Управление включением насосов производится при помощи блока БИУ.
; Контрль давления на выходе насосной производится путем чтения данных из
; тепловычислителя "ВИСТ", подключенного непосредственно к последовательному
; порту компьютера.
; Текст программы может быть использован только в целях обучения, при
; разработке реальных программ необходимо учитывать дополнительные условия,
; такие как время установления выходного давления при переключении насосов,
; вывод насосов из эксплуатации и прочее.
;
; файл defines.inc содержит описания стандартных типов данных
<INCLUDE defines.inc>
;
[SOS]
; описание устройств сети СОС-95

#BIU:255
>1.1.1.1: dtKey, 0                      ; управление насосом 1
>1.1.1.2: dtKey, 1                      ; управление насосом 2

[DIRECT]
; описание устройств подключаемых напрямую
#VIST
PORT="COM2" ; теплосчетчик подключен к порту COM2
; перечень информационных каналов, читаемых из теплосчетчика ВИСТ
>1.1.1.5: dtDouble, 52[vistVolPodCh]         ;ХВС, расход, м3/ч
>1.1.1.6: dtDouble, 52[vistPressPodCh]       ;ХВС, давление, Атм
>1.1.1.7: dtDouble, 52[vistVolPod]           ;ХВС, объем, м3

[CODE:C++Script]
```

```
TChannel pump1, pump2;
word curhour, curpump;

void SwithPump()
{
 if(curpump==0) {
  curpump=1;
  pump1.control=0;
  pump2.control=1;
 } else {
  curpump=0;
  pump1.control=1;
  pump2.control=0;
 }
}
// обработчик события таймера
void TimerProc()
{
 word hour, min, sec, msec;
// проверка смены значения часа, если изменился час, переключаем насо
 DecodeTime(time(), hour, min, sec, msec);
 if(curhour!=hour) {SwithPump(); curhour=hour;}
}
// обработчик изменения значения канала
void OnChannelChange(TChannel C)
{
 try {
  if(C.AddrIs(1,1,1,6)) {
   if(C.Value<0.5) SwithPump();      // переключение насоса если давление < 0.5
  }
 }
 except {
  // если датчик давления неисправен, то ничего не делаем
  Writeln("vist failure");
 }
}
// запуск программы
{
 word hour, min, sec, msec;

 EnableException(True);                // разрешение формирования исключений
 pump1=ChannelList.Get(1,1,1,1);       // насос 1
 pump2=ChannelList.Get(1,1,1,2);       // насос 2
 DecodeTime(time(), hour, min, sec, msec);
 curhour=hour;
 SetTimer("TimerProc", OneSecond*60);  // установка таймера 1 минута
 // начинаем с насоса 1
 curpump=0;
 pump1.control=1;
 pump2.control=0;
}
```

13.4 Изменение настроек регулятора САТ-500 в зависимости от расхода теплоносителя (каналы тип 2)

13.4.1 Описание

На объекте установлены теплосчётчик «ВИС.Т» и программируемый контроллер «САТ-500». Необходимо организовать передачу значения измеренного теплосчётчиком расхода теплоносителя в точку регулирования контроллера «САТ-500». Тепловычислитель «ВИС.Т»

с серийным номером 12285 подключен через последовательный порт RS-232, контроллер «CAT-500» с серийным номером 2202080 подключен через Ethernet по протоколу Modbus/TCP. Для чтения данных обоих приборов используются каналы тип 2. Канал измерения расхода, формируемый драйвером опроса тепловычислителя имеет название «vist_12285_tv1.Gm1», канал точки регулирования контроллера «CAT-500» имеет название «CAT500_2202080_SP0».
При написании скрипта используется язык «C++Script».

13.4.2 Пример конфигурации

```
; Файл demo10.ini
;
; Пример программы автоматического управления на скриптовом языке.
; Тема: Изменение настроек регулятора CAT-500 в зависимости от расхода теплоносителя
;
; Постановка задачи.
; На объекте установлены теплосчётчик «ВИС.Т» и программируемый контроллер «CAT-500».
; Необходимо организовать передачу значения измеренного теплосчётчиком расхода
; теплоносителя в точку регулирования контроллера «CAT-500».
; Тепловычислитель «ВИС.Т» с серийным номером 12285 подключен через
; последовательный порт RS-232, контроллер «CAT-500» с серийным номером 2202080
; подключен через Ethernet по протоколу Modbus/TCP.
; Для чтения данных обоих приборов используются каналы тип 2.
; Канал измерения расхода, формируемый драйвером опроса тепловычислителя
; имеет название «vist_12285_tv1.Gm1», канал точки регулирования
; контроллера «CAT-500» имеет название «CAT500_2202080_SP0».
;
[DIRECT]

#CAT
AUTO=1
USETCP=1
PORT="TCP:192.168.1.238"
ADDR0=48

#VISTM
PORT="/dev/ttyS0"
AUTO=1
PERIOD=0.2

[CODE:C++Script]

// канал точки регулирования CAT-500
TChannel2 cat500;

void onChannel2Change(TChannel2 channel)
{
  if(channel.id == "vist_12285_tv1.Gm1") {
    // изменилось значение расхода теплоносителя
    if(channel.state == stOk) {
      // получены корректные данные расхода
      // проверяем наличие канала управления контроллером
      if(cat500 == Nil) {
        // канал пока не инициализирован, пробуем его получить из списка
        cat500 = Channel2List.get("CAT500_2202080_SP0");
      }
      if(cat500 != Nil) {
        // канал инициализирован, можно присваивать значение
        cat500.Control =  channel.Value;
      } else {
        // канал управления cat500 не найден, нет связи с контроллером или он ещё
        // опрошен
```

```
      }
    } else {
      // тепловычислитель неисправен или неисправность канала измерения расхода
    }
  }
}

{
  // инициализация канала САТ-500 (пока его ещё нет)
  cat500 = Nil;
}
```

13.5 Автоматический терморегулятор (каналы тип 2)

13.5.1 Описание

На объекте установлены блоки ККД-С (адрес 3) и БИУ (адрес 4). Канал управления БИУ выполняет управление включением электрического нагревателя. Температура на объекте измеряется датчиком температуры, подключенным к блоку ККД-С.
Программа должна выполнять автоматическое включение нагревателя, когда измеренное значение температуры ниже значения заданной уставки и выключать нагреватель, когда температура выше значения уставки. Для избежания лишних включений и выключений нагревателя программа должна предусматривать сравнение с учётом гистерезиса. Значение уставки должно устанавливаться через создаваемый канал управления. Установленное значение должно сохраняться в памяти устройства, в том числе после перезагрузки устройства.
При написании скрипта используется язык «C++Script».

13.5.2 Пример конфигурации

```
; Файл demo11.ini
;
; Пример программы автоматического управления электрическим нагревателем.
;
; Постановка задачи:
; На объекте установлены блоки ККД-С (адрес 3) и БИУ (адрес 4). Канал управления БИУ
; выполняет управление включением электрического нагревателя.
; Температура на объекте измеряется датчиком температуры, подключенным к блоку ККД-С.
; Программа должна выполнять автоматическое включение нагревателя, когда измеренное
; значение температуры ниже значения заданной уставки и выключать
; нагреватель, когда температура выше значения уставки. Для избежания лишних включений и
; выключений нагревателя программа должна предусматривать сравнение с учётом
; гистерезиса.
; Значение уставки должно устанавливаться через создаваемый канал управления.
; Установленное значение должно сохраняться в памяти устройства, в том
; числе после перезагрузки устройства.
;

[DIRECT]

#SYS
; системные параметры БКД-ПК
AUTO=1

[SOS]

#KKDS:3
```

```
AUTO=1

#BIU:4
AUTO=1


[CODE:C++Script]
// канал уставки температуры
TChannel2 tempSetPointChannel;
// значение уставки температуры
double tempSetPoint;
// канал измерения температуры
TChannel2 tempChannel;
// канал управления нагревателем
TChannel2 controlChannel;

// обработчик получения нового значения уставки температуры от сервера
void onChannel2Control(TChannel2 channel) {
  if(channel.AddrIs(tempSetPointChannel.Addr)) {
    // новое значение уставки
    tempSetPoint = channel.value;
    // сохраняю значение в хранилище
    LocalStorage.SetItem("temperatureSetPoint", tempSetPoint);
    // пишу об этом в лог
    AddLog(LOG_NOTICE, "new value of " + tempSetPointChannel.Addr + " is " +
FloatToStr(tempSetPoint));
    channel.value = channel.value;
  }
}

// процедура таймера, вызывается раз в 10 секунд
void timer() {
  // проверяю наличие канала измерения температуры
  if(tempChannel == Nil) {
    tempChannel = Channel2List.get(Channel2Prefix + "_kkds_A3_temperature");
    if(tempChannel == Nil) return;
  }
  // проверяю наличие канала управления нагревателем
  if(controlChannel == Nil) {
    controlChannel = Channel2List.get(Channel2Prefix + "_biu_A4_do1");
    if(controlChannel == Nil) return;
  }
  // проверяю статус канала управления
  if(controlChannel.Quality != stOk) return;
  // проверяю статус канала температуры
  if(tempChannel.Quality == stOk) {
    // управление с гистерезисом
    if(tempChannel.Value > tempSetPoint + 1) {
      // выключение нагревателя
      if(controlChannel.Control != 0) {
        AddLog(LOG_NOTICE, "switch heater OFF");
        controlChannel.Control = 0;
      }
    } else
    if(tempChannel.Value < tempSetPoint - 1) {
      // включение нагревателя
      if(controlChannel.Control != 1) {
        AddLog(LOG_NOTICE, "switch heater ON");
        controlChannel.Control = 1;
      }
    }
  } else {
    // канал измерения температуры неисправен, включаем нагреватель
    AddLog(LOG_WARNING, "temperature sensor failure");
    if(controlChannel.Control != 1) {
```

```
      AddLog(LOG_NOTICE, "switch heater ON");
      controlChannel.Control = 1;
    }
  }
}
//
// выполняется при запуске
{
  // создаю канал уставки температуры
  tempSetPointChannel = AddChannel2(Channel2Prefix + "_temperatureSetPoint", varDouble,
0, "C", "уставка температуры", true);
  tempSetPointChannel.Active = true;
  // читаю значение температуры уставки из хранилища
  tempSetPoint = LocalStorage.AsDouble("temperatureSetPoint", 10);
  tempSetPointChannel.value = tempSetPoint;
  // устанавливаю таймер
  SetTimer("timer", OneSecond*10);
}
```