

## **Описание языка бейсик "LanMon"**

Ревизия 3 от 14.07.2005 г.

# Содержание

Содержание.....	2
Описание языка LanMon бейсик .....	3
Переменные .....	3
Числовые и строковые константы .....	3
Представление времени.....	4
Объекты .....	4
Объект «Звуковая подсистема» .....	4
Объект «Карта» .....	4
Объект «Элемент карты» .....	5
Объект «Канал LanMon» .....	6
Объект «Группа каналов» .....	6
Системные переменные.....	7
Операторы.....	7
& .....	7
.....	8
AND .....	8
OR .....	8
+ .....	8
- .....	8
* .....	9
/ .....	9
% .....	9
^ .....	9
Унарный – .....	10
NOT .....	10
; .....	10
= .....	10
Ключевые слова .....	11
IF ... THEN ... ELSE ... ENDIF .....	11
FOR ... TO ... STEP ... NEXT .....	11
WHILE ... WEND.....	12
REM .....	13
END .....	13
GOTO.....	13
RUN ... FROM .....	13
PRINT .....	14
Встроенные функции.....	15
Функции преобразования типов .....	15
Функции общего назначения .....	15
Функции, специфичные для LanMon .....	16
Функции для работы с отчетами RTF .....	18
Функции для работы с генератором отчетов .....	18
Функции для работы с группами каналов .....	20
Функции для работы со встроенным клиентом IP телефонии H323 .....	21
Ошибки, возвращаемые <i>H323LastError()</i> .....	21
Работа со трендами (графиками).....	22
Работа со временем .....	22
Работа со строками .....	22
Математические функции .....	23

## Описание языка LanMon бейсик

LanMon бейсика (в дальнейшем - бейсик) является встроенным языком программирования системы LanMon. Он предназначен для вычислений, вызова встроенных функций LanMon и манипулирования свойствами объектов системы LanMon (каналами, картами, значками, сводками, текстом и пр.). Синтаксис бейсика близок к стандартному синтаксису языка бейсик. Бейсик является интерпретатором (каждый раз при выполнении программы производится синтаксический разбор текста программы). Все элементы языка не чувствительны к регистру символов.

### Переменные

Имена переменных:

- должны начинаться с букв латинского алфавита A...Z или a...z
- не должны быть длиннее 64 символов

Максимально допустимое количество переменных – 200.

Переменные могут содержать значения одного из следующих типов данных:

- строка переменной длины, но не более 300 символов (string)
- число с плавающей точкой 8-ми байтовое (floating)
- целое число со знаком 4-х байтовое (integer)

Переменные не нуждаются в описании. Они создаются при первом упоминании в тексте программы и никогда не удаляются. Все переменные глобальны для программы LanMon. Переменные могут менять тип своего значения при различных операциях. Проверить текущий тип значения переменной можно при помощи встроенных функций IsInteger, IsFloating, IsString. С любыми типами переменных можно проделывать любые определенные операции, при этом произойдет неявное преобразование типа. Явно преобразовать тип значения переменной можно функциями AsInteger, AsFloating, AsString.

### Числовые и строковые константы

Целые знаковые константы типа integer задаются так:

- **Десятичные** числа – содержат символы от '0' до '9'.
- **Шестнадцатеричные** числа - записываются с символом 'H' (HEX) в конце и содержат символы от '0' до '9' и буквы от 'A' до 'F'. Допускается записывать без символа 'H' в конце, но в этом случае число должно начинаться с префикса '0x' или '0X'.
- **Восьмеричные** числа - записываются с символом 'O' (OCTAL) в конце и содержат символы от '0' до '7'.
- **Двоичные** числа - записываются с символом 'B' (BINARY) в конце и содержат символы '0' и '1'.

Константы с плавающей точкой типа floating содержат целую и дробную часть, разделенные точкой.

Строковые константы задаются в двойных кавычках.

## Представление времени

В LanMon время представлено в виде числа типа floating. Целая часть числа это количество дней, прошедших с 30.12.1899. Дробная часть числа это время. Например:

Значение	Пояснение
0	30.12.1899 12:00
2.75	1.1.1900 18:00
-1.25	12.29.1899 6:00
35065	1.1.1996 12:00

Все функции работы со временем работают с таким представлением.

Время можно вычитать и складывать как обычные числа floating. Например:

REM К текущему времени прибавить 1 секунду и результат поместить в переменную t  
 $t = \text{time}() + (1.0/86400.0)$

REM К текущему времени прибавить 1 минуту и результат поместить в переменную t  
 $t = \text{time}() + (1440.0/86400.0)$

REM От текущего времени отнять 1 сутки и результат поместить в переменную t  
 $t = \text{time}() - 1.0$

## Объекты

В LanMon есть несколько типов объектов. Объекты нельзя создать динамически. Объект имеет свойства разного типа. Доступ к свойству объекта может быть как на чтение (Ч) так и на запись (З).

### Объект «Звуковая подсистема»

Предоставляет доступ к звуковой подсистеме LanMon. Это системный объект он доступен по имени: @Sound

Свойство	Доступ	Значение
BUSY	Ч	0-сейчас тишина 1-сейчас что-либо проигрывается
STOP	З	При записи значения отличного от нуля очищается очередь проигрывания и проигрывание останавливается.
FILE	З	При записи имени звукового файла он ставится в очередь на проигрывание. Причем если файл задан без пути – он будет искаться в директории для звуков ..\LanMon\Wav.

### Объект «Карта»

Чтобы карта стала доступна, в параметрах карты задайте имя объекта карты. Все свойства карты будут доступны через это имя.

Свойство	Доступ	Значение
LEFT	ЧЗ	Координата левого верхнего угла карты по оси X
TOP	ЧЗ	Координата левого верхнего угла карты по оси Y
WIDTH	ЧЗ	Ширина карты, в экранных точках
HEIGHT	ЧЗ	Высота карты, в экранных точках

OBJCOUNT	Ч	Текущее кол-во элементов карты (графических объектов)
OHRANA	ЧЗ	Состояние охраны карты (0-снята 1-на охране)
TREVOGA	Ч	Состояние карты: 0-все спокойно 1-есть хоть 1 сработка на карте 2-есть хоть 1 тревога на карте
VISIBLE	ЧЗ	Состояние видимости карты (0-спрятана 1-видна)
NAME	ЧЗ	Название карты
BITMAP	ЧЗ	Путь к файлу подложки карты
SOUND	ЧЗ	WAV файл, назначенный данной карте в параметрах карты
A1,A2,A3,A4	Ч	Адрес канала, ассоциированного с данной картой

### Объект «Элемент карты»

Элемент карты это графический объект на карте. Каждый графический объект имеет следующие свойства:

Свойство	Доступ	Значение
A1,A2,A3,A4	Ч	Адрес канала LanMon, ассоциированного с данным объектом.
STATE	ЧЗ	Качество (состояние) канала LanMon. Расшифровку смотрите в документе <a href="#">manual.doc</a>
VALUE	ЧЗ	Значение канала LanMon. Тип значения зависит от типа канала.*
TIME	ЧЗ	Время последнего изменения состояния или значения канала LanMon. Время представлено в формате 8 байтового числа с плавающей точкой.
DTYPE	ЧЗ	Тип канала LanMon.*
ID_SYSM	Ч	Подсистема, к которой относится данный канал. Все подсистемы перечислены в файле ID_SYSM.txt в директории LanMon.
TREVOGA	Ч	Состояние объекта: 0-норма 1-сработка 2-тревога (сработка и тревога появляются только если у данного объекта настроены алармы)
ObjType	Ч	Тип объекта: 1. набор картинок 2. текст 3. фигура 4. картинка 5. температура воздуха 6. подпись к набору картинок 7. ... 8. график 9. прогресс бар
Text	ЧЗ	Текст объекта. У некоторых объектов он выводится на экран, у других это всплывающая подсказка.
Selected	ЧЗ	Данный объект выделен ?
X	ЧЗ	Координата левого верхнего угла объекта по оси X
Y	ЧЗ	Координата левого верхнего угла объекта по оси Y
WIDTH	ЧЗ	Ширина объекта, в экранных точках
HEIGHT	ЧЗ	Высота объекта, в экранных точках
VISIBLE	ЧЗ	Состояние видимости объекта (0-спрятан 1-виден)
SOUND	Ч	Звуковой файл, назначенный данному объекту

\* примечание: расшифровку этих значений смотрите в документе, посвященном типам данных LanMon.

## Объект «Канал LanMon»

Пример работы со списком каналов приведен в примере «rtvars.bas»

Свойство	Доступ	Значение
A1,A2,A3,A4	Ч	Адрес канала LanMon, ассоциированного с данным объектом.
STATE	ЧЗ	Качество (состояние) канала LanMon. Расшифровку смотрите в документе <a href="#">manual.doc</a>
VALUE	ЧЗ	Значение канала LanMon. Тип значения зависит от типа канала.*
TIME	ЧЗ	Время последнего изменения состояния или значения канала LanMon. Время представлено в формате 8 байтового числа с плавающей точкой.
DTYPE	Ч	Тип канала LanMon.*
ID_SYSM	Ч	Подсистема, к которой относится данный канал. Все подсистемы перечислены в файле ID_SYSM.txt в директории LanMon.
ID	Ч	Идентификатор источника данных. Используется в больших системах с сервером и несколькими опросчиками.
NAME	Ч	Название канала LanMon.
CHANGE	Ч	Если значение $\neq 0$ (true) – значит, произошло изменение свойств STATE или VALUE. Источником такого изменения может быть сервер, драйвер оборудования или изменение соответствующего свойства из бейсика. Чтение свойства CHANGE сбрасывает его в 0 (false). Таким образом, к этому свойству можно обращаться только из одного места. Типовое использование этого свойства – в выражениях событийного тренда.

## Объект «Группа каналов»

Пример работы с группой каналов приведен в примере «groups.bas». Объект группы каналов доступен через вызов функции grpByIndex(i), где i – индекс группы с нуля. Например: grpByIndex(0).ohrana – состояние охраны первой группы каналов. Альтернативный способ доступа к каналам это функции RtVarByAddr(A1,A2,A3,A4) и RtVarByIndex(i) (см. описание функций). Пример работы с группами каналов смотрите в файла \SAMPLES\groups.bas

Свойство	Доступ	Значение
OHRANA	ЧЗ	Состояние охраны группы каналов. Логическое значение: 0-снят с охраны, 1-на охране
STATE	Ч	Целое. Состояние группы: 0-все спокойно 1-есть сработка хоть у 1 канала в группе 2-есть тревога хоть у 1 канала в группе.
NEISPRAY	Ч	Логическое значение. Есть хоть один канал в группе неисправен ? (0/1). Канал считается неисправным, когда его качество STATE>2.
NAME	Ч	Строка. Название группы, заданное при ее создании в редакторе групп каналов.

## Системные переменные

Если имя переменной начинается с символа '@', то такая переменная является системной, т.е. ее назначение предопределено. Список всех системных переменных приведен в следующей таблице.

Имя	Тип	Значение
@TickCount	integer	Число миллисекунд, прошедшее с момента запуска Windows
@Hour	integer	Часы текущего времени (0-23)
@Min	integer	Минуты текущего времени (0-59)
@Sec	integer	Секунды текущего времени (0-59)
@Year	integer	Текущий год полным числом
@Month	integer	Текущий месяц (1-12)
@Day	integer	Текущий день (1-31)
@Dow	integer	Текущий день недели Воскресенье = 0, Понедельник = 1 и т.д.
@Map	объект	Объект «карта»
@MapCnt	integer	Количество карт на объекте
@MapIndex	integer	Индекс карты, на которую ссылается системный объект @Map. Доступен на чтение и запись. Меняя этот индекс можно перебрать все карты.
@Obj	объект	Графический объект
@ObjCnt	integer	Количество графических объектов на текущей карте @Map
@ObjIndex	integer	Индекс графического объекта на текущей карте @Map. Доступен на чтение и запись. Меняя этот индекс можно перебрать все объекты.
@rtvar	объект	Текущий канал LanMon
@rtvarcnt	Integer	Количество каналов LanMon
@rtvarindex	integer	Индекс канала LanMon. Доступен на чтение и запись. Меняя этот индекс можно перебрать все каналы.
@operator	объект	Канал текущего оператора системы. Пример: print "Адрес оператора=",@operator.A1,".",@operator.A2,".",@operator.A3,".",@operator.A4 print "Имя оператора=",@operator.NAME

## Операторы

Операторы являются связующим звеном между остальными элементами программы и служат для выполнения арифметических и логических операций. Все операторы имеют приоритет выполнения. Например: оператор умножения имеет больший приоритет оператора сложения, т.е.  $1+3*2=7$ . Для уточнения последовательности выполнения различных операторов надо использовать круглые скобки ().

Ниже приведены все операторы в порядке увеличения приоритета:

### &

результат = выражение1 & выражение2

Поразрядное логическое «И». Выражение1 и выражение2 приводятся к целому типу. Результат всегда integer.

|

результат = выражение1 | выражение2

Поразрядное логическое «ИЛИ». Выражение1 и выражение2 приводятся к целому типу. Результат всегда integer.

## AND

результат = выражение1 AND выражение2

Логическое «И». Выражение1 и выражение2 приводятся к целому типу. Результат всегда integer (0 или 1).

## OR

результат = выражение1 OR выражение2

Логическое «ИЛИ». Выражение1 и выражение2 приводятся к целому типу. Результат всегда integer (0 или 1).

Операторы отношения.

Оператор	Название
<	Меньше
<=	меньше или равно
>	Больше
>=	больше или равно
=	Равно
<>	не равно

Результат операторов отношения всегда integer 0 или 1 (ложь или истина).

+

результат = выражение1 + выражение2

выражение1	выражение2	результат
integer	integer	integer
floating	floating	floating
string	string	string
integer	floating	floating
integer	string	string
floating	integer	floating
floating	string	string
string	integer	string
string	floating	string

-

результат = выражение1 - выражение2

выражение1	выражение2	результат
integer	integer	integer
floating	floating	floating

string	string	floating
integer	floating	floating
integer	string	floating
floating	integer	floating
floating	string	floating
string	integer	floating
string	floating	floating

**\***

результат = выражение1 \* выражение2

выражение1	выражение2	результат
integer	integer	integer
floating	floating	floating
string	string	floating
integer	floating	floating
integer	string	floating
floating	integer	floating
floating	string	floating
string	integer	floating
string	floating	floating

**/**

результат = выражение1 / выражение2

выражение1	выражение2	результат
integer	integer	integer
floating	floating	floating
string	string	floating
integer	floating	floating
integer	string	floating
floating	integer	floating
floating	string	floating
string	integer	floating
string	floating	floating

**%**

результат = выражение1 % выражение2

Результат равен остатку от деления выражения1 на выражение2.

Выражение1 и выражение2 приводятся к целому типу. Результат всегда integer.

**^**

результат = выражение1 ^ выражение2

Результат равен возведению выражения1 в степень выражение2.

Выражение1 и выражение2 приводятся к типу floating. Результат всегда floating.

## **Унарный –**

результат = - выражение1

Результат равен выражению1 с измененным знаком. Тип результата соответствует типу выражения.

## **NOT**

NOT выражение1

Логическое НЕ. Унарная операция. Результат всегда integer: 0 или 1.

**;**

Оператор точка с запятой применяется для разделения операций, расположенных в одной строке. Например:

a=5; b=1; c=a\*b

**=**

Оператор присваивания. Формат: «переменная» = выражение. Если указанная переменная не существует она будет создана. В любом случае у переменной будет установлен тип выражения.

Пример:

TEST = 4^2

Данный пример создает переменную TEST типа floating со значением 16.0, так как операнд ^ всегда дает тип floating.

## Ключевые слова

### ***IF ... THEN ... ELSE ... ENDIF***

Условное исполнение группы операций в зависимости от условия.

**IF** условие **Then**  
    операции

    [**Else**  
        операции]  
**EndIf**

При выполнении **IF** условие вычисляется. Если условие истина (не ноль) операции после **THEN** выполняются. Если условие ложь (ноль) операции после **ELSE** выполняются. После выполнения блока операций **THEN** или **ELSE** выполнение программы продолжается с операции, следующей за **ENDIF**.

Однострочный **IF** также должен завершаться ключевым словом **ENDIF**, причем все операции после **THEN** должны разделяться точкой с запятой.

**IF** a=0 **THEN** print "OK"; **ENDIF**

```
IF a = 0 AND b = 0 THEN
    REM "a = NOT a" Эквивалентно выражению "IF a=0 THEN a=1 ELSE a=0 ENDIF"
    a = NOT a
    b = NOT b
ELSE
    a = 0
    b = 0
ENDIF
print "a=",a
print "b=",b
```

Можно делать вложенные **IF**. Степень вложенности – не более 15.

```
IF a = 0 Then
    IF b = 0 Then
        c = 0
    Else
        c = 1
    EndIf
EndIf
```

### ***FOR ... TO ... STEP ... NEXT***

Выполнение группы операций заданное количество раз.

**For** *счетчик* = *начальное значение* **To** *конечное значение* [**Step** *приращение*]  
[операции]  
**Next**

«Счетчик» – любая переменная, используемая как счетчик цикла.

«Приращение» – на сколько увеличивать переменную «счетчик» после каждого цикла. Если ключевое слово STEP пропущено в качестве приращения берется единица.

«Начальное значение» и «конечное значение» это произвольные выражения. Их значение вычисляется 1 раз перед началом цикла.

Следующий пример показывает простое использование цикла FOR

```
For i = 0 To 25
  print i
Next
```

Следующий цикл уменьшает переменную i

```
For i = 0 To -25 Step -2.56
  print i
Next
```

Вложенный FOR

```
For i = 0 To 5
  For j = 0 To 5
    print "i = ", i, " j = ", j
  Next
Next
```

## **WHILE ... WEND**

Выполнение группы операций пока условие истинно (не равно нулю).

**While** *условие*  
*операции*  
**Wend**

Если условие истинно (не равно нулю), все операции выполняются пока не встретиться WEND. Потом управление возвращается на WHILE и условие проверяется снова. Если условие остается истинно процесс повторяется. Если условие становится ложным (равно нулю) управление передается на операцию, следующую за WEND.

Пример:

```
a = 0
While a > -5
  print "a = ", a
  a = a - 1
Wend
```

Результат работы:

```
a = 0
a = -1
a = -2
a = -3
a = -4
```

## ***REM***

Вставка комментариев в программу. Все окончание строки, следующее за командой REM считается комментарием.

Пример:

```
REM Следующая команда выводит приветствие
PRINT "Hello !"
```

## ***END***

Завершение работы программы

В следующем примере если переменная «a» равна нулю – выполнение программы останавливается:

```
IF a=0 THEN END ENDIF
```

## ***GOTO***

Переход на метку. Метка должна начинаться с буквы латинского алфавита a...z и заканчиваться двоеточием. Максимальная длина названия метки - 15 символов.

В следующем примере команда PRINT не выполняется:

```
GOTO label1
print "Hello !"
label1:
end
```

## ***RUN ... FROM ...***

Запуск внешнего приложения или функции из DLL библиотеки. Общий вид оператора:

**RUN “filename”**

или

**RUN “funcname” FROM “dllfilename”**

«filename» - имя исполняемого файла

«dllfilename» - имя файла динамической библиотеки

«funcname» - имя функции в динамической библиотеке

Пример:

**REM Запуск приложения**

**RUN "C:\Program Files\WinAmp\WinAmp.exe"**

**REM Вызов функции из библиотеки**

**RUN "@ShowResult\$q" FROM "Bpdd.dll"**

## ***PRINT .....***

Оператор используется для отладочной печати в терминальное окно LanMon.

Общий вид оператора:

PRINT параметр 1, параметр 2, ...

Параметр – Строка в двойных кавычках или выражение

## Встроенные функции

LanMon бейсик имеет ряд предопределенных функции. Количество аргументов у функции может быть от нуля (нет аргументов) до 10. Причем есть функции с переменным числом аргументов. Например, функция форматирования строки `Sprintf`. Функция может получать значения любого из трех допустимых типов. Функции могут использоваться в составе любых выражений, а также на манер процедуры (т.е. без использования ее возвращаемого значения). Далее приводятся все встроенные функции по категориям.

### Функции преобразования типов

Название	Описание
<code>IsInteger()</code>	Возвращает 1 если аргумент типа <code>integer</code> и 0 в противном случае
<code>IsFloating()</code>	Возвращает 1 если аргумент типа <code>floating</code> и 0 в противном случае
<code>IsString()</code>	Возвращает 1 если аргумент типа <code>string</code> и 0 в противном случае
<code>AsString()</code>	Преобразует аргумент в выражение типа <code>string</code>
<code>AsInteger()</code>	Преобразует аргумент в выражение типа <code>integer</code>
<code>AsFloating()</code>	Преобразует аргумент в выражение типа <code>floating</code>

### Функции общего назначения

Название	Аргументов	Описание
<code>iif()</code>	3	Если первый аргумент не равен нулю значение функции равно второму аргументу. Иначе значение функции равно третьему аргументу.
<code>sprintf(...)</code>	Переменное (не более 12)	Форматирование строки как в языке C. Первый аргумент это маска форматирования, остальные аргументы это значения для подстановки в маску. Сформатированная строка возвращается. Например: <code>print sprintf("%s %d", "Значение=", -100)</code> Будет напечатано: <code>Значение=-100</code>
<code>choose(...)</code>	Не менее 3х и не более 12	Первый аргумент функции – это номер (с нуля) аргумента, который и будет значением функции. Ноль – значением функции будет второй аргумент. Тип возвращаемого значения соответствует типу выбранного аргумента.  Пример: <code>print choose(1, «Первый», «Второй», «Третий»)</code> Даст на печать: «Второй».
<code>protokol(...)</code>	Переменное (не более 12)	То же, что и <code>sprintf</code> , но вывод строки в протокол работы LanMon.LOG. Например: <code>protokol("Ошибка выполнения операции %d!", -100)</code>
<code>addfile("имя_файла", "строка")</code>	2	Добавить строку в текстовый файл. Первый аргумент – имя файла. Второй аргумент – строка. После записи строки в файл автоматически добавляются символы перевода строки. Возврат: 0 – ошибка открытия или записи файла

		1- все хорошо
deletefile("имя_файла")	1	Удаление указанного файла. Удаление производится безвозвратно. Возвращает ноль при ошибке.

### Функции, специфичные для LanMon

Название	Аргументов	Описание
LMSendControl(A1, A2, A3, A4, STATE, VAL)	6	Послать команду управления драйверам оборудования и на сервер LanMon. Функция получает адрес канала, его состояние и значение. При ошибке возвращает нулевое значение. Например: <i>LMSendControl(1,2,3,4,0,1)</i>
LMSendState(A1, A2, A3, A4, STATE, VAL)	6	Послать состояние канала на сервер LanMon подобно тому, как это делает опросчик. Функция получает адрес канала, его состояние и значение. При ошибке возвращает нулевое значение. Например: <i>LMSendState(1,2,3,4,0,1)</i>
LMSendStateLocal(A1, A2, A3, A4, STATE, VAL)	6	Послать состояние канала самому себе. Подобно тому, как будто бы оно пришло от сервера или от драйвера оборудования. Функция получает адрес канала, его состояние и значение. При ошибке возвращает нулевое значение. Например: <i>LMSendStateLocal(1,2,3,4,0,1)</i>
LMServerStatus(i)	1	Если i=0 функция возвращает статус подключения к серверу LanMon в числовом виде. Если i=1 функция возвращает статус подключения к серверу LanMon в виде текстовой строки. Возможные возвращаемые значения: 0-"ОТКЛЮЧЕН !" 1-"Подключение..." 2-"Регистрация..." 3-"Запись маски..." 4-"Получение A1..." 5-"Получение A2..." 6-"Получение A3..." 7-"Получение A4..." 8-"Получение каналов..." 9-"ОК" 10-"ОТКЛЮЧЕН ! (РЕЗЕРВНЫЙ)" 11-"Подключение... (РЕЗЕРВНЫЙ)" 12-"Регистрация... (РЕЗЕРВНЫЙ)" 13-"Запись маски... (РЕЗЕРВНЫЙ)" 14-"Получение A1... (РЕЗЕРВНЫЙ)" 15-"Получение A2... (РЕЗЕРВНЫЙ)" 16-"Получение A3... (РЕЗЕРВНЫЙ)" 17-"Получение A4... (РЕЗЕРВНЫЙ)" 18-"Получение каналов... (РЕЗЕРВНЫЙ)" 19-"ОК (РЕЗЕРВНЫЙ)"
LMViewLog()	0	Открыть окно журнала.

LMViewArchive()	0	Открыть окно ввода параметров для просмотра архива.
LMViewAlarm()	0	Открыть окно просмотра алармов.
LMExit()	0	В режиме отладки проекта: завершение отладки. В режиме выполнения проекта: завершение работы LanMon.
LMOperatorChange()	0	Вызов модального окна смены оператора. Пример использования смотрите в файле <code>.\samples\operator.bas</code>
LMFormatString(a1,a2,a3,a4,"mask")	5	Сформатировать параметры канала в текстовую строку. Возвращает полученную строку. Первые 4 аргумента – адрес канала. Пятый аргумент маска форматирования: %DATE – дата изменения канала %TIME – время изменения канала %STATE – состояние канала %VALUEn(%mask) – Значение канала. n – номер значения. %mask – маска форматирования значения канала (как в функции sprintf). %A1 - текстовая расшифровка адреса A1 %A2 - текстовая расшифровка адреса A2 %A3 - текстовая расшифровка адреса A3 %A4 - текстовая расшифровка адреса A4 %NL – вставка символов новой строки  LMFormatString(1,2,3,4,"%DATE %TIME %STATE %VALUEn(%mask) %A1 %A2 %A3 %A4 %NL")
LMGetCSV("имя_файла",строка, колонка)	3	Прочитать одно значение из файла в формате CVS (текстовый файл с колонками, разделенными запятыми). Аргументы: имя файла, номер строки, номер колонки. Возвращает найденное значение типа string или ноль типа integer при ошибке.  Пример использования смотрите в файле <code>.\samples\GetCSV.bas</code>
RtVarByAddr(A1,A2,A3,A4)	4	Имеет 4 целочисленных параметра – адрес канала. Возвращает объект – канал LanMon. Например, текущее значение канала с адресом 1.2.3.4: <i>RtVarByIndex(1,2,3,4).value</i>
RtVarByIndex()	1	Имеет один параметр – индекс канала с нуля. Возвращает объект – канал LanMon. Например, текущее значение канала с индексом 0: <i>RtVarByIndex(0).value</i>

RTV()	1 или 4	<p>Универсальная замена предыдущих двух функций:</p> <p><i>RTV(1,2,3,4).value</i> – значение канала с адресом 1.2.3.4</p> <p><i>RTV(«1.2.3.4»).value</i> - значение канала с адресом 1.2.3.4 Адрес канала задан строковым выражением.</p> <p><i>RTV(0).value</i> – значение канала с индексом ноль в списке каналов.</p>
AAState(индекс аларма, тип результата)	2	<p>Получить состояние аналогового аларма.</p> <p>Первый аргумент - название аларма или его номер с нуля.</p> <p>Если второй аргумент равен нулю, функция возвращает целочисленное состояние аларма:</p> <ul style="list-style-type: none"> <li>-1-указанный аларм не найден</li> <li>0-ок</li> <li>1-повышение</li> <li>2-недопустимое повышение</li> <li>3-понижение</li> <li>4-недопустимое понижение</li> <li>5-неисправность канала</li> </ul> <p>Если второй аргумент равен единице, функция возвращает текстовую расшифровку состояния аларма.</p>

### Функции для работы с отчетами RTF

rptRun("шаблон", "отчет")	2	Формирует отчет из шаблона. Смотри пример: report.bas
rptShow("Заголовок окна", "отчет")	2	Показывает отчет из файла на экране.
rptPrint("отчет")	1	Печатает отчет из файла на принтере по умолчанию.

### Функции для работы с генератором отчетов

Название	Аргументов	Описание
frLoad("имя_файла.fr3")	1	<p>Загружается в память шаблон отчета. Шаблоны отчетов хранятся в файлах с расширением fr3 в поддиректории .\FR</p> <p>Если задано краткое имя файла шаблона отчета, оно будет искажаться в поддиректории .\FR</p> <p>Функция возвращает ноль в случае</p>

		ошибки.
frSave("имя_файла.fr3")	1	Сохраняет шаблон отчета из памяти в файл. Если задано краткое имя файла шаблона отчета, оно будет сохранено в поддиректорию .\FR Функция возвращает ноль в случае ошибки.
frRun(1)	1	Запускает шаблон отчета из памяти на выполнение. После выполнения отчета показывает его в окне предварительного просмотра. Шаблон отчета должен быть предварительно загружен функцией frLoad(). Если аргумент равен единице – ранее построенный отчет очищается. Если нулю – новый отчет будет добавлен к ранее построенному.
frPrepare(1)	1	Запускает шаблон отчета из памяти на выполнение. После выполнения отчета ничего не делает. Для показа выполненного отчета в окне предварительного просмотра вызывайте функцию frShowPrepared(). Шаблон отчета должен быть предварительно загружен функцией frLoad(). Если аргумент равен единице – ранее построенный отчет очищается. Если нулю – новый отчет будет добавлен к ранее построенному. Функция возвращает ноль в случае ошибки.
frShowPrepared()	0	Показывает текущий отчет, ранее подготовленный функциями frPrepare или frRun.
frDesign()	0	Вызывает редактор шаблонов отчетов fr3 в режиме выполнения проекта.
frPrint()	1	Печать отчета, ранее подготовленного функциями frPrepare или frRun. Если аргумент равен единице – показывает окно выбора принтера. Если нулю – печать осуществляется на принтере по умолчанию без запроса пользователя.
frExport(формат_экспорта, "имя_файла")	1 или 2	Производит экспорт текущего

		<p>отчета в файл одного из форматов. Отчет должен быть ранее подготовлен функциями frPrepare или frRun.</p> <p>формат_экспорта:</p> <p>0 – TXT</p> <p>1 – HTML</p> <p>2 – XLS</p> <p>3 – XML</p> <p>4 – RTF</p> <p>5 – BMP</p> <p>6 – JPEG</p> <p>7 – TIFF</p> <p>8 – PDF</p> <p>Если второй аргумент – “имя файла” – не задан, то выдается диалоговое окно параметров экспорта. Если файл задан – производится экспорт без запросов пользователя.</p> <p>Функция возвращает ноль в случае ошибки.</p>
frVariable(“имя_переменной”,”значение”)	1 или 2	<p>Получить или установить значение переменной скрипта отчета. Если переменной с указанным именем нет – она будет создана. Функция всегда возвращает значение указанной переменной.</p> <p>Пример:</p> <pre>print frVariable(“start_date”,”18.07.2005”) print frVariable(“start_date”)</pre>

### Функции для работы с группами каналов

grpCount()	0	Возвращает целое число: текущее количество групп каналов.
grpByIndex()	1	<p>Имеет один параметр – индекс группы каналов с нуля. Возвращает объект – группа каналов.</p> <p>Например, состояние охраны первой группы каналов:</p> <p><i>grpByIndex(0).ohrana</i></p>
grpByName()	1	<p>Имеет один параметр – название группы каналов. Возвращает объект – группа каналов.</p>
grp()	1	<p>Если аргумент типа строка – аналог функции <i>grpByName()</i></p> <p>Иначе – аналог функции <i>grpByIndex()</i></p>

**Функции для работы со встроенным клиентом IP телефонии H323**

Название	Аргументов	Описание
H323State()	0	Получить текущее состояние встроенного IP телефона. Возвращает значение типа integer: 0 – трубка на крючке 1 – Трубка снята, идет разговор 2 – Идет набор номера (ожидание поднятия трубки на удаленной стороне) 3 – Идут входящие звонки
H323Call()	1	Послать вызов (позвонить). Если задан аргумент типа integer – это номер абонента из записной книжки, начиная с нуля. Если задан аргумент типа string – это узел или IP адрес вызываемого абонента. Данная функция выполняется, только если трубка на крючке. При успешном вызове возвращает ненулевое значение. Возвращает ноль при ошибке. Для получения кода ошибки используйте функцию H323LastError().
H323Handup()	0	Положить трубку (завершение разговора). Аргументов не имеет. Возвращает всегда ноль.
H323Answer()	0	Снять трубку когда идут входящие звонки: H323State()=3. Аргументов не имеет. Возвращает всегда ноль.
H323LastError()	1	Получение последней ошибки клиента IP телефонии. Имеет 1 аргумент: 0 – дать числовой код ошибки 1 – дать строковое описание ошибки Список ошибок приведен в таблице ниже.
H323ShowWindow()	1	Показать/скрыть окно клиента H323 с записной книжкой. Имеет один аргумент: 0 – скрыть окно 1 – показать окно Возвращает всегда ноль.

**Ошибки, возвращаемые H323LastError()**

Текстовое описание ошибки	Код ошибки
ОК	0
Неверно указан адресат !	1
Удаленный абонент повесил трубку...	2
Удаленный абонент прервал звонок...	3
Удаленный абонент отказался от разговора...	4
Соединение перегружено	5
Удаленный абонент не ответил на звонок...	6
Сбой соединения	7
Не найден общий кодек	8
Мы не приняли входящий звонок	9
Входящий звонок отвергнут	10
Привратник не нашел указанного абонента	11
Звонок прерван ! (Не хватает полосы канала)	12
Удаленный абонент недоступен	13
Удаленный абонент сейчас отключен	14

На удаленной стороне нет телефона	15
Ошибка звонка !	16

### Работа со трендами (графиками)

Название	Аргументов	Описание
TrendAssign()	2 или 3	Отобразить тренд на один из 10 возможных графиков в окне просмотра графиков. Первый аргумент – название тренда или индекс тренда в списке. Если индекс тренда равен -1 это означает: спрятать указанный график. Второй аргумент индекс графика, на который отображаем тренд от 0 до 9. Третий, опциональный, аргумент номер архивного файла тренда от 1 до 99. Если третий аргумент отсутствует - берутся текущие значения тренда. При ошибке возвращает ноль.
TrendShowWindow()	1	Показать/скрыть окно вывода графиков. Аргумент: 0-скрыть 1-показать
TrendSetCaption()	1	Установить заголовок окна графиков.
TrendLoadParam()	1	Загрузить файл с настройками окна просмотра графиков. Аргумент – имя файла с настройками. Создание файла настроек выполняется из окна настройки параметров графиков в режиме редактирования проекта.

### Работа со временем

Название	Аргументов	Описание
time()	0	Возвращает текущее время в формате, принятом в LanMon. Тип – floating.
decodetime(time(),i)	2	Извлечение из полного времени элементов. Первый аргумент время – тип floating. Второй аргумент i, определяет что извлечь. Возвращаемое значение имеет тип integer. Перечислим возможные значения второго аргумента i: 0-Извлекаем из времени в переменной a текущий год 1-Извлекаем текущий месяц 2-Извлекаем текущий день 3-Извлекаем текущий час 4-Извлекаем текущую минуту 5-Извлекаем текущую секунду 6-Извлекаем текущую миллисекунду Смотрите пример: <b>time.bas</b>

### Работа со строками

Название	Аргументов	Описание
----------	------------	----------

Len()	1	Возвращает длину строки – аргумента.
InStr(str1,str2)	2	Возвращает позицию str2 в str1 (с нуля.) Например: a = InStr("Hello", "e") REM Returns 1. a = InStr("Hello", "He") REM Returns 0. a = InStr("Hello", "abc") REM Not found. Returns -1.
Mid	3	
Right	2	
Left	2	
RTrim	1	
Ltrim	1	
Trim	1	
UCase	1	
LCase	1	

### Математические функции

Название	Аргументов	Описание
min()	2	Возвращает наименьший из двух аргументов
max()	2	Возвращает наибольший из двух аргументов
randomize()	0	Инициализировать датчик случайных чисел
random(i)	1	Возвращает случайное целое число от 0 до i-1
binary(число, количество_бит)	2	Преобразовать целое «число» в двоичный формат. При этом использовать только указанное количество бит. Возвращает символьную строку.
sqr	1	
hypot	2	
abs	1	
Sin	1	
Cos	1	
Tan	1	
Atan	1	
Asin	1	
Acos	1	
atan2	2	
RadToDeg	1	
DegToRad	1	
exp	1	
log	1	
log10	1	