目 录

命令语言函数速查手册

“组态王”支持使用内建的复杂函数，其中包括字符串函数、数学函数、系统函数、控件函数、报表函数、SQL 函数、配方函数、报警函数及其它函数，下面依次介绍各个函数（函数名不区分大小写，按字母排序）：

Abs

此函数用于计算变量值的绝对值，使用格式如下：

Abs(变量名或数值);

返回值：整值或实型值；

例如：

Abs(14); 返回值为 14

Abs(-7.5); 返回值为 7.5

Abs(距离); 返回内存模拟变量“距离”的绝对值。

Ack

对变量进行报警确认，或对报警组进行报警确认。如果函数参数为变量名称，则只对该变量进行报警确认；如果函数参数为报警组名称，则确认所有属于该报警组及其子报警组的变量。该函数的参数只能是变量名或报警组名，不可以为字符串变量。此函数常用于按钮命令语言，当发生报警时，用此函数进行报警确认，它将产生确认报警事件。调用格式：

Ack(报警组名); 或 Ack(变量名);

例如：

Ack(全厂)； 或 Ack(反应罐液位)；

AckByTagName

对变量进行报警确认。函数参数可以是一个字符串变量，也可以是一个表示变量名的字符串。调用格式：

AckByTagName("tag_name");

参数：tag_name：变量名。

例如：

AckByTagName ("\\本站点\液位");

AckByTagName(Varname);其中 Varname 为字符串变量。

AckByGroupName

对报警组进行确认。调用格式：

AckByGroupName("station_name", "group_name");

station_name 为产生报警的 IO 服务器名，

group_name 为报警组名。

函数参数可以是字符串变量，也可以是表示 IO 服务器名或报警组名的字符串。

例如：

AckByGroupName ("PC1","报警组 1");

AckByGroupName (PCName, GroupName);//PCName, GroupName 为字符串变量。

ActivateApp

此函数用于激活正在运行的窗口应用程序，使之变为当前窗口。获得输入焦点。该函数也可配合函数 SendKeys 的使用。调用形式：

ActivateApp(“ExeName”);

参数：ExeName 应用程序的执行文件名

例如：

激活 Microsoft Word 的正确调用为：

ActivateApp("Word.exe");

激活组态王，可使用：

ActivateApp("TouchVew.exe");

ActiveXIsVisible

此函数用于控制窗体控件隐含。调用形式：

ActiveXIsVisible("CtrlName", nMode);

参数：CtrlName 控件名

nMode 控制模式。nMode=0 时，控件隐含。

例如：

实现控件隐含的正确调用为：

ActiveXIsVisible(“Ctrl10”, 0);

ArcCos

此函数用于计算变量值的反余弦值，变量值的取值范围在[-1，1]之间，否则函数返回值无效。调用格式：

ArcCos(变量名或数值);

返回值：整值或实型值；

例如：

ArcCos(1)；此函数返回值为 0

ArcCos(temp)；此函数返回变量“temp”的反余弦值。

ArcSin

此函数用于计算变量值的反正弦值，变量值的取值范围在[-1，1]之间，否则函数返回值无效。调用格式：

ArcSin(变量名或数值);

返回值：整值或实型值；

例如：

ArcSin(1)；此函数返回值为 90

ArcSin(temp)；此函数返回变量“temp”的反正弦值。

ArcTan

此函数用于计算变量值的反正切值，使用格式为：

ArcTan(变量名或数值);

返回值：整值或实型值;

 例如：

ArcTan(1); 此函数返回值为 45

ArcTan (temp); 此函数返回变量“temp”的反正切值。

Average

此函数为对指定的组态王报表表格的多个单元格求平均值，或求多个变量的平均值。语法格式使用如下：

Average(’a1’,’a2’); 或 Average('a1:a10');

a1、a2……为组态王单元格所在的行号列标，或整型或实型变量。其中参数个数为 1-32 个。

当对报表的指定单元格区域内的单元格进行求平均值运算时，结果显示在当前单元格内，语法格式使用如下：

Average（‘a1’, ‘ a2’）;

 例如：

任意单元格选择求平均值:

=Average（‘a1’,‘b2’,‘r10’）;

连续的单元格求平均值:

=Average(‘b1:b10’);

或求变量的平均值：

```
AverageValue= Average(lVar1,fVar1);
```

BackUpHistData

此函数为组态王网络中从 IO 服务器上下载历史数据记录到历史记录服务器。用户在历史记录服务器上调用该函数。函数的使用需要与组态王网络配置相配合，具体内容参见《组态王 6.55 使用手册》中“历史库”一章。

语法使用格式：

```
BackUpHistData (Str chMchinename, Long ftEndtime);
```

参数：chMchinename 字符串型　为进行备份存储的 IO 服务器名

ftEndtime　整型　为备份截止时间

例如：

```
//备份“IO 采集站”的历史数据，截至时间为当前：
endTime=HTConvertTime($年,$月,$日,$时,$分,0);
BackUpHistData("IO 采集站", endTime);
```

Bit

此函数用以取得一个整型或实型变量某一位的值(0 或 1)。用法：

```
OnOff=Bit( Var , bitNo); //OnOff:离散变量
```

参数：Var:整型或实型变量

bitNo:位的序号，取值 1 至 16

返回值：离散型。若变量 Var 的第 bitNo 位为 0，返回值 OnOff 为 0；　若变量 Var 的第 bitNo 位为 1，返回值 OnOff 为 1。

例如：

开关=Bit(DDE1,6)； 从变量 DDE1 的第 6 位得到变量“开关”状态。

BitSet

此函数将一个整型或实型变量的任一位置为指定值(0 或 1)。语法格式：

BitSet(Var, bitNo, OnOff);

参数：Var:整型或实型变量

bitNo:位的序号，取值 1 至 16

OnOff:位的设定值

注意：

对于 IO 变量来说，BitSet 函数只是用于可读可写的变量。

例如：

BitSet(DDE1,6,0)； 将变量 DDE1 的第 6 位置为 0。

ChangePassword

此函数显示“更改口令”对话框，允许登录工程人员更改他们的口令。使用格式：

ChangePassword();

例如：

为画面上某一按钮设置命令语言连接：

ChangePassword( );

运行时单击此按钮，弹出对话框：

提示工程人员输入当前的口令和新口令以及验证新口令。完全正确后，工程人员的口令设置为新值。

chartAdd

此函数用于在指定的棒图控件中增加一个新的条形图。语法格式如下：

chartAdd("ControlName", Value, "label");

参数：ControlName：工程人员定义的棒图控件名称，可以为中文名或英文名。

Value：设定条形图的初始值，整形数据，实型数据。

label：设定条形图的标签值，默认值=索引值 Index，Index 的取值范围是 1-16。

例如：

chartAdd("XYChart", 1, "L6");

此语句将在棒图控件 XYChart 中增加一个标签为 L6 的条形图，其初始值为 1。

chartClear

此函数用于在指定的棒图控件中清除所有的棒形图。语法格式如下：

chartClear("ControlName");

参数：ControlName：工程人员定义的棒图控件名称，可以为中文名或英文名。

例如：

chartClear("XYChart");

此语句把棒图控件 XYChart 中的所有棒图清除。

chartSetBarColor

此函数用于在指定的棒图控件中设置饼图的颜色。条形图不可以。语法格式如下：

chartSetBarColor("ControlName", barIndex, colorIndex);

参数：

ControlName：工程人员定义的棒图控件名称，可以为中文名或英文名。

barIndex: 整型变量，表示条形图索引号，用于设定指定的条形图，其取值范围为 0-15。

colorIndex: 整型变量，表示条形图的颜色索引号，用于设置指定条形图的颜色，其取值范围为 0-15，颜色索引号和相应的颜色如下所示。

颜色索引号	代表颜色	颜色索引号	代表颜色
0	Default	8	Gray
1	Blue	9	Light Blue
2	Green	10	Light Green

3	Cyan	11	Light Cyan
4	Red	12	Light Red
5	Magenta	13	Light Magenta
6	Yellow	14	Light Yellow
7	White	15	Bright White
		16	Black

例如：

chartSetBarColor("XYChart", 0, 1);

此语句将棒图控件 XYChart 中第一块饼图的颜色设为 blue（即兰色）。

chartSetBarColor("XYChart", 2, 4);

此语句将棒图控件 XYChart 中第三块饼图的颜色设为 red（即红色）。

chartSetValue

此函数用于在指定的棒图控件中设定/修改索引值为 Index 的条形图的数据。语法格式如下：

chartSetValue("ControlName", Index, Value);

参数：

ControlName：工程人员定义的棒图控件名称，可以为中文名或英文名。

Value：设定条形图的数据，整形数据，实型数据。

Index：条形图的标签值，，Index 的取值范围是 0-15，组态王自动从 0 开始加 1，给每一个新增加的条形图由小到大设定标签值。

例如：

chartSetValue("XYChart",2, 30);

此语句将在棒图控件 XYChart 中设定索引值为 2（第三条）的条形图的数据为 30。

ClosePicture

此函数用于将已调入内存的画面关闭，并从内存中删除。语法格式如下：

ClosePicture("画面名");

例如：

ClosePicture("反应车间"); 将关闭画面“反应车间”。

ConfigODBC

此函数用于配置 odbc 的数据源，语法格式如下：

ConfigODBC(nDatatbasetype, szAttributes);

参数：

nDatatbasetype：数据库类型，目前支持 Access，SQL server 该参数为 0 时表示数据库类型为 Access，参数为 1 时表示数据库类型为 SQL server ；

szAttributes：配置字符串：

例 1：

配置 Access 数据库，DSN 名称为 demo2，数据库文件为 E:\Program Files\Kingview\Example\Kingdemo1\数据库.mdb。

ConfigODBC(0,"DSN=demo2\ODBQ=C:\Program Files\kingview\Example\kingdemo1\SQL 数据库.mdb\OPWD=1234\OUID=shihf");

例 2：

配置 SQL 数据库：

ConfigODBC(1,"DSN=MyDSN\ODBQ=SQLConfigDSN Sample\OSERVER=MySQL\OADDRESS=MyServer\ONETWORK=dbmssocn\ODATABASE=pubs\O");

Cos

此函数用于计算变量值的余弦值，语法格式如下：

Cos(数值或变量名);

例如：

Cos(90); 返回值为 0

Cos(temp); 返回变量“temp”的余弦值。

Date

此函数为根据给出的年、月、日整型数，返回日期字符串，默认格式为：年：月：日。语法使用格式如下：

Date（LONG nYear, LONG nMonth, LONG nDay）;

例如：

年、月、日变量分别为：“$年”、“$月”、“$日”，用日期来显示由以上三个整数决定的“日期”字符串，则在命令语言中输入：

日期=Date（年,月,日）；

DisplayMCI

此函数提供了一个对多媒体设备的通用接口，具有强大的功能。语法使用格式：

DisplayMCI("MCICommand", option);

下面举例说明此函数的使用方法。

例如：

DisplayMCI(“PLAYCD”,3);

用于播放 CD 唱片中的第 3 支歌曲。

DisplayMCI(“STOPCD”,” ”);

用于停止播放 CD。

DisplayMCI(“PLAYMIDI”,” c:\midi.mid”);

用于播放 MIDI 格式的背景音乐”c:\midi.mid”。

DisplayMCI(“PAUSEMIDI”,” c:\midi.mid”);

暂停播放 MIDI 格式的背景音乐”c:\midi.mid”。

DisplayMCI(“RESUMMIDI”,” c:\midi.mid”);

继续播放 MIDI 格式的背景音乐”c:\midi.mid”。

DisplayMCI(“CLOSEMIDI”,” c:\midi.mid”);

停止播放 MIDI 格式的背景音乐”c:\midi.mid”。

DisplayMCI(“EJECTCD”);

将光驱中的 CD 盘片弹出。

Dtext

此函数用于按离散变量的值动态地改变字符串变量。语法格式如下：

Str = Dtext(Discrete_Tag, OnMsg, OffMsg);

参数描述

Discrete_Tag 离散变量名。

OnMsg 字符串变量名

OffMsg 字符串变量名

当 Discrete_Tag＝1 时，Str 的值为 OnMsg；当 Discrete_Tag＝0 时，Str 的值为 OffMsg。

例如：

Str = Dtext(电源开关，"电源打开","电源关闭");

当电源开关＝1 时，Str 的值为"电源打开"

当电源开关＝0 时，Str 的值为"电源关闭"。

EditUsers

此函数常用于按钮的命令语言连接，功能是在画面程序运行中配置工程人员。调用形式：

EditUsers( );

为配置其他工程人员，当前工程人员的权限必须不小于 900。

EnableAllAlarm

全局报警使能函数。

调用形式：

EnableAllAlarm(is_enable);

参数描述：

is_enable 整数变量或数值

0--禁止所有报警

非 0--使能所有报警

返回值：整数型，反映当前全局报警使能的状态：若为非 0：使能；为 0：禁止

EnableNavigate

此函数用于显示/关闭导航图。调用形式：

EnableNavigate(is_enable);

参数：is_enable，整型。

0: is_enable 为 0 时，关闭导航图；

1:is_enable 为 1 时，显示导航图。

注意

使用 EnableNavigate 函数关闭导航图后，除非使用该函数来显示导航图，否则无法显示导航图。

例如：

EnableNavigate(0); //关闭导航图

EnableDisableKeys

此函数用来定义 Alt/Win/ESC 键锁定禁止还是解除禁止。

调用形式：EnableDisableKeys(AltKey, EscKey, WinKey);

参数描述：

AltKey　　Alt 禁止标志(非 0：禁止，0：启用)

EscKey　　Esc 禁止标志

WinKey　　Win 禁止标志

参数类型为整型

当 AltKey 为非 0 时，不会屏蔽 Alt+Del+Ctrl，但会屏蔽“任务管理器”；

当 AltKey 为 0 时，会自动解除“任务管理器”的屏蔽。

注意：

此函数只适用于 NT 2000 XP 系统,一旦将其进行设置，若想再次恢复原有的状态，需要再次使用该函数将其设置到所需要的状态。

EnableSaveTagValueToIniValueWhenValueChange

对于设置了“保存数值”和“保存参数”的变量，使用该函数后，当变量的值和参数发生变化时，系统会自动将变量的值和参数保存到文件 tagname.db 中，无论组态王运行系统是否正常退出，再次运行后，将保存的变量值和变量参数作为变量的初始值和初始参数。

“保存数值”和“保存参数”的具体含义请参见《组态王手册》第七章 变量定义和管理 部分。

调用形式：EnableSaveTagValueToIniValueWhenValueChange(is_enable);

参数：

is_enable：整型变量或数值

1：当变量的值和参数发生变化时，保存变量数值和参数。组态王运行系统退出，再次运行后，将保存的变量值和变量参数作为变量的初始值和初始参数。

0：与组态王“保存数值”和“保存参数”实现的功能相同。

 例如：

组态王开发系统数据词典中定义变量，设置“保存数值”。执行函数：

EnableSaveTagValueToIniValueWhenValueChange(1);

该变量的值发生变化时，系统保存该变量的值。组态王运行系统退出，再次运行后，将保存的变量值作为变量的初始值。

Exit

此函数使组态王运行环境退出。调用形式：

Exit(Option);

参数：

Option: 整型变量或数值

0-退出当前程序；

1-关机；

2-重新启动 windows；

Exp

此函数返回指数函数 e^x 的计算结果，使用格式如下：

Exp(数值或变量名);

例如：

Exp(1); 返回 e1 的计算值 2.718

Exp(temp); 计算 e 常量的 temp 次幂并返回计算结果。

FileCopy

此函数复制一个源文件到目的文件，它与 DOS 的 Copy 命令或者 Windows 文件管理器中的 Copy 功能相似。调用格式：

FileCopy(SourceFile，DestFile，DoneTag);

参数描述

SourceFile 源文件名(包含完整的路径)。

DestFile 目的文件(包含完整的路径)或目录名(参见下面的例如子)。

DoneTag 该参数目前无效。用来报告复制过程进展情况的变量名称。此参数须是一个内存长整数或内存模拟型，随着复制过程的进行，该值从 0 变化到 100。

返回值：

成功返回 1；

不能启动返回 0；

出错返回-1；

例如：

Status=FileCopy("C:*.TXT", "C:\BACKUP", 'DoneTag');

Status：一个将被写为 1、-1 或 0 的整型变量。

FileCopy()函数在后台执行，这样它不会干扰组态王的运行。Status 表明的是复制过程是否已成功启动。一旦复制过程已成功启动，此过程成功结束，Status 被置为 1。若此过程结束前发生错误，则 Status 被置为-1。

SourceFile 和 DestFile 一般为文件名。但用 FileCopy() 函数复制单一文件时，目标文件名可以是一个目录，如：

FileCopy("C:\DATA.TXT", "C:\BACKUP", 'DoneTag');

将把文件“DATA.TXT”复制到“C：\”驱动器上一个叫做“BACKUP”的目录下。变量 Monctor 在复制完成后置为 1。

若 SourceFile 包含任何通配符的话，DestFile 必须是一个目录(而非文件名)，否则此函数将返回一个错误代码，如：

FileCopy("C:*.TXT", "C:\BACKUP", 'DoneTag');

将把 C 盘根目录下所有的.TXT 文件复制到 C：\BACKUP 目录下。

FileDelete

此函数删除不需要或不想要的文件。调用格式：

FileDelete(Filename);

参数描述

Filename 要删除的文件名。

若找到要删除的文件，并成功地删除，此函数将返回 1，否则此函数返回 0。

例如：

Status=FileDelete("C:\DATA.TXT");

若在 C:\找到 "DATA.TXT" 则 Status 等于 1, 未找到该文件则为 0。

FileMove

此函数与 FileCopy ()函数相似，但只是将文件从一个位置转移到另一个位置，而不是复制。调用格式：

FileMove(SourceFile, DestFile, DoneTag);

参数描述

SourceFile 源文件名(包含完整的路径)

DestFile 目的文件名(包含完整的路径)

DoneTag 用来报告移动过程进展情况的变量名称。此参数须是一个内存长整数或内存模拟型，随着转移过程的进行，该值从 0 变化到 100。

返回值：

成功返回 1；

不能启动返回 0；

出错返回-1；

例如：

Status=FileMove("C:\DATA.TXT","D:\DATA.TXT",Monitor);

Status 是一个将被写为 1、-1 或 0 的整型变量。

Monitor ：在数据词典中定义过的内存整数。

FileMove() 函数在后台执行，这样它不会干扰“组态王”的运行。使用 DoneTag 是为了允许应用程序或工程人员监视转移操作的进展。用这种方法，在转移过程启动后可能发生的任何错误都能使工程人员察觉。(此处用变量 Monitor 监测) 这与上述返回的 Status 不同，Status 表明的是转移过程是否已成功启动。一旦转移过程已成功启动，Monitor 就会被赋值 0。随着转移过程的进行，该值不断增加。当此过程成功结束时达到 100，Status 被置为 1。若此过程结束前发生错误，Status 被置为-1。

若源文件和目的文件位于同一驱动器上，此函数可以简单地更改此文件的目录参照表(计算机在此表中保存磁盘上的文件名和存储位置)，而不用实际转移任何数据。在这种情况下，不管此文件的大小，转移操作将会很快。若源文件和目的文件位于不同的驱动器上，转移操作所费的时间将随文件的大小不同而不同。这是因为数据必须由一个物理磁盘传送到另一物理磁盘上，如：

FileMove("C:\DATA.TXT","C:\BACKUP\DATA.TXT",Monitor);

将把“C”驱动器上根目录下的名为“DATA.TXT 的文件转移到名为“BACKUP”的目录下，变量 Monitor 在转移完成后将被置为 1。

此函数也可用于文件更名，只要源文件和目的文件指定了相同的目录，但不同的文件名，如：

FileMove ("C:\DATA.TXT","C:\DATA.BAK",Monitor);

将把 C 盘根目录下文件“DATA.TXT”更名为“DATA.BAK”。变量 Monitor 在其完成后被置为 1。

FileReadFields

此函数从一个指定文件中读出 CSV(逗号分隔变量)记录。调用格式：

FileReadFields(Filename,FileOffset,” StartTag”,NumberOfFields);

参数描述

Filename　　指定要读的文件。

FileOffset　指定读此文件的起始位置。若为 1，则表明从头开始

StartTag　　指定第一个数据要写到的那个组态王变量的名称。此变量名必须以一个数字结尾(如 MyTag1)。此参数必须是一个表明变量名的字符串(而非实际的变量本身)。所以，若变量叫做 MyTag1，就需要给出 MyTag1 或 MyTag1.name，而不仅仅是 MyTag1。

NumberOfFields　指定要读的字段数目(此文件的每条记录中以逗号隔开的字段的数目)。

若 StartTag 为“MyTag1”而 NumberOfField 为 3，则有 3 个字段从文件中读出并保存在 MyTag1、MyTag2 和 MyTag3 中。这些具有连续名字的变量必须先在组态王中创建，并可以属于不同的类型(整型，文字等等)。

例如：

若 C：\DATA\FILE.CSV 的第一行内容为：

“This is text, 3.1416, 5”，调用函数

BytePosition=FileReadFields("C:\DATA\FILE.CSV",1, "MyTag1", 3);

将读出此行，并把“This is text"保存在 MyTag1 中，3.1416 保存在 MyTag2 中，5 保存在 MyTag3 中：

此函数在读出之后返回新的字节位置。你可以在下次读时使用此返回值作为 FileOffset 的值，如：

BytePosition=FileReadFields(c:\DATA\FILE.CSV",FileOffset,"MyTag1",3)

。


注意:

StartTag 两侧必须加引号。

FileReadStr

此函数从指定文件中读出一指定数目的字节(或一整行)。调用格式:

FileReadStr(Filename,FileOffset,Str_Tag,CharsToRead);

参数描述

Filename 指定要读的文件。

FileOffset 指定读此文件的起始位置。若为 1，则表明从头开始。

Str_Tag 指定将从文件中读出的数据保存于何处。

CharsToRead 指定要从文件中读出多少字节。为处理文本文件，可将 CharsToRead 置为 0，函数从文件中一直读到下一个 LF(换行符)。

此函数在读出之后返回新的字节位置。可以在下次读时使用此返回值作为 FileOffset 值。


例如:

FileReadStr ("C:\DATA\FILE.TXT", 1, Str_Tag, 0);

文件“C: \DATA\FILE.TXT”的第一行将被读出并保存到 Str_Tag 中。

FileWriteFields

此函数往指定文件写入 CSV(逗号分隔变量)记录，或者使用已安装的打印机打印相关

的数据。调用格式：

FileWriteFields(Filename,FileOffset,” StartTag”,NumberOfFields);

参数描述

Filename　指定要写的文件。若文件不存在，则创建它。如果 FileOffset 取 2 或 3 值，此处是指定的打印机名称。

FileOffset 指定写此文件的位置。若 FileOffset 为 0，此函数将写到文件末尾。若为 1，则写到开头。若为 2，则调用打印机进行打印。此时打印数据类型使用打印机处理器中选择的默认打印数据类型（在打印机属性->高级->打印处理器中选择)。推荐针式打印机单行时，使用 RAW，喷墨，激光打印机整页打印时，使用 TAXT。

StartTag　指定第一个数据项的变量名称。此变量名必须以一个数字结尾(如 MyTag1)。此参数必须是一个表明变量名的字符串(而非实际的变量本身)。比如，变量名为MyTag1,就需要给出”MyTag1”(注意引号)或MyTag1.name,而不仅仅是MyTag1。

NumberOfFields　指定要写的字段数目(此文件的每条记录中以逗号隔开的字段的字段数目)。

此函数在写入之后返回新的字节位置。可以在下次调用函数时使用此返回值作为 FileOffset 值。

若 StartTag 为“MyTag1”，而 NumberOfFields 为 3，则有 3 个字段被写入文件中(写入的是 MyTag1、MyTag2 和 MyTag3)。这些具有连续名字的变量必须先在组态王中创建，并可以属于不同的类型(整型，字符串等等)。

例如：

将一行“This is text 3.1416，5”写到文件 C：\DATA\FILE.CSV 的第一行中。

“This is text”是 MyTag1 的当前值，3.1416 是 MyTag2 的当前值，5 是 MyTag3 的当前值。调用函数

FileWriteFields ("C:\DATA\FILE.CSV", 1, "MyTag1", 3);

若将文本串 MyTag1 写到 C：\DATA\FILE.CSV 的末尾，调用函数

FileWriteFields ("C:\DATA\FILE.CSV", 0, "MyTag1", 3);

若将文本串 MyTag1 使用打印机 EPSON LQ-1600K 打印出来使用函数

FileWriteFields ("EPSON LQ-1600K", 2, "MyTag1", 3);

注意：StartTag 两侧必须加引号。

FileWriteStr

此函数往指定文件写入指定数目的字节(或一整行)。调用格式：

FileWriteStr(Filename,FileOffset,String,LineFeed);

参数描述

Filename 指定写入的文件。若文件不存在，则创建它。

FileOffset　指定此文件的起始位置。若 FileOffset 为 0，此函数将写到文件末尾。若为 1，则写到开头

String　指定要写入文件中的字符。

LineFeed 规定是否在写操作之后添加换行。当写入一文本文件时，可以把 LineFeed 置为 1。

此函数在写入后返回新的字节位置。你可以在下次写时将此返回值当作 FileOffset() 函数的返回值来使用。

例如：

将名为 MsgTag 的字符串变量写入文件 C:\DATA\FILE.TXT 的末尾。调用函数：

FileWriteStr ("C:\DATA\FILE.TXT", 0, MsgTag, 1);

GetAlarmNumInGroup

此函数用来对某一个报警组当前的报警变量数量进行统计

语法格式：

long GetAlarmNumInGroup("MachineName", "GroupName");

参数说明：

MachineName：报警组所在的站点名称(对于单机网络，使用时用空字符串代替)

GroupName：要获取报警变量个数的报警组名

返回值：整型，返回值表示报警变量数量

特别说明：对于网络节点，需要把“本机为报警服务器”钩上，才能正确获取到远程节点的报警变量个数。

GetBackupProgress

此函数用于在组态王进行网络历史数据备份合并时获得进度百分比。使用时需要通过命令语言调用来获得进度值。语法使用格式：

GetBackupProgress(str szStationName);

参数：szStationName 远程站点名称。

返回值：整型，为 0~100 间的进度值。

例如：

备份进度= GetBackupProgress(“IO 采集站”);

GetCursorPosX

此函数返回当前鼠标的 X 坐标。

调用格式：

result=GetCursorPosX();

返回值为整型。

GetCursorPosY

此函数返回当前鼠标的 Y 坐标。

调用格式：

result=GetCursorPosY();

返回值为整型。

GetDate

此函数将以秒为单位的长整型数转换为相应的日期数值，分别以年、月、日等的日期数值输出。该长整型秒数的基准为 UTC(格林尼治)时间 1970 年 1 月 1 日 00:00:00。转换完成输出的日期也为 UTC(格林尼治)日期。语法使用格式

GetDate(DateTime,Year,Month,Day);

参数描述

DateTime:　需要进行日期转换的数，整型，为输入参数

Year： 年，整型，转换后得到的数据，输出参数

Month： 月，整型，转换后得到的数据，输出参数

Day： 日，整型，转换后得到的数据，输出参数

 例如：

自 UTC(格林尼治)时间 1970 年 1 月 1 日 00:00:00 到 2003 年 8 月 8 日 0:8:9 的秒的数值为 1060301289，使用 GetDate()函数可以从这个数值中分离出所表示的日期——年、月、日。

函数 GetDate(1060301289,年,月,日)； 执行后，得到的“年”的值为 2003，“月” 的值为 8，“日” 的值为 8。

获得其中时间的函数为 GetTime()。

GetDatelocal

此函数将以秒为单位的长整型数转换为相应的日期数值，分别以年、月、日的日期数值输出。该长整型秒数的基准为 UTC(格林尼治)时间 1970 年 1 月 1 日 00:00:00。转换完成输出的日期为本地日期。语法使用格式

GetDatelocal(DateTime,Year,Month,Day);

参数描述

DateTime： 需要进行日期转换的数，整型，为输入参数

Year： 年，整型，转换后得到的数据，输出参数

Month： 月，整型，转换后得到的数据，输出参数

Day： 日，整型，转换后得到的数据，输出参数

例如：

使用 HTConvertTime 函数将自 UTC(格林尼治)时间 1970 年 1 月 1 日 00:00:00 到北京时间2006年9月26日9:22:4转换为以秒为单位的长整数数值为1159233724，使用 GetDatelocal 函数可以从这个数值中分离出所表示的北京日期——年、月、日。

GetDatelocal (1159233724,年,月,日);

执行后，得到的“年”的值为 2006，“月” 的值为 9，“日” 的值为 26。

GetGroupName

此函数为通过报警组 ID 号获得报警组名称。在组态王中，每个报警组除了名称外，还有 ID 号。组态王的变量域“.Group”显示的是变量所属报警组的 ID 号，如果要获得相应的报警组名称，就需要使用该函数。语法使用格式

sGroupName= GetGroupName(StationName,GroupID);

参数描述

StationName：报警组所在的站点名称（该项暂时无效，使用时用空字符串代替）

GroupID：要获取名称的报警组的 ID 号

返回值为字符串型。

例如：

GroupName=GetGroupName(“”, \\本站点\原料罐液位.Group);

GetHistAveData

此函数用来获取某段时间中历史数据的平均值

语法使用格式：

RealResult=GetHistAveData(TagName,StartTime,EndTime);

参数描述：

TagName：所要查询的变量的名称，类型为字符串型，即带引号。

StartTime：数据查询的开始时间，该时间是通过组态王 HTConvertTime 函数转化的以 1969 年 12 月 31 日 16：00：00 为基准的长整型数，所以用户在使用本函数之前，应先将查询起始时间转换为长整型数值。

EndTime:数据查询的结束时间，类型同 StartTime。

返回值：返回一个实数值。

GetHistData

FLOAT GetHistData(const char* strTagName, short year, short mon, short day, short hour, short min, short sec)：用于取得某个时间点上的历史数据

函数参数说明：

const char* strTagName:变量名称，如\\本站点\变量名、“\\172.16.0.1\变量名”

short year:年、short mon:月、short day:日、short hour:时、short min:分、short sec:秒。

GetHistMaxData

此函数用来获取某段时间中历史数据的最大值

语法使用格式：

RealResult=GetHistMaxData(TagName,StartTime,EndTime);

参数描述：

TagName：所要查询的变量的名称，类型为字符串型，即带引号。

StartTime：数据查询的开始时间，该时间是通过组态王 HTConvertTime 函数转化的以 1969 年 12 月 31 日 16：00：00 为基准的长整型数，所以用户在使用本函数之前，应先将查询起始时间转换为长整型数值。

EndTime:数据查询的结束时间，类型同 StartTime。

返回值：返回一个实数值。

GetHistMinData

此函数用来获取某段时间中历史数据的最小值

语法使用格式：

RealResult=GetHistMinData(TagName,StartTime,EndTime);

参数描述：

TagName：所要查询的变量的名称，类型为字符串型，即带引号。

StartTime：数据查询的开始时间，该时间是通过组态王 HTConvertTime 函数转化的以 1969 年 12 月 31 日 16：00：00 为基准的长整型数，所以用户在使用本函数之前，应先将查询起始时间转换为长整型数值。

EndTime：数据查询的结束时间，类型同 StartTime。

返回值：返回一个实数值。

GetHistMaxTime

此函数用来获取某段时间中历史数据的最大值对应的时间

语法使用格式：

RealResult=GetHistMaxTime(TagName,StartTime,EndTime);

参数描述：

TagName：所要查询的变量的名称，类型为字符串型，即带引号。

StartTime：数据查询的开始时间，该时间是通过组态王 HTConvertTime 函数转化的以 1969 年 12 月 31 日 16：00：00 为基准的长整型数，所以用户在使用本函数之前，应先将查询起始时间转换为长整型数值。

EndTime: 数据查询的结束时 间，类型同 StartTime。

返回值： 返回一个字符串，格式为 2008/10/10 11:11:11。

GetHistMinTime

此函数用来获取某段时间中历史数据的最小值对应的时间

语法使用格式：

RealResult=GetHistMinTime(TagName,StartTime,EndTime);

参数描述：

TagName：所要查询的变量的名称，类型为字符串型，即带引号。

StartTime：数据查询的开始时间，该时间是通过组态王 HTConvertTime 函数转化的以 1969 年 12 月 31 日 16：00：00 为基准的长整型数，所以用户在使用本函数之前，应先将查询起始时间转换为长整型数值。

EndTime:数据查询的结束时间，类型同 StartTime。

返回值：返回一个字符串，格式为 2008/10/10 11:11:11。

GetKey

此函数为获得组态王当前使用的加密锁的序列号。语法使用格式

KeyID=GetKey();

该函数没有任何参数。返回值为字符串型。

GetPictureScrollXPos

此函数用于获取目标画面当前窗口的左上角的 X 坐标值。语法使用格式

GetPictureScrollXPos(STRING PictureName);

参数描述

PictureName：画面名称

返回值为目标画面当前窗口的左上角的 X 坐标值。

 例如：

xx=GetPictureScrollXPos("监控中心");

GetPictureScrollYPos

此函数用于获取目标画面当前窗口的左上角的 Y 坐标值。语法使用格式

GetPictureScrollYPos(STRING PictureName);

参数描述

PictureName：画面名称

返回值为目标画面当前窗口的左上角的 Y 坐标值。

 例如：

yy=GetPictureScrollYPos ("监控中心");

GetProjectPath

此函数用于获取当前工程的路径。语法使用格式

GetProjectPath();

例如：在组态王数据词典中定义内存字符串变量 VarName：

```
VarName=GetProjectPath();
```

GetRealDBForBool

此函数用于获取变量的当前实时离散值。语法使用格式

```
GetRealDBForBool( "VarName");
```

参数描述

VarName：变量名称，字符串格式

说明：变量只能是离散量。

返回值为离散型。

例如：

```
bb= GetRealDBForBool( "原料油出料阀");
```

GetRealDBForFloat

此函数用于获取变量的当前实时模拟值。语法使用格式

```
GetRealDBForFloat ("VarName");
```

参数描述

VarName： 变量名称，字符串格式

说明：变量只能是实型变量。

例如：

```
ff= GetRealDBForFloat ( "原料油液位");
```

GetRealDBForInt

此函数用于获取变量的当前实时整型值。语法使用格式

GetRealDBForInt ("VarName");

参数描述

VarName： 变量名称，字符串格式

说明：变量只能是整型变量。

例如：

ii= GetRealDBForInt ("水果");

GetRealDBForString

此函数用于获取变量的当前实时字符串型值。语法使用格式

GetRealDBForString ("VarName");

参数描述

VarName： 变量名称，字符串格式

说明：变量只能是字符串型变量。

例如：

ss= GetRealDBForString ("记录日期");

GetRDBData

此函数可以取得某个时间点上存储于工业库的历史数据。语法格式如下：

GetRDBData(const char* strTagName, short year, short mon, short day, short hour, short min, short sec);

参数：

const char* strTagName:变量名称，格式为“\\服务器名\变量名”，比如\\127.0.0.1\aa_RAD_ss；short year:年、short mon:月、short day:日、short hour:时、short min:分、short sec:秒。

例如：

GetRDBData（“\\127.0.0.1\aa_RAD_ss”,2010,11,1, 12,00,00）；

获取服务器名为 127.0.0.1 上的变量 aa_RAD_ss 在 2010 年 11 月 1 日 12:00:00 时刻的数据。（如果当前时刻没有记录，则自动往前时刻寻找最近的数据；如果之前一直没有数据或者找到的数据质量戳不为 GOOD，则返回“---”）。

GetRDBStatisData

此函数用来获得某时间段的统计数据,可以在用户脚本中使用。返回值类型为字符串。

语法条用格式：

String GetRDBStatisData("variable name", "StartTime", timespan, statistic type,"EndTime");

参数说明：

variable name：变量名称，格式为“\\服务器名\变量名”，比如：“\\127.0.0.1\aa_RAD_ss”

StartTime：查工业库数据的开始时间，格式为“年-月-日-时-分-秒”，如：“2010-6-23-12-00-00”为 2010 年 6 月 23 日 12 时 00 分 00 秒。输入其他格式提示

错误。

timespan：取值时间长度，整型，单位为秒。

StatisType：统计类型。0-最小值，1-最大值，2-平均值。输入其它值提示错误。

EndTime：查工业库数据的结束时间，格式为“年-月-日-时-分-秒”，如：“2010-6-23-23-00-00”为 2010 年 6 月 23 日 23 时 00 分 00 秒。

例如：

GetRDBSatisData(“\\127.0.0.1\aa_RAD_ss”，“2010-11-1-12-00-00”，10，2，“2010-11-1-12-5-0”)，表示 2010-11-1 12:0:0 至 2010-11-1 12:5:0 时间长度为 5 分钟每隔 10 秒取一个 127.0.0.1 上变量 aa_RAD_ss 的瞬时值，再用这些值求平均值。

注意：如果开始与结束时间间隔过大，取值时间长度过小，导致需要查询的数据量过大，程序等待结果的时间将变长，无法进行其他操作。请选取好开始时间，结束时间与取值时间长度。

GetStationStatus

此函数用于在组态王进行网络历史数据备份合并时获得备份的状态。使用时需要通过命令语言调用来获得状态值。语法使用格式：

BOOL GetStationStatus(str szStationName);

参数：szStationName 远程站点名称。

返回值：离散型， >0 正在备份数据 =0 空闲。

例如：

备份状态= GetStationStatus（“IO 采集站”）;

GetStatisData

此函数用来获得某时间段的统计数据，可以在用户脚本中使用。返回值类型为字符串。语法调用格式：

String GetStatisData("variable name", "StartTime", timespan, statistic type,"EndTime");

参数说明：

variable name:变量名称，字符串格式。

StartTime:查历史库数据的开始时间，格式为“年-月-日-时-分-秒”，如：“2010-6-23-12-00-00”为 2010 年 6 月 23 日 12 时 00 分 00 秒。输入其他格式提示错误。

timespan:取值时间长度，整型。

StatisType:统计类型。0-最小值，1-最大，、2-平均值。输入其它值提示错误。

EndTime：查历史库数据的结束时间，格式为“年-月-日-时-分-秒”，如：“2010-6-23-23-00-00”为 2010 年 6 月 23 日 23 时 00 分 00 秒。

例如：

GetSatisData(“本站点\液位”，“2010-4-17-1-0-0”，5，2，“2010-4-17-4-0-0”)，表示 2010-4-17 1:0:0 至 2010-4-17 4:0:0 时间长度为 3 个小时每隔 5 分钟取一个值的平均值。

注意：

1. 此函数在脚本中用时，仅支持对本地历史数据的查询。

2. 如果开始与结束时间间隔过大，取值时间长度过小，导致需要查询的数据量过大，程序等待结果的时间将变长，无法进行其他操作。请选取好开始时间，结束时间与取值时间长度。

GetStruct

此函数的作用是使用工程中定义好的一个结构变量对另一个结构变量赋值。调用格式：

GetStruct(TagName, TempName);

参数说明：

TagName：源结构变量名称，字符串类型。

TempName：目标结构变量名称，系统中已定义的结构变量。

上面两个参数都必须是本站点结构变量，而且使用的是相同结构。

返回值：类型为整型，0-------成功；

-1-------结构变量赋值错误；

-2-------结构变量名称错误或者不存在；

例如：

定义结构：储料罐，包含成员：压力，温度。定义结构变量：储料罐 1，储料罐 1 备份。把储料罐 1 的数据拷贝到储料罐 1 备份中，并把返回值赋给 nRet：

nRet = GetStruct（“储料罐 1”,储料罐 1 备份);

GetTime

此函数将以秒为单位的长整型数转换为相应的时间数值，分别以时、分、秒等的时间数值输出。该长整型秒数的基准为 UTC(格林尼治)时间 1970 年 1 月 1 日 00:00:00。转换完成输出的时间也为 UTC(格林尼治)时间。语法使用格式

GetTime(DateTime,Hour,Minute,Second);

参数描述

DateTime： 需要进行时间转换的数，整型，为输入参数

Hour： 时，整型，转换后得到的数据，输出参数

Minute：分，整型，转换后得到的数据，输出参数

Second： 秒，整型，转换后得到的数据，输出参数

 例如：

自 UTC(格林尼治)时间 1970 年 1 月 1 日 00:00:00 到 2003 年 8 月 8 日 0:8:9 的秒的数值为 1060301289，使用 GetTime ()函数可以从这个数值中分离出所表示的日期——时、分、秒。

函数 GetTime (1060301289,时,分,秒)； 执行后，得到的“时”的值为 0，“分” 的值为 8，“秒” 的值为 9。

获得其中日期的函数为 GetDate ()。

GetTimelocal

此函数将以秒为单位的长整型数转换为相应的时间数值，分别以本地的时、分、秒的时间数值输出。该长整型秒数的基准为 UTC(格林尼治)时间 1970 年 1 月 1 日 00:00:00。转换完成输出的时间为本地时间。语法使用格式

GetTimelocal(DateTime,Hour,Minute,Second);

参数描述

DateTime： 需要进行时间转换的数，整型，为输入参数

Hour： 时，整型，转换后得到的数据，输出参数

Minute：分，整型，转换后得到的数据，输出参数

Second： 秒，整型，转换后得到的数据，输出参数

 例如：

使用 HTConvertTime 函数将自 UTC(格林尼治)时间 1970 年 1 月 1 日 00:00:00 到北京时间2006年9月26日9:22:4转换为以秒为单位的长整数数值为1159233724，使用 GetTimelocal 函数可以从这个数值中分离出所表示的北京时间——时、分、秒。

GetTimelocal (1159233724,时,分,秒);

执行后，得到的“时”的值为 9，“分” 的值为 22，“秒” 的值为 4。

HidePicture

此函数用于隐藏正在显示的画面，但并不将其从内存中删除。调用格式：

HidePicture("画面名");

参数：画面名称 字符串型

 例如：

HidePicture("反应车间");

HistoryDBServerRun

此函数用来改变历史服务记录开停的状态。无返回值。

调用格式：HistoryDBServerRun(is_enable);

参数说明：Is_enable：整型，表示是否开启历史服务记录。如果与当前的历史服务开停状态不同，则函数执行，在信息窗口中出现提示。如果相同，函数直接返回。

 例如：

当前历史服务记录状态是开启，要想关闭历史服务记录，使用函数：HistoryDBServerRun（0)；关闭当前的历史服务记录，信息窗口输出“停止记录历史数据”。

HTConvertTime

此函数将指定的时间格式（年，月，日，时，分，秒）转换为以秒为单位的长整型数，转换的时间基准是 UTC(格林尼治)1970 年 1 月 1 日 00:00:00。例：北京为东八区，那么转换的时间基准为 1970 年 1 月 1 日 8:00:00。

语法使用格式

HTConvertTime(Year,Month,Day,Hour,Minute,Second);

参数描述

Year：	年，整型，此值必须介于 1970 和 2019 之间
Month：	月，整型，此值必须介于 1 和 12 之间
Day：	日，整型，此值必须介于 1 和 31 之间
Hour：	小时，整型，此值必须介于 0 和 23 之间
Minute：	分钟，整型，此值必须介于 0 和 59 之间
Second：	秒，整型，此值必须介于 0 和 59 之间

返回值：整型

注意：

调用此函数将用年、月、日、时、分、秒表示的时间转换成自 1970 年 1 月 1 日 00:00:00 即 UCT 起到该时刻所经过的秒数。在定义返回值变量时，应注意将其最大值置为整型数的最大范围，如 2×10^9，否则可能会因为返回数据超出范围导致转换的时间不正确。

例如：

语句 HTConvertTime(1970,1,1,9,0,0)执行后返回长整型数为 3600;

HTGetPenName

此函数返回指定趋势的指定笔号当前所用的变量名。调用格式：

MessageResult=HTGetPenName(HistoryName,PenNum);

参数描述

HistoryName 历史趋势变量，代表趋势名称。

PenNum 表示笔号的整型变量或整数值(从 1 到 8)。函数将返回代表此指定笔的字符串变量。

例如：

用变量名 Trend1 检索趋势笔 Pen2 的变量名，并将结果放在字符串变量 TrendPen 中。调用函数

TrendPen=HTGetPenName(Trend1 , 2);

HTGetPenRealValue

此函数用于获取指定历史趋势曲线中的趋势笔所对应的实际值。调用格式：

HTGetPenRealValue(HistroyName,PenNum,ContentString);

参数说明

HistroyName 指在“历史趋势曲线”对话框中定义的历史趋势曲线名称

PenNum 与历史趋势曲线中的一个变量相对应的趋势笔的索引号

ContentString 字符串常量

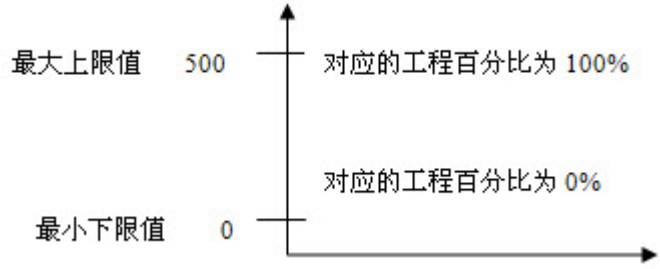
“start” 表示获取与历史趋势曲线的域ValueStart相对应的实际值，ValueStart是用工程百分比来表示变量的下限值，start则表示将下限值的工程百分比转换为实际值。

“end” 表示获取与历史趋势曲线的域ValueEnd相对应的实际值，ValueEnd是用工程百分比来表示变量的上限值，end则表示将上限值的工程百分比转换为实际值。

 例如:

设有一温度历史曲线，其最大上限值为 500，最小下限值为 0，

如下图所示：



如果用 ValueStart 和 ValueEnd 输出显示，则显示的数据是温度值的工程百分比，如 ValueEnd 的输出为 50，表示百分比是 50%，如果使用函数语句

HTGetPenRealValue(histroy,1,“end”);

则函数返回工程百分比 50%对应的实际值 500*50%=250，其中 history 为历史趋势曲线名，1 表示对应温度的趋势笔。

HTGetTimeAtScooter

此函数返回一个长整数，表示以 GMT(格林尼治时间)1970 年 1 月 1 日 00:00:00 为起点（北京时间为 1970 年 1 月 1 日 08:00:00）的以秒计的相对时间，指示器位置由 ScootNum 指定。调用格式：

IntegerResult=HTGetTimeAtScooter(HistoryName,ScootNum);

参数描述

HistoryName 历史趋势变量，代表趋势名。

ScootNum 整数，代表左或右指示器(1=左指示器，2=右指示器)。

当趋势曲线的 ChartStart、ChartLength、ScootNum 或指示器位置改变时都会引起此表达式被计算。

例如：

下面的语句在趋势曲线 Trend1 左指示器的当前位置给出以秒为单位的时间值：

TimeLength=HTGetTimeAtScooter(Trend1 ,1);

HTGetTimeStringAtScooter

此函数返回包含时间/日期的字符串，指示器的位置由 ScootNum 和 ScootLoc 指定。调用格式：

MessageResult=HTGetTimeStringAtScooter(HistoryName,ScootNum,pTextFormat);

参数描述

HistoryName 历史趋势变量，代表趋势名。

ScootNum 整数，代表左或右指示器(1=左指示器，2=右指示器)。

pTextFormat 指定要使用的时间/时期格式的字符串。可为下列值之一。

"Date" 以 Windows 控制面板相同的格式显示日期。

"Time" 以 Windows 控制面板相同的格式显示时间。

"DateTime" 同时显示日期和时间。

当趋势曲线的 ChartStart、ChartLength、ScootNum 或指示器位置改变时都会引起此表达式被计算。字符串的格式决定了返回值的内容。

例如：

在变量为 Trend1 的右指示器的当前位置给出日期/时间值。这个值被存在字符串变量 NewRightTimeString 中，格式是”Time”，调用函数

NewRightTimeString=HTGetTimeStringAtScooter (Trend1, 2, "Time");

HTGetValue

此函数返回一个按整个趋势的指定笔所要求的类型的值。调用格式：

RealResult=HTGetValue(HistoryName,PenNum,ContentString);

参数描述

HistoryName 历史趋势变量，代表趋势名。

PenNum 代表笔号的整型变量或值(1 到 8)。

ContentString 表明返回值类型的字符串。可以是以下字符串之一。

"AverageValue" 整个趋势的平均值。

"MaxValue" 整个趋势的最大值。

"MinValue" 整个趋势的最小值。

此函数返回一内存实型变量，它代表按给出类型算出的值。

 例如：

下面的语句得到趋势 Trend1 的 Pen2 所取得数据的最大值。算出的值存到内存实型变量 LeftHemisphereSD 中：

LeftHemisphereSD=HTGetValue(Trend1,2 , "MaxValue");

HTGetValueAtScooter

此函数返回一个样本在指定的指示器位置、趋势和笔号所要求的类型的的值。调用格式：

RealResult=HTGetValueAtScooter(HistoryName,scootNum,

PenNum,ContentString);

参数描述

HistoryName 历史趋势变量，代表趋势名。

ScootNum 代表左或右指示器的整数(1=左指示器，2=右指示器)。

PenNum 代表笔号的整型变量或值(1 到 8)。

ContentString 代表返回值类型的字符串，可以为以下字符串之一：

"Value" 取得在指示器位置处的值。

"Valid" 判断取得的值是否有效。返回值为 0 表示取得的

值无效，为 1 表示有效。

若是“Value”类型，则返回模拟值。若是“Valid”类型，则返回离散值。

 例如：

采集趋势曲线 Trend1 的笔 Pen3 在右指示器的当前值。若此值有效，则在内存离散变量 ValidFlag 中存入 1，无效，则存入 0：

ValidFlag=HTGetValueAtScooter(Trend1,2,3 , "Valid");

HTGetValueAtZone

此函数返回一包含在某一趋势指定笔的左右指示器之间的数据中所要求类型的值。

调用格式：

RealResult=HTGetValueAtZone(HistoryName,PenNum, ContentString);

参数描述

HistoryName 历史趋势变量，代表趋势名。

PenNum 代表笔号的整型变量或值。(1 到 8)

ContentString 表明要返回值的类型的字符串,可以是以下 3 个字符串之一：

"AverageValue"　　左右指示器间区域上的平均值。

"MaxValue"　　左右指示器间区域上的最大值。

"MinValue"　　左右指示器间区域上的最小值。

此函数直接使用运行数据库的趋势变量的.ScooterPosLeft 和.ScooterPosRight 域来确定区域边界，并返回计算值。


例如：

取得趋势曲线“Trend1”的 Pen1 代表的变量左右指示器之间的平均值,并把结果存入内存实型变量 AvgValue 中。调用函数:

AvgValue=HTGetValueAtZone(Trend1, 1, "AverageValue");

HTResetValueZone

此函数将趋势曲线的数值轴恢复到初始值状态（我们把在组态王开发系统下历史趋势曲线的标识定义中设定的数值轴的起始值和最大值称为数值轴的初始值）。调用格式：

HTResetValueZone(HistoryName);

参数描述

HistoryName　历史趋势变量，代表趋势名。


例如：

将趋势曲线 Trend1 的数值轴恢复到初始状态

HTResetValueZone (Trend1);

HTScrollLeft

此函数将趋势曲线的起始时间左移（提前）给定的百分比值。百分比是相对于趋势曲线的时间轴长度。移动后时间轴的长度保持不变。调用格式：

HTScrollLeft(HistoryName,Percent);

参数描述

HistoryName　历史趋势变量，代表趋势名。

Percent　实数，代表图表要滚动的百分比(0.0 到 100.0)。

 例如：

将趋势曲线 Trend1 的时间轴向左滚动（提前）10%，调用函数:

HTScrollLeft(Trend1,10.0);

若当前显示起始于下午 12:00:00 ，而且显示宽度为 60 秒。在函数执行后，新的趋势曲线将起始于上午 11:59:54。

HTScrollRight

此函数将趋势曲线的起始时间右移给定的百分比值。百分比是相对于趋势曲线的时间轴长度。移动后时间轴的长度保持不变。调用格式：

HTScrollRight(HistoryName,Percent);

参数　描述

HistoryName　历史趋势变量，代表趋势名。

Percent　实数，代表图表要滚动的百分比(0.0 到 100.0)。

 例如：

将趋势曲线 Trend1 的间轴范围向右滚动 20%。调用函数:

HTScrollRight(Trend1,20.0);

若当前显示起始于下午 12:00:00，而且显示宽度为 60 秒，则新的趋势将起始于下午 12:00:12 (在函数执行后)。

HTSetLeftScooterTime

此函数用于设置历史趋势曲线的时间坐标起点。调用格式:

HTSetLeftScooterTime(HistoryName,LeftScooterTime);

参数描述

HistoryName 历史趋势曲线名称，在“历史趋势曲线”对话框中定义。

LeftScooterTime 历史趋势曲线的时间坐标起点值。一个以GMT(格林尼治时间)1970年1月1日00:00:00为起点(北京时间为1970年1月1日08:00:00)的以秒计的相对时间。

HTSetPenName

此函数给一趋势笔赋以不同的变量名。调用格式:

HTSetPenName(HistoryName,PenNum,TagNameString);

参数描述

HistoryName	历史趋势变量，代表趋势名。
PenNum	代表笔号(从 1-8)的整型变量名或整数值。
TagnameString	代表赋给此笔的新变量名，可以使用远程变量名。

 例如:

将趋势曲线 Trend1 的 Pen3 代表的变量设置为 OutletPressure，调用函数

HTSetPenName(Trend1, 3, ”OutletPressure”);

HTUpdateToCurrentTime

此函数将趋势曲线的终止时间设置为当前时间，时间轴长度保持不变。主要用于查看最新数据。调用格式：

HTUpdateToCurrentTime(HistoryName);

参数描述

HistoryName 历史趋势变量，代表趋势名。

 例如：

显示历史趋势曲线“Trend1”在当前时间的最新数据，调用函数

HTUpdateToCurrentTime(Trend1);

若现在是下午 3:04 且趋势宽度为 60 秒，则新的终止时间将为下午 3:04。新的起始时间将为下午 3:03。

HTZoomIn

此函数更改趋势曲线的起始时间和截止时间。通过缩短时间轴长度，以使趋势曲线局部放大。调用格式：

HTZoomIn(HistoryName,AlignPosString);

参数描述

HistoryName 代表趋势名称的历史趋势变量

AlignPosString 代表缩放类型的字符串。可以为以下字符串之一：

"StartTime" 保持起始时间与缩放前相等

"Center" 保持中心时间与缩放前相等

"EndTime" 保持终止时间与缩放前相等

例如：

下面的语句将显示缩小为原来的二分之一，并保持趋势变量“Trend1”的起始时间不变。Trend1.ScorterPosRight 等于 0.0。若缩小前的起始时间为下午 1:25:00，图表宽度为 30 秒，则缩小后起始时间将仍为 1:25:00，而图表宽度则变为 15 秒。调用函数

HTZoomIn(Trend1,"StartTime");

注意：

更改后的起始时间和截止时间与更改前两个指示器的位置（曲线的 ScortPosLeft 和 ScortPosRight 域的值）有关。若更改前指示器的位置不在曲线两端，则更改后宽度为于 ScortPosLeft 与 .ScortPosRight 之间的时间，在此情况下，LockString 值无用。最小的图表宽度为 1 秒。在缩放后，新曲线的 ScootPosLeft 域为 0.0，ScorterPosRight 域为 1.0。

HTZoomOut

此函数计算新曲线的宽度和起始时间，曲线宽度为函数调用前的二倍，新起始时间依 AlignPosString 的值算出。调用格式：

HTZoomOut(HistoryName,AlignPosString);

参数描述

HistoryName　　代表趋势名称的历史趋势变量。

AlignPosString　代表缩放类型的字符串，可为以下三个字符串之一：

"StartTime"　　保持起始时间与缩放前相等

"Center"　　保持中心时间与缩放前相等

"EndTime"　　保持终止时间与缩放前相等

指示器的位置对 HTZoomOut 没有影响，但函数被调用后将置 ScortPosLeft 域为 0.0，ScorterPosRight 域设为 1.0。

 例如：

下面的语句将时间轴长度设置为原来的二倍，并保持趋势变量“Volume”的中心时间不变。若在缩放前的起始时间为下午 12:15:00，图表宽度为 30 秒，则伸缩后的起始时间将变为 2:14:45，图表宽度将变为 60 秒，趋势的中心时间仍为 2:15:15。

HTZoomOut("Volume","Center");

InfoAppActive

此函数测试一个应用程序是否为活动的。调用格式：

DiscreteResult=InfoAppActive("ExeName");

返回值：离散值；

参数描述

ExeName　　应用程序的执行文件名

 例如：

InfoAppActive("Excel.exe");

若返回 1，表明 Excel 程序正在运行；返回 0 表明未运行。

InfoAppDir

此函数返回当前组态王的工程路径。调用格式：

MessageResult=InfoAppDir();

当前组态王工程路径返回给 MessageResult.

例如：

DemoPath=InfoAppDir();

将返回 "C:\Program Files\Kingview\Example\Kingdemo3"。

InfoAppTitle

此函数返回应用程序的标题或者一个当前正在运行的指定程序的 Windows 任务列表名。调用格式：

MessageResult=InfoAppTitle(ProgramEXEName);

返回值：字符型值；

参数描述

ProgramEXEName　应用程序的执行文件名。

例如：

InfoAppTitle;("calc.exe"); // 将返回 "Calculator"

InfoAppTitle;("excel.exe"); //将返回 "Microsoft Excel"

InfoDisk

此函数返回指定的本地(或网络)磁盘驱动器信息。调用格式：

IntegerResult=InfoDisk(Drive,InfoType,Trigger);

参数描述

Drive 代表驱动器号的字符串或字符串变量。若提供的字符串变量包含多于一个的字符，则只使用此变量的首字符。

InfoType代表信息类型的整数，可为以下两个值之一：

1 返回磁盘驱动器的总空间数(以字节计)。

2 返回磁盘驱动器上可用的空闲空间数(以字节计)。

Trigger 每当 Trigger 的值改变时，执行 InfoDisk() 函数。Trigger 可为任何变量名(不受系统变量的限制)。

由驱动器号指定的磁盘驱动器的有关信息返回给 IntegerResult。

 例如：

下面的语句每分钟执行一次并返回当前的值：

InfoDisk("C", 1, $分);//将返回 C 盘总空间数

InfoDisk("C", 2, $分);//将返回 C 盘空闲空间数

InfoFile

此函数返回指定文件或子目录的有关信息。调用格式：

IntegerResult=InfoFile(Filename,InfoType,Trigger);

参数描述

Filename 代表要处理的文件名的字符串。

InfoType 代表要获取的信息的类型的整数，可为以下值之一：

1 查找文件是否存在。若文件名是一个实际文件，返回 1。若找不到文件则返回 0。

2 文件大小（字节数）。

3 文件日期/时间（自 1970 年 1 月 1 日起的相对秒数）

4 与文件名描述相匹配的文件数。仅当使用通配符查找并找到多个匹配的文件时，返回值大于 1。

Trigger 为任一变量名，每当 Trigger 的值改变时，将执行 InfoFile() 函数。

由文件名指定的文件的有关信息返回给 IntegerResult。文件名必须包括文件的完整路径，可包含通配符(*，?)。

 例如：

下面的语句每分钟执行一次并返回下列值：

InfoFile("c:\kingview\touchvew.exe", 1, $分);将返回 1，{文件找到}

InfoFile("c:\kingview\touchvew.exe", 2, $分);将返回 634960，{文件大小}

InfoFile("c:\kingview\touchvew.exe", 3, $分);将返回 736701852， {自 70 年 1 月 1 日起的秒数}

InfoFile("c:\kingview*.exe", 4, $分);将返回 4，{找到 4 个可执行文件}。

InfoResource

此函数返回各种系统资源值。调用格式：

IntegerResult=InfoResource(ResourceType,Trigger);

参数描述

ResourceType 代表要监视的资源类型的整数，可为以下值之一：

1　　返回 GDI 资源可用空闲空间的百分比。

2　　返回 USER 资源可用空闲空间的百分比。

3　　返回当前内存中空闲空间字节数。

4　　返回当前正在运行的任务数。

Trigger　　每当 Trigger 值改变时，执行 InfoResource() 函数。Trigger 可为任一变量名(不受系统变量限制)。

由整数 ResourceType 指定的特定系统资源信息存放在 IntegerResult 中。

例如：

下面的语句每分钟执行一次并返回当前值：

```
InfoResource(1, $分);//将返回空闲百分比
InfoResource(2, $分);//将返回空闲百分比
InfoResource(3, $分);将返回内存中空闲空间字节数
InfoResource(4, $分);将返回任务数
```

注意:

在 WIN NT 下返回 GDI 和 USER 的资源可用空闲空间的百分比是一样的,与 WIN NT 系统有关。

Int

此函数返回小于等于指定数值的最大整数。调用格式：

IntegerResult=Int(Number);

参数描述

Number　任一数字或者组态王的实型或整型变量名。

例如：

Int(4.7);将返回 4

Int(-4.7);将返回 -5

listLoadList

此函数用于将CSV文件Filename中的列表项调入指定的列表框控件ControlName中，并替换列表框中的原有列表项。列表框中只显示列表项的成员名称（字符串信息），而不显示相关的数据值。语法格式使用如下：

listLoadList("ControlName","Filename");

参数说明：

ControlName：工程人员定义的列表框控件名称，可以为中文名或英文名。

Filename：csv 文件，用写字板程序进行编辑，用以存放列表框中要显示的列表项。

例如：

listLoadList("组合框信息","c:\组态王\list.csv");

此语句将指定的文件 list.csv 调入名为组合框信息的列表框中并显示出来。

注：

如果没有给出 csv 文件所在的完整路径，则该函数就从组态王所在的路径下寻

找指定的文件。

listSaveList

此函数用于将列表框控件 ControlName 中的列表项信息存入 CSV 文件 Filename 中。如果该文件不存在，则直接创建。语法格式使用如下：

listSaveList("ControlName","Filename");

参数说明：

ControlName：工程人员定义的列表框控件名称，可以为中文名或英文名。

Filename：CSV 文件，按一定格式用以存放列表框中的列表项。

例如：

listSaveList("组合框信息","c:\组态王\list.csv");

此语句将组合框信息列表框中的列表项存入到文件 c:\组态王\list.csv 中。

注：

如果没有给出 CSV 文件所在的完整路径，则该函数在组态王所在的路径下创建该文件。

listAddItem

此函数将给定的列表项字符串信息 MessageTag 增加到指定的列表框控件 ControlName 中并显示出来。组态王将增加的字符串信息作为列表框中的一个成员项 Item，并自动给这个成员项定义一个索引号 ItemIndex，索引号 ItemIndex 从 1 开始由小到大自动加 1。语法格式如下：

listAddItem("ControlName","MessageTag");

参数说明：

ControlName：工程人员定义的列表框控件名称，可以为中文名或英文名。

MessageTag：字符串值，表示增加到指定列表框控件的成员项字符串信息。

例如：

listAddItem("报警信息","温度报警");

此语句将“温度报警” 字符串信息增加到列表框控件报警信息中并显示出来。

listAddItem("配方信息","巧克力面包");

此语句将“巧克力香型面包” 字符串信息增加到列表框控件配方信息中并显示出来。

listClear

此函数将清除指定列表框控件 ControlName 中的所有列表成员项。语法格式如下：

listClear("ControlName");

参数说明：

ControlName：工程人员定义的列表框控件名称，可以为中文名或英文名。

例如：

listClear("报警信息");

此语句将清除报警信息列表框中的所有列表成员项。

listDeleteItem

此函数将在指定的列表框控件 ControlName 中删除索引号为 ItemIndex 的成员项。

语法格式如下：

listDeleteItem("ControlName",ItemIndex);

参数说明：

ControlName：工程人员定义的列表框控件名称，可以为中文名或英文名。

ItemIndex：列表框控件中的成员项索引号，通常为数字常量或整型变量。

例如：

listDeleteItem("报警信息",1);

此语句将在报警信息列表框中删除索引号为 1 的成员项。

listDeleteItem("配方信息",5);

此语句将在配方信息列表框中删除索引号为 5 的成员项。

listDeleteSelection

此函数将删除列表框控件 ControlName 中当前选定的成员项。语法格式如下：

listDeleteSelection("ControlName");

参数说明：

ControlName：工程人员定义的列表框控件名称，可以为中文名或英文名。

例如：

listDeleteSelection("报警信息");

此语句将在报警信息列表框中删除当前选定的成员项。

listFindItem

此函数用于查找与给定的成员字符串信息 MessageTag 相对应的索引号，并送给整型变量 IndexTag。语法格式如下：

listFindItem("ControlName","MessageTag",IndexTag);

参数说明：

ControlName：工程人员定义的列表框控件名称，可以为中文名或英文名。

MessageTag：字符串值，表示列表成员项字符串信息。

IndexTag：整型变量，用以存放与给定的成员字符串信息 MessageTag 相对应的索引号.

 例如：

以 CSV 文件 list.csv 中存放的列表项信息为例如，

listFindItem("组合框信息","温度", IndexTag);

此语句将“温度” 字符串信息相对应的索引号送给整型变量 IndexTag。在此例如中 IndexTag=1。

listGetItem

此函数用于获取索引号为 ItemIndex 的列表项成员字符串信息，并送给字符串变量 StringTag。语法格式如下：

listGetItem("ControlName",ItemIndex,StringTag);

参数说明：

ControlName：工程人员定义的列表框控件名称，可以为中文名或英文名。

ItemIndex：数值常量或变量，表示列表索引号。

StringTag：字符串变量，用以存放索引号为 ItemIndex 的列表项成员字符串信

息。

 例如：

以 CSV 文件 list.csv 中存放的列表项信息为例如，

listGetItem("组合框信息",2, StringTag);

此语句将索引号为 2 的列表项成员字符串信息字符串变量 StringTag。在此例如中，StringTag=压力。

listGetItemCount

此函数用于获取指定控件“ControlName”中列表项的数目。语法格式如下：

listGetItemCount("ControlName");

参数说明：

ControlName：工程人员定义的列表框控件名称，可以为中文名或英文名。

 例如：

Count=listGetItemCount("list");

将列表框控件“list”中列表项的数目赋给变量 Count。

ListGetCurSel

此函数用于获取指定控件“ControlName”当前选中列表项的 ID 号(从 0 开始)，返回值为-1 说明当前控件没有选中项目。语法格式如下：

listGetCurSel("ControlName");

参数说明：

ControlName：工程人员定义的列表框控件名称，可以为中文名或英文名。

例如：

ItemIndex=listGetCurSel("list");

将列表框控件“list”中当前选中的列表项的 ID 号赋给变量 ItemIndex。

ListSetCurSel

此函数用于将控件中索引号为 ItemIndex 的列表项设置为当前选中项，返回值为控件中当前选中项的索引号(从 0 开始)。如果返回值为-1，说明没有设置成功（ItemIndex 为负数或超过列表项数)。语法格式如下：

listSetCurSel("ControlName",ItemIndex);

参数说明：

ControlName：工程人员定义的列表框控件名称，可以为中文名或英文名。

ItemIndex: 数值常量或变量，表示列表项索引号。

例如：

ID=listSetCurSel("list",1);

设置列表框控件“list”当前选中项为索引号为 1 的列表项。并将当前选中列表项的索引号赋给变量 ID。

listGetItemData

此函数用于获取索引号为 ItemIndex 的列表项中的数据值，并送给整型变量 NumberTag。语法格式如下：

listGetItemData("ControlName",ItemIndex,NumberTag);

参数说明：

ControlName：工程人员定义的列表框控件名称，可以为中文名或英文名。

ItemIndex：数值常量或变量，表示列表索引号。

NumberTag：整型变量，用以存放索引号为 ItemIndex 的列表项的数据值。

例如：

以 CSV 文件 list.csv 中存放的列表项信息为例如，

listGetItemData("组合框信息",2, NumberTag);

此语句将索引号为 2 的列表项的数据值送给变量 NumberTag。在此例如中，NumberTag=40。

listInsertItem

此函数将字符串信息 StringTag 插入到列表项索引号 ItemIndex 所指示的位置。如果 ItemIndex=-1，则字符串信息 StringTag 被插入到列表项的最尾端。语法格式如下：

listInsertItem("ControlName",ItemIndex, "StringTag");

参数说明：

ControlName：工程人员定义的列表框控件名称，可以为中文名或英文名。

ItemIndex：数值常量或变量，表示列表索引号。

StringTag：字符串常量。

例如：

listInsertItem("组合框信息",2, “炉温”);

此语句将字符串信息“炉温”插入到索引号 2 所指示的位置。

listSetItemData

此函数用于将变量 Number 的值设置索引号为 ItemIndex 的列表项中。语法格式如下：

listSetItemData("ControlName",ItemIndex, Number);

参数说明：

ControlName：工程人员定义的列表框控件名称，可以为中文名或英文名。

ItemIndex：数值常量或变量，表示列表索引号。

Number：整型变量，用以存放数据值。

 例如：

listSetItemData("组合框信息",2, Number);

此语句将变量 Number 中的值设置到索引号为 2 的列表项。

ListLoadFileName

此函数将字符串*.ext 指示的文件名显示在列表框中。

ListLoadFileName("CtrlName","*.ext");

参数说明：

CtrlName: 工程人员定义的列表框控件名称，可以为中文名或英文名。

*.ext: 字符串常量，工程人员要查询的文件，支持通配符。

 例如：

ListLoadFileName(“报警文件列表”, “c:\appdir\alarm*.al2”);

此语句将 c:\appdir\alarm 目录下的后缀为.al2 的文件名显示在列表框中。

LoadDriverConfig

根据“sDriverFilePath”中指定的参数数据库路径和文件名称及“nRow”指定的要下载的参数的段号，此函数用于将指定控制段的一个或全部数据一次性写入控制设备中。此函数为组态王公用函数。调用格式：

LoadDriverConfig(sDriverFilePath, nRow);

参数说明：sDriverFilePath： 字符串型 指定的参数数据库路径和文件名称

nRow： 整型 要下载的参数的段号，当 nRow＝9999 时，下载驱动配置表中指定的所有参数的所有数据。

返回值：整型

1 表示下载失败（此处的下载失败只指组态王读取数据错误或驱动配置校验错误，而不是指数据写设备成功）

0 表示下载成功

-1 打开数据库文件失败

-2 驱动配置表中无记录

-3 获得控制字段名失败

-4 根据字段得到驱动信息失败

-5 为未定义

-6 为未定义

-7 为未定义

-8 为未定义

-9 驱动配置信息无参数

-10 为未定义

-11 下载点数超过限制

-12 表没有被记录

-13 非温控版不支持数据下载

-14 演示模式不支持数据下载

 例如：

1、下载驱动配置中指定的段号为 1 的参数

```
long lRet;
String strDBPath="E:\数据库\control.mdb";
long nRow=1;
lRet=LoadDriverConfig（strDBPath，nRow）；
```

2、下载驱动配置中指定的全部参数

```
long lRet;
String strDBPath="E:\数据库\control.mdb";
lRet=LoadDriverConfig（strDBPath，9999）；
```

LoadText

此函数将指定的 RTF 或 TXT 格式文件调入到超级文本显示控件中加以显示。语法格式如下：

```
LoadText( "ControlName", "FileName", ".Txt Or .Rtf" );
```

参数说明：

ControlName：工程人员定义的超级文本显示控件名称，可以为中文名或英文名。

FileName：RTF 或 TXT 格式的文件，可用 WINDOWS 的写字板编写这两种格式的文件。

.Txt Or .Rtf：指定文件为 RTF 格式或 TXT 格式。

例如：

LoadText("hypertext1","D:\Test\recipe\ht1.rtf", ".Rtf");

此语句把 RTF 格式的文件 ht1.rtf 调入到名称为 hypertext1 的超级文本显示控件中加以显示。

LogE

此函数返回对数函数 $\log_e^x$ 的计算结果，X 为变量值，调用格式：

LogE(变量值);

例如：

LogE(100);返回 loge100 计算值 4.605

LogE(1);返回 loge1 计算值 0

LogN

此函数返回以 n 为底的 x 的对数。以 1 为底的对数没有定义。调用格式：

Result=LogN(Number,Base);

参数描述

Number　　任一数字或者组态王的实型或整型变量名。

Base　　做底的整数。

例如：

LogN(8, 3);将返回 1.89279...

LogN(3, 7);将返回 0.564...

LogOff

此函数用于在 TOUCHVEW 中退出登录。调用格式：

LogOff( );

参数　　无

LogOn

此函数用于在 TouchVew 中登录。调用格式：

LogOn( );

参数　　无

例如：

为画面上某个按钮建立命令语言连接：

LogOn( );

画面程序运行时单击此按钮，弹出“登录”对话框：

工程人员在此对话框中输入用户名和口令，以获得操作权限。

LogOnEx

此函数用于在 Touchvew 中登录。区别于 LogOn 函数，不弹出“登录”对话框。调用格式：

LogOnEx("UserName", "PassWord");

参数：UserName 用户名

PassWord 密码

LogString

此函数写一个工程人员自定义消息到组态王，信息将被输出到组团瓦信息窗口中。也可以使用 Trace()函数实现。调用格式：

LogString(String);

参数　　描述

String　要记录到组态王的字符串。

例如：

LogString("Report Script is Running");

Max

此函数用于求得给定的数中最大的一个数。其参数个数为 1-16 个。调用格式：

Max(Val1, Val2);

例如：

MaxValue = Max(Max(var1,var2), var3);

此函数返回值 MaxValue 为 var1、var2、var3 中最大的数。

Min

此函数用于求得给定的数中最小的一个数。其参数个数为 1-16 个。

例如：

MinValue=Min(Min(var1,var2),var3);

此函数返回值 MinValue 为 var1、var2、var3 中最小的数。

ModifyTagField

此函数用于修改变量的属性值

语法格式使用如下：

ModifyTagField("VarName",Value,Quality,Year,Month,Date,Hour,Minute,Second,millSecond);

参数说明：

VarName：变量名称。

Value： 变量值。

Quality：变量的质量戳。

Year：变量时间戳的年。

Month：变量时间戳的月。

Date：变量时间戳的日。

Hour：变量时间戳的时。

Minute：变量时间戳的分。

Second：变量时间戳的秒。

millSecond：变量时间戳的毫秒。

 例如：

ModifyTagField("R1",999.9,192,2005,10,1,9,0,0,0);

 注意:

1. 设置的时间戳数值需符合组态王设定的规则。

2. 只有函数设定的变量值不是当前值时，质量戳设值才能成功；只有质量戳的设值为好（即 192）时，时间戳的设值才能成功。

MovePicture

此函数用于在系统运行时通过命令语言脚步来移动画面到所在的位置。语法格式如下：

MovePicture(PicName,left,top);

参数描述

PicName： 要移动画面的画面名称，字符串型

Left： 画面移动目标位置——画面的左边界坐标，整型

Top： 画面移动目标位置——画面的上边界坐标，整型

 例如：

MovePicture(“信息提示”,50,100);

将画面“信息提示”移动到左边界距离坐标50，上边界距离坐标100的位置。

PageDown

用于报警窗口信息的向前翻页显示。调用形式：

PageDown(AlmWin, Lines);

参数：

AlmWin：报警窗口名

Lines： 翻页行数

 例如：

PageDown(全厂历史报警记录窗口,7);

该调用将“全厂历史报警记录窗口”的报警记录向下翻 7 行（如果有足够报警记录的话）。

PageUp

用于报警窗口信息的向后翻页显示。调用格式：

PageUp(AlmWin, Lines);

参数：

AlmWin：报警窗口名

Lines： 翻页行数

 例如：

PageUp(全厂历史报警记录窗口,7);

该调用将“全厂历史报警记录窗口”的报警记录向上翻 7 行（如果有足够报警记录的话）.

PI

此函数返回圆周率的值。调用格式：

RealResult=PI();

 例如：

PI( );将返回 3.1415926...

PlayAvi

此函数用于播放动画，动画为.avi 文件。调用格式：

PlayAvi("CtrlName",filename,option);

参数及其描述

CtrlName:用于播放播放 AVI 动画的控件的名称。

filename:代表要播放的动画文件的字符串或字符串变量。

option:可为下述之一：

0　　停止播放 AVI 动画

1　　播放一遍 AVI 动画

2　　连续播放 AVI 动画，直到接收到停止播放的信息为止


例如：

PlayAvi("ctl_avi","c:\demo\Winner.avi",1);

此函数的功能是在名称为“ctl_avi”的控件中播放 Winner.avi 中存放的动画，只播放一次。画面停止在动画的最后一帧。

PlaySound

此函数通过安装了 Windows wave 形式音频设备驱动器的机器播放声音。声音为 wav 文件。调用格式：

PlaySound(SoundName,Flags);

参数说明：

SoundName：代表要播放的声音文件的字符串或字符串变量。声音文件名前可加声音文件所在的目录，也可以不加。声音文件目录的查找按以下顺序：当前工程目录，Windows 目录，Windows 系统目录，在 SoundName 参数中列出的目录。若缺省的目录或列出的目录中找不到该声音文件，则不播放声音。

Flags：Flags 可为下述之一：

0：停止播放声音，注意，使用该值不能停止播放同步声音

1：同步播放声音

2：异步播放声音

3：重复播放声音直到下次调用 PlaySound() 函数为止

4：蜂鸣器报警

5：停止同步播放声音。当前声音完整播放，停止的是当前声音之后的声音

例如：

1、同步播放声音文件 a，a 所在路径为当前组态王工程路径下。

PlaySound("a.wav", 1);

2、停止播放同步声音。

PlaySound("", 5);

注意：

停止播放声音（包括同步播放和非同步播放）时，不需要输入 SoundName 参数。

PlaySound2

该函数为播放WAV声音文件。调用格式：

PlaySound2(SoundName, DevideId, Flags);

参数：

SoundName　字符串型　声音文件名称及路径

DevideId　整型　播放声音的声音设备（声卡）序号，如果是第一块声卡，则为1，如果是第二块声卡，则为2。

Flags　整型　对声音文件的控制。如果设为0，则关闭当前播放的音乐；如果设为1，则当前音乐播放一次后停止播放；如果设为2，则当前音乐播放完后继续从头循环播放。

例如：

使用声卡播放一次“c:\horns.wav”声音文件.

PlaySound2(“c:\horns.wav”, 1, 0);

Pow

此函数求得一模拟值或模拟变量的任意次幂。调用格式：

Result=Pow(x, y);

参数描述

x 底数

y 指数

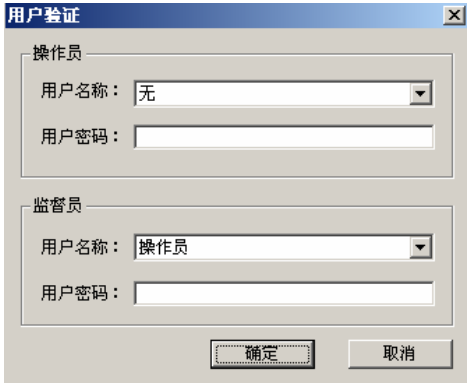
返回值为 x 的 y 次幂。

例如：

Result=Pow(2, 3); 函数调用后 Result=8.0

PowerCheckUser

此函数当用户希望进行一项操作时（如分闸或合闸），为防止误操作，需要进行双重认证。即在身份认证对话框中，既要输入操作者的名称和密码，又要输入监控者的姓名和密码，两者验证无误时方可操作。调用该函数后，弹出身份验证对话框。如下图所示。其中使用的用户信息是通过组态王中的用户配置得到的。



调用格式：

Result= PowerCheckUser(string OperatorName, string MonitorName);

参数描述

OperatorName	返回的操作者姓名
MonitorName	返回的控制者姓名
Result	1：验证成功，0：验证失败

例如：

Result= PowerCheckUser(OperatorName, MonitorName);

PreviewWindow

该函数具有组态王画面的打印预览功能。调用格式：

PreviewWindow("Window", xScale, yScale , option, xStart, yStart);

参数说明：

Window：要打印预览的组态王窗口名，即组态王画面名称。字符串类型。

xScale：打印输出的宽度占页面总宽度的百分比。整型或实型。此参数为 0 时，option 参数起作用。

yScale：打印输出的高度度占页面总高度的百分比。整型或实型。此参数为 0 时，option 参数起作用。

option：仅当 xScale 和 yScale 为 0 时有效。整型，取值 0 或 1。

如果组态王画面上不包含位图，OCX 控件，报表，报警窗，那么，option 为 0 时，保持画面的纵横比不变，以适合打印页面的最大比例打印（画面不失真）；option 为 1 时，按页面的大小对画面进行缩放（画面有可能失真）。

如果组态王画面上包含位图， OCX 控件，报表，报警窗，中的任何一个，那么，option 为 1 时，保持画面的纵横比不变，以适合打印页面的最大比例打印（画面不失真）；option 为 0 时，按页面的大小对画面进行缩放（画面有可能失真）。

当画面上包含位图， OCX 控件，报表，报警窗时，建议将 option 参数设为 1。

xStart：要打印窗口的横向空白占页面宽度的百分比。整型或实型。如果组态王画面上包含位图， OCX 控件，报表，报警窗，中的任何一个时，xStart 参数值无意义。

yStart：要打印窗口的纵向空白占页面高度的百分比。整型或实型。如果组态王画面上包含位图， OCX 控件，报表，报警窗，中的任何一个时，yStart 参数值无意义。

注意：

1、组态王运行系统预览打印画面后，如果在开发系统对画面进行了修改，请重新预览。

2、只能预览当前显示的画面。

3、请在组态王运行系统启动，显示运行画面之后，再进行打印预览。

4、如果画面的显示尺寸小于画面的实际尺寸时（画面的显示尺寸和实际尺寸在开发系统的画面属性中设置），预览到的画面与组态王运行系统显示的画面相同，也就是说，预览到的是部分画面。

5、建议将画面的显示尺寸设在计算机显示屏的像素点以下。

例如：

预览报表打印窗口，可以使用下面的按钮命令语言：

PreviewWindow("报表", 0, 0 , 1, 0, 0);

其中“报表”为包含报表的组态王画面名。

PrintWindow

此函数打印指定窗口。调用格式：

PrintWindow("Window", xScale, yScale , option, xStart, yStart);

参数描述

Window　　要打印的窗口名。

xScale　打印输出的宽度占此页总宽的百分比。此参数可以取 0，以使用缺省最大的纵横比或者取一指定的宽度。

YScale　　打印输出的高度占此页总高度的百分比。此参数可以取 0，以使用缺省最大的纵横比或者取一指定的高度。

Options 离散值：0 或 1，仅在 Width 和 Height 都为 0 时使用。若 Options 为 1 窗口在最大纵横比下以窗口尺寸的整数倍数打印。若 Options 为 0，以适于此页的最大纵横比打印。若窗口包含位图，置 Options 为 1，以免位图被拉长。

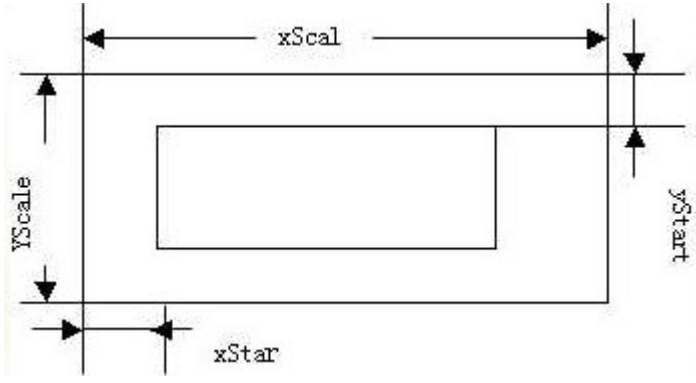
xStart　　要打印的窗口横向空白长度的百分比。

YStart　　要打印的窗口纵向空白长度的百分比。

若要确保窗口中的文本能被正确打印，建议将所有要被打印的窗口中的文体域设置为“True Type”字体。

当打印画面上的按钮时，按钮上的文本中可能被“切除”，因为用在按钮文本上的字体为“System”字体，它不是“True Type”字体。另外，“System”字体用在打印机上与用在屏幕上相比略有不同。若发生了这种情况。请试着把按钮放大。

下图显示了 xScale，YScale，xStart，yStart 之间的关系:



例如：

```
PrintWindow("反应车间 ",10,10,0,80,80);
```

pvAddNewRealPt

此函数用于在指定的温控曲线控件中增加一个采样实时值。

语法格式使用如下：

pvAddNewRealPt("ControlName",timeOffset,Value, "commentTag");

参数说明：

ControlName：工程人员定义的温控曲线控件名称，可以为中文名或英文名。

timeOffset：相对前一采样点的时间偏移量(即距前一值的时间间隔值)，第一个值取 0。

Value：温度的采样值，实型数据，此变量通常为组态王数据库中定义的 I/O 实数变量。

commentTag：注释性字符串，也可以是字符串变量，当游标移动到此点时，给出提示性信息。

 例如 1：

pvAddNewRealPt("反应罐温控曲线", 1, 38, "温度值为 38 度");

此语句在反应罐温控曲线控件中增加一个 38 度的温度采样实时值。此采样实时值距前一值的时间间隔值为 1，当游标移动到此点时，给出提示性信息“温度值为 38 度”。

 例如 2：

设反应罐实时温度是组态王数据库中定义的一个 I/O 实数变量，接收从下位机中送来的温度值，TimeString 为组态王数据库中定义的一个字符串变量。

TimeString=StrFromInt($时)+ ":"+ StrFromInt($分)+ ":"+ StrFromInt($秒)

pvAddNewRealPt("反应罐温控曲线", 10, 反应罐实时温度, TimeString);

此语句在反应罐温控曲线控件中给出变量反应罐实时温度的采样实时值。此采

样实时值距前一值的时间间隔值为 10，当游标移动到此点时，给出 TimeString 中的提示性信息。

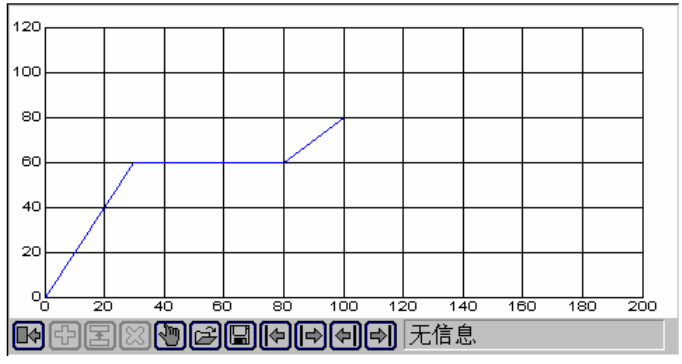
注意：

设定曲线将根据实时曲线第一点的位置而变。

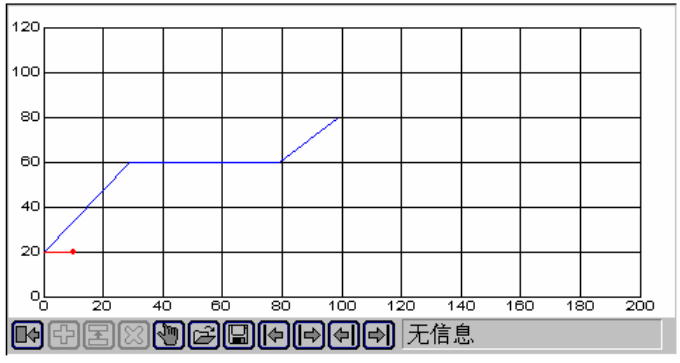
例如 3:

实时曲线第一点的位置为：（用一按钮添加实时曲线第一点）

pvAddNewRealPt("反应罐温控曲线", 10, 20, TimeString);



图中为一设定曲线，按下按钮后，如下图示：



原设定曲线第一点（0，0）变为（0，20）；

pvAddNewSetPt

此函数用于在指定的温控曲线控件中增加一段温度设定曲线。适用于自由设定模式。

语法格式使用如下：

pvAddNewSetPt("ControlName", TimeOffset, Value);

参数说明：

ControlName：工程人员定义的温控曲线控件名称，可以为中文名或英文名。

timeOffset：相对前一采样点的时间偏移量(即距前一值的时间间隔值)，第一个值取 0。

Value：温度的设定值，实型数据。

例如：

pvAddNewSetPt("反应罐温控曲线", 1, 38,);

此语句在反应罐温控曲线控件中增加一段温度设定曲线，其设定温度为 38 度，

此设定值距前一值的时间间隔值为 1。

pvClear

此函数用于在指定的温控曲线控件中删除温度实时曲线和温度设定曲线。

语法格式使用如下：

pvClear("ControlName", IsRealCurve);

参数说明：

ControlName：工程人员定义的温控曲线控件名称，可以为中文名或英文名。

IsRealCurve：布尔型变量。为 0：删除设定曲线和实时曲线；为 1：删除实时曲线。

 例如：

pvClear("反应罐温控曲线", 0);

此语句清除反应罐温控曲线控件中的温度实时曲线和温度设定曲线。

pvClear("反应罐温控曲线", 1);

此语句清除反应罐温控曲线控件中的温度实时曲线。

pvGetValue

此函数用于在指定的温控曲线控件中获取指定时刻的温度设定值或指定时刻的温度实时值，若指定时刻无采样，则返回该时刻前最近的一次采样值。

语法格式使用如下：

pvGetValue("ControlName", timeOffset, TagName, "option");

参数说明：

ControlName：工程人员定义的温控曲线控件名称，可以为中文名或英文名。

timeOffset： 相对时间坐标原点的时间偏移量（也就是绝对时间坐标），第一个值取 0。

TagName： 组态王数据库中定义的 I/O 实数变量。

option： 确定是温度设定值或温度实时值，字符串常量，如下所示。

RealValue 温度实时值

SetValue 温度设定值

 例如：

pvGetValue("反应罐温控曲线",5, 反应罐实时温度,"RealValue");

此语句从反应罐温控曲线控件中获取指定时刻的温度实时值，该值距坐标原点的时间间隔为 5，并将该值存放到变量反应罐实时温度中。

pvGetValue("反应罐温控曲线",5, 反应罐设定温度,"SetValue");

此语句从反应罐温控曲线控件中获取指定时刻的温度设定值，该值距坐标原点的时间间隔为 5，并将该值存放到变量反应罐设定温度中。

pvIniPreCuve

此函数用于初始化设定曲线。

语法格式使用如下：

pvIniPreCuve("ControlName", "fileName");

参数说明：

ControlName：工程人员定义的温控曲线控件名称，可以为中文名或英文名。

fileName：fileName 文件以文本文件格式(.csv),编排格式:

SetData

曲线点数

曲线第一点的位置

第一段升温速率,设定温度,保温时间

第二段升温速率,设定温度,保温时间

第三段升温速率,设定温度,保温时间

...

第 n 段升温速率,设定温度,保温时间

 编排格式注意：

SetData、曲线点数、曲线第一点的位置末和每一段曲线参数升温速率、设定温度、保温时间输完以后，该行末不能加空格或其他符号。

 例如：

设文件 pvset.csv 以.csv 格式存放数据如下:

SetData

20

39.000000

10.000000,20.000000,0

10.000000,20.000000,10

20.000000,20.000000,10

30.000000,20.000000,10

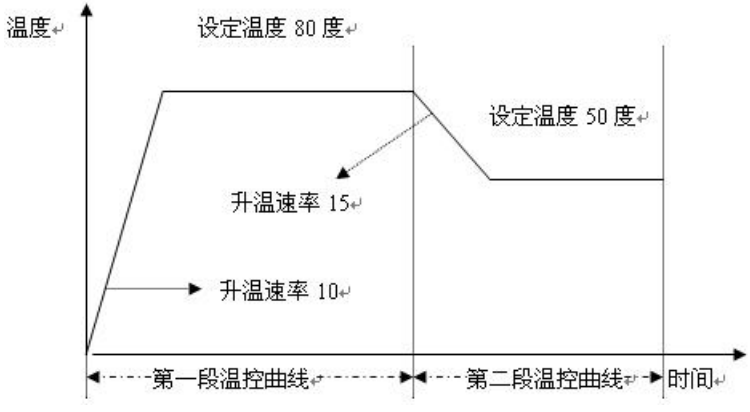
40.000000,20.000000,10

如下图所示，用计事本编写完后用.csv 为后缀存盘，则该文件即可被此控件函数使用。

用记事本编辑格式文件的示意图如下：

pvIniPreCuve("加热炉温控曲线","c:\pvset.csv");

此语句将加热炉的初始化温控曲线设定为下图所示的曲线：



pvLoadData

此函数用于从指定的文件中读取温控设定曲线或温控实时曲线的采样历史数据值，文件名后缀必须为.csv。

语法格式使用如下：

pvLoadData("ControlName", "FileName", "option");

参数说明：

ControlName：工程人员定义的温控曲线控件名称，可以为中文名或英文名。

FileName： FileName 文件以.csv 格式按曲线段数、各段升温速率、设定温度、保温时间依次存放设定温控曲线信息或温控实时曲线的采样历史数据值，文件名后缀必须为.csv

option： 确定是读取温控设定曲线或温控实时曲线的采样历史数据值，字符串常量。

“RealValue” 读取温控实时曲线的采样历史数据值

“SetValue” 读取温控设定曲线

 例如：

设文件以.csv 格式存放数据同初始化设定曲线的文件格式，例如:

SetData

20

39.000000

10.000000,20.000000,0

10.000000,20.000000,10

20.000000,20.000000,10

30.000000,20.000000,10

40.000000,20.000000,10

存为 c:\setvalue.csv ，并用下面的函数调用，在温控曲线控件中显示出来:

pvLoadData("反应罐温控曲线"," c:\setvalue.csv ",

"SetValue");

此语句读取温控设定曲线并传送到反应罐温控曲线控件中显示出来。

pvLoadData("反应罐温控曲线","fileName ","RealValue");

此语句读取温控实时曲线的采样历史数据值并传送到反应罐温控曲线控件中显示出来。

pvModifyPreValue

此函数用于在指定的温控曲线控件中修改某段温控设定曲线。

语法格式使用如下：

pvModifyPreValue("ControlName", Index, Tane, SetValue, timeStore);

参数说明：

ControlName：工程人员定义的温控曲线控件名称，可以为中文名或英文名。

Index： 温控曲线段索引编号。

Tane：设置温控曲线段的升温速率。

SetValue：设置温控曲线段的设定温度。

timeStore：设置温控曲线段的保温时间。

 例如：

pvModifyPreValue("反应罐温控曲线", 2, 20,80,25);

此语句将反应罐温控曲线控件中的第二段温控设定曲线设置为：升温速率 20；设定温度 80；保温时间 25。

pvMoveSlide

此函数用于在指定的温控曲线控件中设置游标左移或右移。

语法格式使用如下：

pvMoveSlide("ControlName", leftORrightSlide, direction, numPt);

参数说明：

ControlName：工程人员定义的温控曲线控件名称，可以为中文名或英文名。

leftORrightSlide： 设置左游标或右游标。

1 左游标

0 右游标

direction：设置游标的移动方向

1 向左移动，越界则不动作。

0 向右移动，越界则不动作。

numPt：设置游标移动的采样点数

 例如：

pvMoveSlide("反应罐温控曲线", 0, 1, 3);

此语句执行后在反应罐温控曲线控件中将右游标左移动 3 个采样点，并在游标指示处显示注释信息。

pvSaveData

此函数用于将指定的温控曲线控件中的温控设定曲线存放到指定的文件中，存盘时，文件名自动添加.csv 后缀。

语法格式使用如下：

pvSaveData("ControlName", "FileName", "option");

参数说明：

ControlName：工程人员定义的温控曲线控件名称，可以为中文名或英文名。

FileName：FileName 文件以.csv 格式按曲线段数、各段升温速率、设定温度、保温时间依次存放温控曲线信息。

option： 确定是存放温控设定曲线或温控实时曲线，字符串常量。

“RealValue” 存放温控实时曲线

“SetValue” 存放温控设定曲线

 例如：

pvSaveData("反应罐温控曲线","fileName ","RealValue");

此语句把反应罐温控曲线控件中的温控实时曲线的采样历史数据值以.csv 格式存放到文件 fileName 中。

pvSaveData("反应罐温控曲线","fileName ","SetValue");

此语句把反应罐温控曲线控件中的温控设定曲线以.csv 格式存放到文件 fileName 中。

pvSetLimits

此函数用于改变指定的温控曲线控件的温度最大值、温度最小值、温度分度数、时

间最大值和时间分度数。

语法格式使用如下：

pvSetLimits("CtrlName",TempMax,TempMin,TempScale,TimeMax,TimeScale);

参数说明：

CtrllName：工程人员定义的温控曲线控件名称，可以为中文名或英文名。

TempMax：设置温控曲线的温度最大值，可以为正数或负数。

TempMin：设置温控曲线的温度最小值，可以为正数或负数。

TempScale：设置温控曲线的温度分度数，该变量应设置为整型变量。

TimeMax：设置温控曲线的时间最大值。

TimeScale：设置温控曲线的时间分度数，该变量应设置为整型变量。

 例如：

pvSetLimits(" 反 应 罐 温 控 曲 线 ",TempMax,TempMin,TempScale,TimeMax,TimeScale);

此语句将反应罐温控曲线控件中温度最大值设置成变量 TempMax 的值，将温度最小值设置成变量 TempMin 的值，将温度分度值设置成变量 TempScale 的值，将时间最大值设置成变量 TimeMax 的值，将时间分度值设置成变量 TimeScale 的值。

ReadTag

此函数用于在组态王运行时对属性为“读”、“读写”型的 IO 变量的采集频率进行修改。

语法格式使用如下：

ReadTag(tagName, freq);

参数说明：

TagName：字符串型　组态王数据词典中定义的 IO 变量名

freq：整型　采集频率设定值。范围 0－－3,000,000，单位：毫秒

当 Freq 设置为 0 时表示对变量进行单次采集，每执行一次函数，对变量采集一次；

Freq 设置为 1－－55 之间的某一值时，系统按照 55ms 的频率对变量进行数据采集；

Freq 设置为 56－－3,000,000 之间的某一值时，系统按照设置的频率对变量进行数据采集。

 例如：

ReadTag(“反应罐温度”，0)，此语句设定变量－反应罐温度的采集频率为 0，每执行一次，对反应罐温度采集一次数据。

ReadTag(“反应罐温度”，1000)，此语句设定变量－反应罐温度的采集频率为 1000，系统将按照 1000ms 的采集频率对反应罐温度进行数据采集。

ReBuildDDE

此函数用于重新建立 DDE 连接。

调用形式：

ReBuildDDE();

此函数无参数。

ReBuildUnConnectDDE

此函数用于重新建立未成功的 DDE 连接。

调用形式：

ReBuild UnConnectDDE();

此函数无参数。

RecipeDelete

此函数用于删除指定配方模板文件中当前指定的配方。

语法格式使用如下：

RecipeDelete("filename", "recipeName");

filename：指配方模板文件存放的路径和相应的文件名。

recipeName：指配方模板文件中特定配方的名字。

注意：

文件名和配方名如果加上双引号，则表示是字符串常量，若不加双引号，则可以是组态王中的字符串变量。

例如：

RecipeDelete（“C：\recipe\北京面包厂.csv”，“配方 3”）；

此语句将配方模板文件“北京面包厂.csv”中的配方 3 删除。

RecipeInsertRecipe

此函数用于在配方中选定的位置插入一个新的配方。执行该函数后，系统会弹出一个选择插入配方的对话框，对话框中列出了当前配方中所有的配方名称，选择要插

入的位置，确定后新的配方将被插入到指定配方的前面。调用格式如下：

RecipeInsertRecipe(filename, InsertRecipeName);

参数说明：filename 字符型 指配方模板文件存放的路径和相应的文件名

InsertRecipeName 字符型 要插入的新配方的名称

 例如：

RecipeInsertRecipe(“C:\recipe\北京面包厂.csv”,“新型配方”);

RecipeLoad

此函数将指定配方调入模板文件中的数据变量中。

语法格式使用如下：

RecipeLoad（“filename”,“recipeName”);

filename：指配方模板文件存放的路径和相应的文件名。

recipeName：指配方模板文件中特定配方的名字。

 注意：

文件名和配方名如果加上双引号，则表示是字符串常量，若不加双引号，则可以是组态王中的字符串变量。

 例如：

RecipeLoad（“C：\recipe\北京面包厂.csv”,“水果香型面包”);

此语句将配方模板文件“北京面包厂.csv” 中的配方“水果香型面包”调入

到项目模板定义中的数据变量中。

RecipeManage

此函数用来调用配方管理界面对配方进行管理。调用格式：

RecipeManage("recipe_group_name");

参数说明：

recipe_group_name：字符串类型，当前使用的配方组名称，如果没有就是第一个配方。

例如：

使用函数：RecipeManage（“新配方”）；可以调出下面的界面：

RecipeSave

此函数用于存放一个新建配方或把对原配方的修改变化存入已有的配方模板文件中。

语法格式使用如下：

RecipeSave（“filename”,“recipeName”）;

Filename：指配方模板文件存放的路径和相应的文件名。

recipeName：指配方模板文件中特定配方的名字。

注意：

1．文件名和配方名如果加上双引号，则表示是字符串常量，若不加双引号，则可以是组态王中的 I/O 型或内存型字符串变量。

2．配方模板文件必须存在，如果配方模板文件不存在，则要事先创建配方模板文件，否则，调用此函数将失败，并返回 FALSE。

例如：

RecipeSave（“C：\recipe\北京面包厂.csv”,“配方 3”）;

此语句将配方的修改变化存入到配方模板文件“北京面包厂.csv” 中的配方 3 中。如果“北京面包厂.csv” 中没有配方 3，则系统自动创建。

RecipeSelectNextRecipe

此函数用于在配方模板文件中选择指定配方的下一个配方。

语法格式使用如下：

RecipeSelectNextRecipe（“filename”, “recipeName”）;

filename：指配方模板文件存放的路径和相应的文件名。

recipeName：是一个字符串变量，存放工程人员选择的配方名字，

 注意：

文件名和配方名如果加上双引号，则表示是字符串参数，若不加双引号，则可以是组态王中的 I/O 型变量或内存型变量。

 例如：

RecipeSelectNextRecipe（“C：\recipe\北京面包厂.csv”，“配方 3”）；

此语句运行后读取模板文件中“配方 3”的下一个配方， 如果字符串变量 RecipeName 的值为空或没有找到,则返回文件中的第一个配方;如果变量 RecipeName 的值为文件中的最后一个配方，则仍返回此配方。

RecipeSelectPreviousRecipe

此函数用于在配方模板文件中选择当前配方的前一个配方。

语法格式使用如下：

RecipeSelectPreviousRecipe（“filename”，“recipeName”）；

filename：指配方模板文件存放的路径和相应的文件名。

recipeName：是一个字符串变量，存放工程人员选择的当前配方名字，

 注意：

文件名和配方名如果加上双引号，则表示是字符串参数，若不加双引号，则可以是组态王中的 I/O 型变量或内存型变量。

 例如：

RecipeSelectPreviousRecipe（“C:\recipe\北京面包厂.csv”，“配方 3”）；

此语句运行后读取模板文件中“配方 3”的的上一个配方，如果变量 RecipeName 的值为空或没有找到，则返回文件中的最后一个配方；如果变量 RecipeName 的值为文件中的第一个配方，则仍返回此配方。

RecipeSelectRecipe

此函数用于在指定的配方模板文件中选取工程人员输入的配方，运行此函数后，弹出对话框，工程人员可以输入指定的配方，并把此配方名送入字符串变量中存放。

语法格式使用如下：

RecipeSelectRecipe（“filename”，“recipeNameTag”，“Mess”）；

filename：指配方模板文件存放的路径和相应的文件名。

recipeNameTag：是一个字符串变量，存放工程人员选择的配方名字。

Mess：字符串提示信息，由工程人员自己设定。

 例如：

RecipeSelectRecipe（“C:\recipe\北京面包厂.csv”，RecipeName，“请输入配方名！”）；

此语句运行后将弹出一个“选择配方”对话框，给出提示信息“请输入配方名！”，一旦工程人员从对话框中选择了一个配方，则此函数将该配方的名字返回到变量 RecipeName 中存放。

Report1

此函数将按源文件中规定的数据报告格式生成相应的实时数据报告，此函数为 6.0

函数，建议 6.55 不使用该种报表制作方式。

使用格式：

Report1("Source", "OutputFile");

参数说明

Source：字符串常量，规定了数据报告格式的 RTF 文件，可用 WINDOWS 的写字板进行编写。

OutputFile：字符串常量，指定数据报告输出的路径和存放的文件。

 例如：

Report1(“c:\program files\kingview\example\Kingdemo3\report1.rtf”,“c:\program files\kingdemo3\实时数据.rtf”);

调用此函数后系统将在指定目录下以文件”实时报表.RTF”中规定的格式生成相应的名称为”实时数据.RTF” 的实时数据报告。

Report2

此函数将按源文件中规定的数据报告格式把指定时间内的数据生成相应的历史数据报告，此函数为 6.0 函数，建议 6.55 不使用该种报表。使用格式：

Report2(ST, "Source", "OutputFile");

参数说明

ST：存放工程人员设定的起始时间（StartTime），此值是通过组态王提供的命令语言函数 HTConvertTime()设定。

Source：字符串常量，规定了数据报告格式的 RTF 文件，可用 WINDOWS 的写字

板进行编写。

OutputFile：字符串常量，指定数据报告输出的路径和存放的文件。

例如：

ST= HTConvertTime(Year,Month,Day,Hour,Minute,Second);

Report2(ST,“c:\program files\kingview\example\Kingdemo1\report2.rtf”,“c:\program files\kingdemo1\历史数据.rtf”);

调用此函数后系统将在指定目录下以文件”历史报表.RTF”中规定的格式生成相应的名称为”历史数据.RTF” 的历史数据报告。

ReportPrint

此函数用于将指定的数据报告文件输出到“系统配置\打印配置”中规定的打印机上，点击工程浏览器中的“系统配置\打印配置”可以出现如下的对话框,”报告打印“规定了报告输出的打印机。

使用格式：

ReportPrint("OutputFile");

参数说明

OutputFile：指定要打印的数据报告文件。

 例如：

ReportPrint(“实时数据.rtf”);

调用此函数后将打印实时数据文件“实时数据.rtf”。

ReportPrint2

此函数为报表专用函数。将指定的报表输出到打印配置中指定的打印机上打印，语法使用格式如下：

ReportPrint2(String szRptName)　或者 ReportPrint2(String szRptName, EV_LONG|EV_ANALOG|EV_DISC) ;

参数说明：szRptName：要打印的报表名称

EV_LONG|EV_ANALOG|EV_DISC：整型或实型或离散型的一个参数，当该参数不为 0 时，自动打印，不弹出“打印属性”对话框。如果该参数为 0，则弹出“打印属性”对话框。

 例如：

自动打印“实时数据报表”：

ReportPrint2(“实时数据报表”)； 或 ReportPrint2(“实时数据报表”，1)；

手动打印时，弹出“打印属性”对话框：

ReportPrint2(“实时数据报表”，0);

ReportPrintSetup

此函数对指定的报表进行打印预览并且可输出到打印配置中指定的

打印机上进行打印。语法格式使用如下：

ReportPrintSetup(szRptName);

参数说明：szRptName：要打印预览的报表名称

 例如：

打印预览“实时数据报表”：

ReportPrintSetup(“实时数据报表”);

ReportGetCellString

此函数为报表专用函数。获取指定报表的指定单元格的文本，语法格式使用如下：

ReportGetCellString(ReportName, Row, Col);

返回值为字符串型

参数说明：

ReportName：报表名称

Row：要获取文本的报表的行号（可用变量代替）

Col：要获取文本的报表的列号（这里的列号使用数值，可用变量代替）

 例如：

获取报表“实时数据报表”中的第 2 行第 5 列的文本，赋给字符串型变量“文

本”：文本= ReportGetCellString(“实时数据报表”, 2, 5);

ReportGetCellValue

此函数为报表专用函数。获取指定报表的指定单元格的数值，语法格式使用如下：

ReportGetCellValue（ReportName, Row, Col）;

返回值为实型量

参数说明：

ReportName：报表名称

Row：要获取数据的报表的行号（可用变量代替）

Col：要获取数据的报表的列号（这里的列号使用数值，

可用变量代替）

 例如：

获取报表“实时数据报表”中的第 2 行第 4 列的数值，赋给实型变量“数值”：

数值= ReportGetCellValue(“实时数据报表”, 2, 4);

ReportGetColumns

此函数为报表专用函数。获取指定报表的行数，语法格式使用如下：

ReportGetColumns(ReportName);

参数说明：ReportName：报表名称

 例如：

获取报表“实时数据报表”的列数，赋给变量“列数”：

列数= ReportGetColumns(“实时数据报表”);

ReportGetRows

此函数为报表专用函数。获取指定报表的行数，语法格式使用如下：

ReportGetRows(ReportName);

参数说明：ReportName：报表名称

例如：

获取报表“实时数据报表”的行数，赋给变量“行数”：

行数= ReportGetRows(“实时数据报表”);

ReportSetRows

此函数为报表专用函数。设置指定报表的行数，语法格式使用如下：

ReportSetRows(String szRptName, long RowNum);

参数说明：

szRptName：报表名称

RowNum：要设置的行数

例如：

将 “实时数据报表”报表的行数设置为 1000 行

ReportSetRows(“实时数据报表”, 1000);

ReportSetColumns

此函数为报表专用函数。设置指定报表的列数，语法格式使用如下：

ReportSetColumns(String szRptName, long ColumnNum);

参数说明： szRptName：报表名称

ColumnNum：要设置的列数

例如：

将 “实时数据报表”报表的列数设置为 1000 列

ReportSetColumns (“实时数据报表”, 1000)

ReportLoad

此函数为报表专用函数。将指定路径下的报表读到当前报表中来，语法格式使用如下：

ReportLoad(ReportName, FileName)

返回值：返回存储是否成功标志 0 - 成功

-3 - 失败（注意定义返回值变量的范围）

参数说明：ReportName：报表名称

FileName：报表存储路径和文件名称

例如：

将文件名为“数据报表 1”，路径为“C:\My Documents”的报表读取到当前报表中,返回值赋给变量“读文件”：读文件= ReportLoad(“实时数据报表”,”C:\My Documents\报表.RTL”)；

ReportPageSetup

此函数为在运行状态下对报表进行页面设置函数，语法格式使用如下：

ReprotPageSetup(String szRptName);

参数说明：szRptName：要进行页面设置报表的名称

例如：

设置“实时数据报表”页面属性：

ReportPageSetup(“实时数据库表”);

ReportSaveAs

此函数为报表专用函数。将指定报表按照所给的文件名存储到指定目录下，ReportSaveAs 支持将报表文件保存为 rtl、xls、csv 格式。保存的格式取决于所保存的文件的后缀名。语法格式使用如下：

ReportSaveAs(ReportName,FileName);

返回值：整型　返回存储是否成功标志　0 - 成功

参数说明:

ReportName：报表名称

FileName：存储路径和文件名称

例如 1：

将报表“实时数据报表”存储为文件名为“数据报表 1.RTL”，路径为“C:\My Documents”，返回值赋给变量“存文件”：

存文件=ReportSaveAs(“实时数据报表”, “C:\My Documents\数据报表 1.RTL”);

例如 2：

将报表“实时数据报表”存储为 EXCEL 格式的文件，文件名为“数据报表 1.xls”，路径为“C:\My Documents”，返回值赋给变量“存文件”：

存文件=ReportSaveAs(“实时数据报表”, “C:\My Documents\数据报表 1.xls”);

ReportSetCellString

此函数为报表专用函数。将指定报表的指定单元格设置为给定字符串。语法格式使用如下：

ReportSetCellString(ReportName, Row, Col, Value)

返回值为整型量　0---成功；

-1---行列数小于等于零；

-2---报表名称错误；

-3---设置文本失败

参数说明：

ReportName：字符串型　报表名称

Row：　整型　要设置数值的报表的行号（可用变量代替）

Col：整型　要设置数值的报表的列号（这里的列号使用数值，　可用变量代替）

Value：字符串型　要设置的文本

例如：

根据组态王实型变量“压力” 的数据变化设置报表“实时数据报表”的第2行第 5 列为字符串变量“压力说明”的值，并且返回设置是否成功结果“字符串设置结果”（组态王变量），在数据改变命令语言中输入：

字符串设置结果=ReportSetCellString(“实时数据报表”, 2, 5, 压力说明)；

ReportSetCellString2

此函数为报表专用函数。将指定报表的指定单元格区域设置为给定字符串。即指定多个单元格的字符串值。语法格式使用如下：

ReportSetCellString2(ReportName, StartRow, StartCol, EndRow, EndCol, Value)

返回值：整型 0---成功；

-1---行列数小于等于零；

-2---报表名称错误；

-3---设置文本失败

参数说明：

ReportName：报表名称

StartRow：要设置数值的报表的开始行号（可用变量代替）

StartCol：要设置数值的报表的开始列号（这里的列号使用数值，可用变量代替）

EndRow：要设置数值的报表的结束行号（可用变量代替）

EndCol：要设置数值的报表的结束列号（这里的列号使用数值，可用变量代替）

Value：要设置的文本

 例如：

根据组态王实型变量“压力” 的数据变化设置报表“实时数据报表”的第8行第5列到第10行第7列为字符串变量“压力说明”的值，并且返回设置是否成功结果“字符串设置结果2”（组态王变量），在数据改变命令语言中输入：

字符串设置结果 2=ReportSetCellString2(“实时数据报表”, 8, 5, , 10, 7, 压力说明)；

ReportSetCellValue

此函数为报表专用函数。将指定报表的指定单元格设置为给定值。语法格式使用如下：

ReportSetCellValue(ReportName, Row, Col,Value)

返回值为整型量

0---成功；

-1---行列数小于等于零；

-2---报表名称错误；

-3---设置值失败

参数说明：

ReportName：报表名称

Row：要设置数值的报表的行号（可用变量代替）

Col：要设置数值的报表的列号（这里的列号使用数值， 可用变量代替）

Value：要设置的数值

 例如：

根据组态王实型变量“压力” 的数据变化设置报表“实时数据报表”的第2行第 4 列为变量“压力”的值，并且返回设置是否成功结果“实数设置结果”（组态王变量），在数据改变命令语言中输入：

实数设置结果=ReportSetCellValue(“实时数据报表”, 2, 4, 压力);

ReportSetCellValue2

此函数为报表专用函数。将指定报表的指定单元格区域设置为给定值，即指定多个单元格的值。语法格式使用如下：

ReportSetCellValue2(ReportName, StartRow, StartCol, EndRow, EndCol, Value)

返回值为整型量 0---成功；

-1---行列数小于等于零；

-2---报表名称错误；

参数说明：

ReportName：报表名称

StratRow：要设置数值的报表的开始行号（可用变量 代替）

StartCol：要设置数值的报表的开始列号（这里的列号使用数值，可用变量代替）

EndRow：要设置数值的报表的结束行号（可用变量代替）

EndCol：要设置数值的报表的结束列号（这里的列 号使用数值，可用变量代替）

Value：要设置的数值

 例如：

根据组态王实型变量“压力” 的数据变化设置报表“实时数据报表”的第3行第 4 列到第 6 行第 7 列区域为变量“压力”的值，并且返回设置是否成功结果“实数设置结果 2”（组态王变量），在数据改变命令语言中输入：

实数设置结果 2=ReportSetCellValue2(“实时数据报表”, 3, 4, 6, 7, 压力);

ReportSetHistData

此函数为报表专用函数，按照用户给定的参数查询历史数据，语法格式使用如下：

ReportSetHistData(ReportName, TagName, StartTime, SepTime, szContent) ;

参数说明：

ReportName：要填写查询数据结果的报表名称

TagName：所要查询的变量名称，类型为字符串型。

StartTime：数据查询的开始时间，该时间是通过组态王

HTConvertTime 函数转换的以 1970 年 1 月 1 日 8：00：00 为基准的长整型数，所以用户在使用本函数查询历史数据之前，应先将查询起始时间转换为长整型数值。

SepTime：查询的数据的时间间隔，单位为秒。

szContent：查询结果填充的单元格范围。

 例如：

查询变量“压力”自 2001 年 5 月 1 日 8：00：00 以来的数据查询间隔为 30 秒，数据报表的填充范围为’a2 :a50’，表示竖排第一列从第二行到第五十行。

long StartTime; （StartTime 为自定义变量）

StartTime=HTConvertTime(2001, 5, 1, 8, 0, 0);

ReportSetHistData(“历史数据报表”, “压力”, StartTime, 30, “a2:a50”);

ReportSetHistData2

此函数为报表专用函数。查询历史数据，使用该函数，只要设置查询的数据在报表中填充的起始位置，即输入起始行数（StartRow）、列数（StartCol）。系统会自动弹出历史数据查询对话框，语法使用格式如下：

ReportSetHistData2(StartRow,StartCol);

参数说明：

StartRow：查询的数据在报表中填充的起始行。

StartCol：查询的数据在报表中填充的起始列。

具体使用请参见《组态王 6.55 使用手册》中“报表”一章。

ReportSetHistData3

此函数可以控制变量在所关联的设备通讯失败，质量戳为坏，运行系统退出时在报表中的显示方式。

格式如下：

ReportSetHistData3("ReportName", "TagName", StartTime, SepTime, "szContent",bShowInvalidData);

参数说明：

ReportName：要填写查询数据结果的报表名称

TagName：所要查询的变量名称

StartTime：数据查询的开始时间，该时间是通过组态王 HTConvertTime 函数转换的以 1970 年 1 月 1 日 8：00：00 为基准的长整型数，所以用户在使用本函数查询历史数据之前，应先将查询起始时间转换为长整型数值。

SepTime：查询的数据的时间间隔，单位为秒

szContent：查询结果填充的单元格区域

bShowInvalidData：0，在报表中不显示变量通讯失败，质量戳为坏或者关机时段数据，1，在报表中显示最后记录的数据。

ReportSetHistDataEx

该函数可以按照用户给定的参数从组态王历史库或工业库查询历史数据，兼容 ReportSetHistData、ReportSetHistData3 的功能。调用格式：

ReportSetHistDataEx("ReportName",QueryFrom,"ServerName", "TagName", StartTime, SepTime, "szContent",bShowInvalidData);

参数说明：

ReportName：要填写查询数据结果的报表名称

QueryFrom：当 QueryFrom==1 时，从历史库中查询；当 QueryFrom==2 时，从工业库中查询。

ServerName：可以为工业库的站点名，也可以是历史库查询时的远程站点名或“本站点”。从历史库查询时,如果 TagName 是全名的形式，即:站点名\变量名，此项可省略。

TagName：为工业库或历史库中的变量名

StartTime：数据查询的开始时间，该时间是通过组态王 HTConvertTime 函数转换的以 1970 年 1 月 1 日 8：00：00 为基准的长整型数，所以用户在使用本函数查询历史数据之前，应先将查询起始时间转换为长整型数值

SepTime：查询的数据的时间间隔，单位为秒

szContent：查询结果填充的单元格区域

bShowInvalidData：0：在报表中不显示变量通讯失败或者关机时段数据，1：在报表中显示最后记录的数据。

 例如：在报表中插入的时间为自 2007 年 7 月 3 日 13：27：00 以来的时间段，时间段间隔为 2 秒，数据报表填充的区域为”a2:a50”数据来自本站点历史库；数据报表填充的区域为”c2:c50”数据来自本机工业库：

```
long StartTime;
StartTime=HTConvertTime(2007, 7, 3, 13, 27, 0);
ReportSetHistDataEx("Report1",1,"本站点", "nIODec50", StartTime, 2, "a2:a50",0);
ReportSetHistDataEx("Report1",1," ", "\\本站点\nIODec40", StartTime, 2, "b2:b50",0);
ReportSetHistDataEx("Report1", 2 ,"127.0.0.1", "ZFYLYP_nIODec50.Value", StartTime, 20, "c2:c50",0);
```

ReportSetLock

此函数为报表专用函数，用于锁定报表的前 nRow 行和 nCol 列，语法格式使用如下：

BOOL ReportSetLock(const CHAR* ReportName,BOOL is_enable,int nRow,int nCol)。

参数说明：

const CHAR* ReportName：报表名称。

BOOL is_enable：是否锁定行列，为 TRUE 时执行锁定操作；为 Fales 时，执行取消锁定操作。

int nRow：锁定的行数，int nCol：锁定的列数。

例如：ReportSetLock（“值班表”，1,1,2)：对值班表执行锁定操作，锁定第1行和前2列。

注意：

1.锁定操作只能在报表没有锁定时执行，如果报表已锁定，必须先取消锁定，才能执行锁定。

2.锁定操作必须在报表滚动条没有滚动过的情况下执行，否则无效。

ReportSetRowColResize

此函数为报表专用函数，按照用户给定的参数设置报表的行和列是否可以调整大小，语法格式使用如下：

```
Bool ReportSetRowColResize( "ReportName", bRow,bCol );
```

参数说明：

"ReportName"：报表名称

bool bRow：行设定，0为不能调整，1为可以调整

bool bCol：列设定，0为不能调整，1为可以调整

成功返回1，失败返回0.

ReportSetStartTime

此函数为报表专用函数，用于在使用了报表向导功能时，在运行系统中重新设置报表查询的起始时间，可以在按钮弹起的脚本中使用。运行后将弹出设置报表起始时间的对话框。语法格式使用如下：

```
void ReportSetStartTime (const char *pReprotName)
```

参数说明：

const char* pReportName: 报表名称。

ReportSetTime

此函数为报表专用函数，向报表设置连续的时间字符串，配合函数 ReportSetHisData 设置返回的历史数据的时间，日期和时间显示格式可以在开发系统报表单元格中进行设置。语法格式使用如下：

ReportSetTime("ReportName", StartTime, SepTime, "szContent");

参数说明：

ReportName：要填写查询数据结果的报表名称。

StartTime：数据查询的开始时间，该时间是通过组态王 HTConvertTime 函数转换的以 1970 年 1 月 1 日 8：00：00 为基准的长整型数，所以用户在使用本函数查询历史数据之前，应先将查询起始时间转换为长整型数值。

SepTime：生成时间的时间间隔，单位为秒。

szContent：生成的时间字符填充的单元格区域。

 例如：

在报表中插入的时间为自 2001 年 5 月 1 日 8：00：00 以来的时间段，时间段间隔为 30 秒，数据报表填充的区域为”a2:a100”：

long StartTime; （StartTime 为自定义变量）

StartTime=HTConvertTime(2001, 5, 1, 8, 0, 0);

ReportSetTime (“历史数据报表”, StartTime, 30, “a2:a100”);

ReportSetTimeEx

此函数为报表专用函数，向报表设置连续的日期和时间字符串，配合函数 ReportSetHisData 设置返回的历史数据的时间；日期和时间的显示格式也可以在开发系统报表单元格中设置。语法格式使用如下：

ReportSetTimeEx("ReportName", StartTime, SepTime, "szContent", nShowType);

参数说明：

ReportName：要填写查询数据结果的报表名称。

StartTime：数据查询的开始时间，该时间是通过组态王 HTConvertTime 函数转换的以 1970 年 1 月 1 日 8：00：00 为基准的长整型数，所以用户在使用本函数查询历史数据之前，应先将查询起始时间转换为长整型数值。

SepTime：生成时间的时间间隔，单位为秒。

szContent：生成的时间字符填充的单元格区域。

nShowType：显示格式设置

0：同时显示日期和时间

1：仅显示日期(年、月、日)

2：仅显示时间(时、分、秒)

ReportWebDownload

此函数是为使用组态王 WEB 版的用户准备的。可以实现两个功能：

1. 从组态王 WEB 服务器上下载指定的报表内容到 IE 浏览器上的对应报表中。
2. 将 IE 浏览器上显示的报表内容本地下载到指定的文本文件中。

由于组态王 WEB 版在 IE 浏览器上不支持后台命令语言，所以该函数的执行必须通过按钮动作命令语言来实现。

语法使用格式如下：

ReportWebDownload(ReportName, DownloadType);

参数描述

ReportName： 要下载内容的报表名称，字符串型

DownloadType： 下载方式，整型。下载方式共有三种：

1. DownloadType==0 时：在浏览器端执行该函数，将 IE 浏览器上显示的报表内容下载到一个“.csv”格式的指定文件中。

2. DownloadType==1 时：在浏览器端执行该函数，把 WEB 服务器上组态王运行系统中指定的报表内容下载到 IE 浏览器上对应的报表中。

3. DownloadType==2 时：直接把 WEB 服务器上组态王运行系统中指定的报表内容下载到 IE 浏览器端。然后将给报表的内容本地下载到指定的“.csv”格式的文件中。

 例如：

在 IE 客户端上浏览报表 WEB 页面时，刷新报表中数据为 WEB 服务器上组态王画面中对应报表已更新的数据：

ReportWebDownload(“实时数据报表”, 1);

ResetAllFieldForDataChange

该函数使所有变量的 datachanged 域复位。无返回值。无参数。调用格式：

ResetAllFieldForDataChange();

 例如：

使用函数：ResetAllFieldForDataChange ();

复位所有变量的 datachanged 域值。

SampleVar

该函数是为要进行间歇采集的 IO 变量提供的解决方法。在使用前，需要将进行间歇采集的 IO 变量的采集频率定义为 0 毫秒（否则该函数没有用处），在需要进行采集时，执行该函数，将变量准备写入组态王数据采集队列，然后执行 SampleVarEnd()函数，进行一次数据采集。调用格式：

SampleVar(TagName);

参数：TagName　字符串型　要进行间歇采集的变量名称

 例如：

环境监测中的“水质含氧量”为在需要时才进行采集的变量，其它时间不需要采集。则在组态王变量词典中将该变量的采集频率定义为 0，在命令语言脚步中使用该函数：

SampleVar(“水质含氧量”);

当该函数执行时，将“水质含氧量”变量的采集信息准备写入组态王的数据采集队列。当需要正式采集时执行 SampleVarEnd()函数，进行一次数据采集。

 注意：

1. 这两个函数要在同一命令语言脚步中同时使用。可以执行多个 SampleVar()函数后，执行一次 SampleVarEnd()函数，才能进行采集。
2. 要使用该函数进行间歇采集的 IO 变量的采集频率一定要设置为 0 毫秒。

SampleVarEnd

执行 SampleVar()函数后，执行该函数，将需要进行采集的变量信息写入组态王数据采集队列。调用格式：

SampleVarEnd();

没有参数。

SavePicToFile

该函数的作用是把画面保存成为 BMP 或者 JPG 文件。调用格式：

SavePicToFile("PicName",StartX,StartY,Height,Width,"FileName");

参数说明：

PicName：需要保存的画面的名称。

StartX：保存画面 X 坐标开始位置。

StartY：保存画面 Y 坐标开始位置。

Height：保存画面的高度。

Width：保存画面的宽度。

FileName：包含保存路径的保存文件名，结尾必须是文件扩展名.bmp 或者 jpg。

例如：

工程有画面“年报表”，高 200，宽 200。要把整个画面都保存在 C 盘根目录下则使用函数：

SavePicToFile("年报表",0,0,200,200,"C:/年报表.bmp");

SaveText

此函数用于把超级文本显示控件中显示和编辑输入的文本字符串保存到指定的 RTF

或 TXT 格式文件中。

语法格式使用如下：

SaveText("ControlName", "FileName", ".Txt Or .Rtf");

参数说明：

ControlName：工程人员定义的超级文本显示控件名称，可以为中文名或英文名。

FileName：RTF 或 TXT 格式的文件，可用 WINDOWS 的写字板编写这两种格式的文件。

.Txt Or .Rtf：指定文件为 RTF 格式或 TXT 格式。

 例如：

SaveText("hypertext1","D:\Test\recipe\ht1.rtf",

"".Rtf");

此语句把超级文本显示控件 hypertext1 中显示和编辑输入的文本字符串保存到文件 ht1.rtf 中。

ScrollPicture

该函数是画面存在滚动条时，将画面滚动到目标区域。调用格式：

ScrollPicture("PictureName", xPoint , yPoint);

无返回值。

参数说明：

PictureName：需要滚动的画面名称。

xPoint：目标区域矩形的左上角的 X 坐标。

yPoint：目标区域矩形的左上角的 Y 坐标。

例如：

要是画面“报表”滚动到（20,20）的位置，使用函数：

ScrollPicture("报表", 20 , 20);

SendKeys

此函数与StartApp、ActivateApp配合使用，使“组态王”具备了远程控制其它应用程序的能力，这是“组态王”的重要功能之一。它可以启动另一应用程序，如Excel，然后又可以命令该应用程序执行一组功能，如产生报表，趋向图或记录数据。所需的过程可以用某一应用程序（比如Excel）的“宏”语言来写。这就是说，只要按一次键调用该宏命令就能启动很复杂的过程。这种用其他功能很强的应用程序作为从属程序的能力大大增强了“组态王”的功能。

该函数用于将击键信息发送至当前获得输入焦点的应用程序。对于此应用程序来说，键似乎已由键盘输入。在调用此函数时，必须使接受键信息的应用程序获得输入焦点。因此需要先调用ActivateApp。

例如：

ActivateApp("Excel.exe");

SendKeys("^(X)");

将 Control X 键信息发送至 Excel。对于 Excel 来说，这可能为报告生成宏命令的命令码。

其调用形式为：

SendKeys(keyT);

参数 keyT 为特定键的代码，代码意义和用法与 Microsoft 的 Excel 的函数 Send

Keys 中参数 keyT 相同，可参照下面的表：

键码	意义
{BACKSPACE}or{BS}	Backspace
{BREAK}	BreakCaps
{CAPSLOCK}	Caps Lock
{CLEAR}	Clear
{DELETE}or{DEL}	Delete or Del
{DOWN}	Down direction key
{END}	End
{ENTER}or~	Enter
{ESCAPE}or{ESC}	Esc(Escape)
{HOME}	Home
{INSERT}	Insert
{LEFT}	Left direction key
{NUMLOCK}	Num Lock
{PGDN}	Page Down
{PGUP}	Page up
{PRTSC}	Print Screen
{RIGHT}	Right direction key
{SCROLLLOCK}	Scroll Lock
{TAB}	Tab
{UP}	Up direction key

{F1}through{F12}　　Function keys F1 through F12

可以用大写或小写的字符定义一个键命令，还可以同时与下面的键配合使用：

键码	意义
+	Shift
^	Ctrl
%	Alt


例如：为了发送一键序列来拷贝已选定的区，调用函数

SendKeys("^{insert}");

为了表示在另一键按下时按下 SHIFT，CONTROL 或 ALT，可以将其它键放入括号内。如：

SendKeys("%(TFR)~");

这表示先发出击键信号：Alt-t、Alt-f 和 Alt-r,然后是 Enter 回车键。%指代 Alt 键，因为跟在 Alt 键码后面的字母都在括号中，所以当每一键按下时 Alt 键好象同时也被按下。

SendKeys("secret~");表示先发出字符串 secret，然后按回车键。

由于字符+、^和%都有特殊含义，为了输入这些字符本身而不取其特殊含义，应给字符加花括号，如：SendKeys("A{+}B")，表示发出字符串 A+B。

SetAlarmWinDis

此函数用力对已经发生报警还未恢复的变量，分别显示确认和未确认的变量；即增加报警消失条件的设定函数（默认是恢复即消失）。语法使用格式如下：

void SetAlarmWinDis(string name, long nSet);

参数说明：

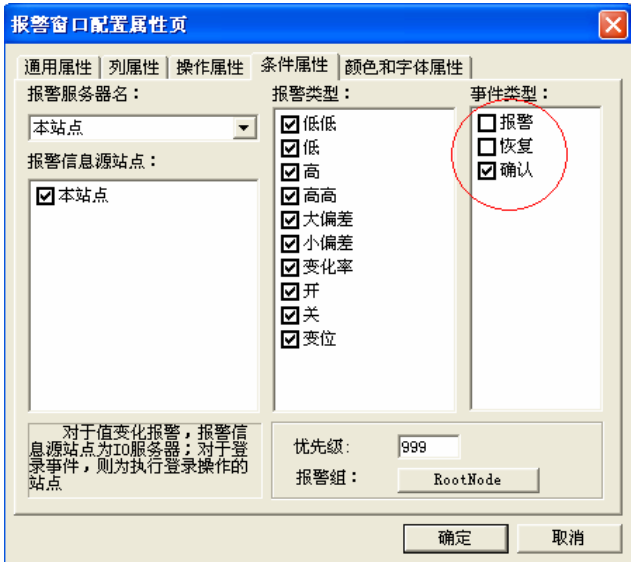
string name：报警窗的名称

nSet=1：恢复+确认，nSet=2：确认，其它为恢复

注意：

此函数仅对于实时报警窗有效， 并且仅在全新版的 Web 发布中可用，在旧 web 发布上不支持。

当设置为确认报警消失时，可以实现显示未确认的报警；要想显示已确认的报警，在“报警窗口属性配置页”中“事件类型”只勾选“确认”，如下图所示。



SetIoDeviceRunState

该函数是设置设备当前状态。调用格式：

SetIoDeviceRunState("LoigicDeviceName", RunState);

参数说明：

LoigicDeviceName：工程中定义的设配名称。

RunState：设置的设备状态。0 表示运行，非零表示暂停使用。


例如：

工程使用设备“莫迪康 PLC”，需要暂时停止这个设备的采集，使用函数：

SetIoDeviceRunState("莫迪康 PLC ", 1);

SetNetNodeValid

此函数用于手动屏蔽网络节点;对于被屏蔽的站点，断开连接，不进 行尝试，不进行数据的交换。重新激活后，重新连接。

语法格式使用如下：

SetNetNodeValid(szNodeName, bFlag);

参数：

szNodeName：节点名

bFlag： 0,屏蔽 1 激活

返回 0 表示成功


例如：

SetNetNodeValid("king", 0);

SetPrintAlarm

该函数用来执行实时打印功能。

语法格式：

BOOL SetPrintAlarm(BOOL bitset);

参数描述：

bitset：bool 类型，为 1 时则停止报警打印，为 0 开始报警打印

返回值：bool 类型，能反映设置后的打印状态，为 1 时则停止报警打印，为 0 开始报警打印

注意：

此函数只有当配置了报警配置中的报警打印后才能起作用。

SetRealDBForBool

此函数用于生成离散型变量的操作事件。除按钮命令语言外，在组态王命令语言中修改变量的值可以生成操作事件。

调用形式：

SetRealDBForBool("VarName",Value);

参数：

VarName：变量名称，离散型格式

Value：为变量值

例如：

在画面 “显示时”命令语言中输入:

SetRealDBForBool ("\\本站点\阀门",1)

设置实型变量“\\本站点\阀门”的值为 1。

画面显示时，生成实型变量\\本站点\阀门的操作事件。

SetRealDBForFloat

此函数用于生成实型变量的操作事件。除按钮命令语言外，在组态王命令语言中修改变量的值可以生成操作事件。

调用形式:

SetRealDBForFloat("VarName",Value):

VarName 为变量名，Value 为变量值

例如:

在画面 “显示时”命令语言中输入:

SetRealDBForFloat ("\\本站点\液位",1.5)

设置实型变量“\\本站点\液位”的值为 1.5。

画面显示时，生成实型变量\\本站点\液位的操作事件。

SetRealDBForInt

此函数用于生成整型变量的操作事件。除按钮命令语言外，在组态王命令语言中修改变量的值可以生成操作事件。

调用形式:

SetRealDBForInt("VarName",Value);

VarName 为变量名，Value 为变量值

 例如：

在画面 “显示时”命令语言中输入:

SetRealDBForInt ("\\本站点\行",10);

设置整型变量“\\本站点\行”的值为 10。

画面显示时，生成整型变量“\\本站点\行”的操作事件。

SetRealDBForString

此函数用于设置变量的当前实时字符串型值。

调用形式：

SetRealDBForString ("VarName"," Value");

参数：

VarName：变量名称

Value：要设置的字符串

说明：变量只能是字符串型变量

 例如：

新建按钮，弹起事件命令语言连接弹起时：

SetRealDBForString ("\\本站点\v ","abc");

设置字符串变量“\\本站点\v”的值为 abc。

SetTrendPara

此函数用于建立一个按钮，供“组态王”运行中弹出对话框以改变历史趋势曲线的参数，如起始时间、数据长度、纵轴的起点、纵轴的终点等。

调用形式：

SetTrendPara(Trend_Tag);

参数：

Trend_Tag:历史曲线名称。

 例如：

SetTrendPara(历史趋势曲线); //"历史趋势曲线"为历史趋势曲线名称。

Sgn

此函数判别一个数值的符号(正、零或负)。调用格式：

IntegerResult=Sgn(Number);

参数描述

Number 任一数值或组态王实型或整型变量名。

若数值为正，则返回值为 1。数值为负的则返回值为 -1，数值为 0 则返回 0。

 例如：

Sgn(425);//将返回 1

Sgn(0);//将返回 0

Sgn(-37.3);//将返回 -1

ShowNavigateWindow

此函数用于实现导航窗口的显示与隐藏。调用格式：

ShowNavigateWindow(nCmdShow);

参数描述

nCmdShow：导航窗口的显示与隐藏控制。nCmdShow=0，隐藏导航窗口；nCmdShow=1，显示导航窗口。

 例如：显示导航窗口。

ShowNavigateWindow(1);

ShowPicture

此函数用于显示画面。调用格式：

ShowPicture("PictureName");

 例如：

ShowPicture("反应车间");

Sin

此函数用于计算变量值的正弦值，调用格式：

Sin(变量值);

 例如：

Sin(90); 此函数返回值为 1

Sin(0); 此函数返回值为 0

SQLAppendStatement

在SQLSetStatement()后，附加一条语句。

语法：

[ResultCode=]SQLAppendStatement(DeviceID,"SqlStatement");

参数描述

DeviceID SQLConnct()产生的连接号

SqlStatement 附加的SQL语句

例如：附加条件：列名年龄=24

SQLAppendStatement(DeviceID, “where 年龄=24”);

注意：

本函数自动删除语句前的空格，所以同一个变量或连接符不能分开到两条语句中

SQLClearStatement

释放和SQLHandle相关联的资源。

语法：

[ResultCode=]SQLClearStatement(DeviceID,SQLHandle);

参数描述

DeviceID SQLConnct()产生的连接号

SQLHandle SQLPrepareStatement()产生的内存整数

SQLClearTable

删除表格中的所有记录，但保留表格。

语法：

[ResultCode=]SQLClearTable(DeviceID, “TableName”);

参数描述

DeviceID SQLConnct()产生的连接号

TableName 需访问的数据库表名

例如：删除表格kingview中的所有记录

[ResultCode=]SQLClearTable(DeviceID, “kingview”);

SQLCommit

定义了一组访问语句的结尾。在SQLTransact()指令和SQLCommit()指令之间的一组指令称为一个指令集。一个指令集的管理如同一个单一指令。SQLTransact()后的指令暂不执行，直到执行了SQLCommit()。语法：

SQLCommit(DeviceID);

参数描述

DeviceID SQLConnct()产生的连接号

注意：

由于同时执行多个指令，SQLCommit()执行速度会变慢。

例如：连续实现三次插入

```
SQLTransact(DeviceID);
SQLInsertPrepare(DeviceID,TableName,BindList,SQLHandle);
SQLInsertExecute(DeviceID,BindList,SQLHandle);
SQLInsertExecute(DeviceID,BindList,SQLHandle);
SQLInsertExecute(DeviceID,BindList,SQLHandle);
SQLCommit(DeviceID);
```

SQLConnect

连接组态王和数据库。

语法：

```
[ResultCode=]SQLConnect(DeviceID, "dsn=;uid=;pwd=");
```

参数描述

DeviceID SQLConnct()产生的连接号

"dsn=;uid=;pwd=" 连接语句

"dsn=;uid=;pwd="格式如下：

“DSN=data source name [;attribute= value[;attribute = value]…”

例如：

组态王以sa身份登录（无密码）和名为wang的SQL Server中的pubs数据库连接

```
[ResultCode=]SQLConnect(DeviceID, “DSN=wang;DATABASE=pubs;UID=sa;
```

PWD=”);

属性描述：

属性	值
DSN	ODBC 中定义的数据源名
UID	登录 ID 号
PWD	密码，区分大小写
DATABASE	所要访问的数据库名

SQLCreateTable

以表格模板中定义的表格类型，在数据库中创建新表。

语法:

[ResultCode=]SQLCreateTable(DeviceID,"TableName","TemplateName");

参数描述

DeviceID SQLConnct()产生的连接号

TableName 想要创建的数据库名

TemplateName 表格模板名

 例如：

创建一个名为kingview的新表，模板为table1。

SQLCreateTable(DeviceID, “kingview”, “table1”);

SQLDelete

删除一条或多条记录。

语法：

[ResultCode=]SQLDelete(DeviceID, "TableName", "WhereExpr");

注意：

SQLDelete()函数的条件表达不能为空。

参数描述

DeviceID SQLConnct()产生的连接号

TableName 表名

WhereExpr 指定函数起作用行的条件

注意：

如果列名是字符串，表达式必须在单引号中。

下例选择“名字”列中等于Asia的行：

名字= ’ Asia’

下例选择“年龄”列中在20和30之间的行：

年龄>=20 and 年龄<30

例如：删除kingview表格中所有LogNo列等于11的记录

SQLDelete(DeviceID, “kingview”, “LogNo=11”);

SQLDisconnect

从使用的数据库中断开连接。

语法：

[ResultCode=]SQLDisconnect(DeviceID);

参数	描述
DeviceID	SQLConnct()产生的连接号

SQLDropTable

删除一个表格（包括结构）。

语法：

[ResultCode=] SQLDropTable(DeviceID, TableName);

参数	描述
DeviceID	SQLConnct()产生的连接号
TableName	表格名称

SQLEndSelect

在使用SQLSelect()之后使用此函数释放用来存储结果表格的资源

语法：

[ResultCode=]SQLEndSelect(DeviceID);

参数	描述
DeviceID	SQLConnct()产生的连接号

SQLErrorMsg

返回和特定的ResultCode相关的错误字符串信息。

语法：

SQLErrorMsg(ResultCode,buf);

参数描述

ResultCode　大多数SQL函数都返回一个整数。如果为零，函数调用成功，如果为负，调用失败。

buf 显示部分错误信息提示

更多的信息，请参阅SQL函数疑难解答（组态王SQL Sever使用手册）

 例如：返回信息

ErrorMsg = SQLErrorMsg(ResultCode,buf);

其中 buf 对应组态王中的 I/O 字符型变量，buf 仅能显示部分错误信息提示，大部分错误信息提示在组态王信息窗口中会有相应显示。

SQLExecute

执行SQL语句。

语法：

[ResultCode=]SQLExecute(DeviceID, "BindList" , SQLHandle);

参数描述

DeviceID SQLConnct()产生的连接号

BindList 记录体，指定组态王变量和表格列之间的对应关系

SQLHandle　如果调用前执行了SQLPrepareStatement()，此参数为返回的一个整数，如果没有准备的句柄，此值为零。

注意：

如果没有准备好的句柄，此函数只能执行一次，如果经过SQLPrepareStatement()准备，可以重复执行。

SQLFirst

从SQLSelect()函数产生的结果集中选取首项记录。

语法：

[ResultCode=]SQLFirst (DeviceID);

参数描述

DeviceID：SQLConnct()产生的连接号

SQLGetRecord

返回当前选择集中的指定序号的记录。

语法：

[ResultCode=]SQLGetRecord(DeviceID, RecordNumber);

参数描述

DeviceID SQLConnct()产生的连接号

RecordNumber 序号


例如：返回选择集中的第三条记录

SQLGetRecord(DeviceID, 3);

SQLInsert

使用记录体中定义的连接在表格中插入一个新的记录。

语法：

[ResultCode=]SQLInsert(DeviceID, "TableName", "BindList");

参数描述

DeviceID SQLConnct()产生的连接号

TableName 表格名

BindList 记录体


例如：

在表格kingview中插入一条记录，记录体bind1

SQLInsert(DeviceID, “kingview”, “bind1”);


注意：

以下三个函数配合使用可以取代标准的 SQLInsert()实现快速插入：SQLInsetPrepare(),SQLInsertExecute(),SQLInsertEnd()。

SQLInsert()是一个一步完成程序，包括插入和释放资源。因此，当多次使用时，整个过程反复执行，资源也在被反复分配和释放。。

而SQLInsertPrepare()分配句柄SQLHandle后，可以使用该句柄连续执行多个

SQLInsertExecute(), 最后执行SQLInsertEnd()释放句柄。这样，同样的资源反复使用以达到提高效率的目的。

SQLInsertEnd

释放语句。

语法：

[ResultCode=]SQLInsertEnd(DeviceID, SQLHandle);

参数	描述
DeviceID	SQLConnct()产生的连接号
SQLHandle	SQLInsertPrepare()产生的句柄

SQLInsertExecute

插入语句执行。

语法：

[ResultCode=]SQLInsertExecute(DeviceID,"BindList", SQLHandle);

参数描述

DeviceID SQLConnct()产生的连接号

BindList 记录体

SQLHandle SQLInsertPrpare 产生的句柄

SQLInsertPrepare

产生和准备一个插入语句。插入语句不执行。执行后将产生一个句柄。

语法：

[ResultCode=]SQLInsertPrepare(DeviceID,"TableName","BindList",SQLHandle);

参数描述

DeviceID SQLConnct()产生的连接号

TableName 表名

BindList 记录体

SQLHandle 产生的句柄

SQLLast

选取由SQLSelect()创建的选择集的末条记录。此句执行之前，必须执行SQLSelect()。

语法：

[ResultCode=]SQLLast(DeviceID);

参数描述

DeviceID SQLConnct()产生的连接号

例如：SQLLast(DeviceID);

SQLLoadStatement

读取包含在文件中的语句。类似于SQLSetStatement()函数。此函数也可以用SQLAppendStatement()附加语句。每个文件只能包含一个指令。

语法：

[ResultCode=]SQLLoadStatement(DeviceID, "OutputFile");

参数描述

DeviceID SQLConnct()产生的连接号

OutputFile　文件名


例如：

SQLLoadStatement(DeviceID, “C:\InTouchAppname\SQL.txt”);

在文件 SQL.txt 中，包含以下信息：

Select ColumnName from TableName where ColumnName>100;

SQLNext

选取由SQLSelect()产生的选择集中的下一条记录。

语法：

[ResultCode=]SQLNext(DeviceID);

参数描述

DeviceID SQLConnct()产生的连接号


例如：SQLNext(DeviceID);

SQLNumRows

指出SQLSelect()函数指定的选择集中包含多少行。

SQLNumRows(DeviceID);

参数描述

DeviceID SQLConnct()产生的连接号

 例如：

NumRows=SQLNumRows(DeviceID);

SQLPrepareStatement

SQLSetStatement() 或SQLLoadStatement()和SQLAppendStatement()指定的语句。返回句柄。

语法：

[ResultCode=]SQLPrepareStatement(DeviceID, SQLHandle);

参数描述

DeviceID SQLConnct()产生的连接号

SQLHandle 产生的句柄

 例如：将返回的句柄赋给内存变量handle

SQLPrepareStatement(DeviceID, SQLHandle);

SQLPrev

选取选择集中的上一条记录。

语法：

SQLPrev(DeviceID);

参数描述

DeviceID SQLConnct()产生的连接号

SQLRollback

撤消最近的位于SQLTransact()后的尚未“提交”的指令。

语法：

[ResultCode=]SQLRollback(DeviceID);

参数描述

DeviceID SQLConnct()产生的连接号

例如：

```
SQLTransact(DeviceID);
SQLInsertPrepare(DeviceID, “kingview”, “bind1”, handle);
SQLInsertExecute(DeviceID, “bind1”, handle);
SQLInsertEnd(DeviceID, handle);
/*如果这时执行 SQLCommit(DeviceID), 以上语句将执行。*/
SQLRollback(DeviceID);
/*撤消 SQLTransact()后的语句。*/
```

SQLSelect

访问数据库，得到一个特定的选择集。选择集中的记录可以由SQLFirst(), SQLNext(), 等函数访问。

语法：

[ResultCode=]SQLSelect(DeviceID,"TableName","BindList", "WhereExpr","OrderByExpr");

参数描述

DeviceID SQLConnct()产生的连接号

TableName　表格名称

BindList 记录体

WhereExpr　指定函数起作用行的条件。注意：如果列名是字符串，表达式必须在单引号中。

下例选择“名字”列中等于Asia的行：

名字= ’ Asia’

下例选择“年龄”列中在20和30之间的行：

年龄>=20 and 年龄<30

OrderByExpr 定义排序的列和方向。只有列名可以用来排序，表达式：列名[ASC|DESC]。下例将以“温度”列的升序排序

“温度 ASC”

排序中也可使用多重表达式。例如；

“温度 ASC，时间 DESC”

WhereExpr 举例：

字符串例子：

“Ser_No=’ abcd’ ”

字符串中使用like语句：

“Ser_No like ab%”。注意：使用%代表广义字符。

字符串和模拟量中间使用and连接

“Ser_No=’ abcd’ and Number=150”

注意：当SQLSelect()产生的选择集完成后，总要使用SQLEndSelect () 释放资源。

例如：

选择表格kingview中列名“Ser_No”为abcd的行，以温度列的升序排序。变量对应关系在bind1中指定。

SQLSelect(DeviceID, “kingview”, “bind1”, “Ser_No=’abcd’, “温度 ACS”);

例如：

字符串变量：FindDate 表示条件；这是相似查询

string WhereExpr="日期 like+'%"+FindDate+"%'";

SQLSelect(DeviceID, "数据记录表", "日报表", WhereExpr, "");

例如：String str1="炉号="+"'"+\\本站点\test+"'";

SQLSelect(DeviceID, "表 2", "Bind2", str1, ""); //SQL变量条件查询, \\本站点\test：字符串变量

例如：String strtime=StrFromInt(inttime, 10);

//inttime:为整数类型。先转换为字符串类型

string str1="Times="+"'"+strtime+"'";

SQLSelect(DeviceID, "数据查询", "BIND", str1, "");

例如：选取表格 kingview 中的所有行。

SQLSelect(DeviceID, “kingview”, “bind1”, “”, “”);

SQLSelectTop()

该函数用于查询 SQL 数据库中符合查询条件的前 N 条记录，语法格式如下：

SQLSelectTop(DeviceID, "TableName", "BindList", "WhereExpr", "OrderByExpr", "TopExpr");

参数：

DeviceID：SQLConnct()产生的连接号

TableName：表名

BindList：记录体名

WhereExpr：条件子句

OrderByExpr：排序子句

TopExpr：符合查询条件和排序的前 N 条，如 TopExpr＝3，则表示符合查询条件和排序的前 3 条记录。

SQLSetParamChar

以指定的参数为指定的字符串赋值。

语法：

[ResultCode=]SQLSetParamChar(SQLHandle, ParameterNumber, "ParameterValue", MaxLen);

参数描述

SQLHandle 由 SQLPrepareStatement()函数准备的句柄

ParameterNumber 语句中参数出现的序号

ParameterValue 赋给的值（可以是组态王变量）

MaxLen 参数相关列的最大长度

SQLSetParamDate

为指定的字符型参数赋日期值。

语法：

[ResultCode=]SQLSetParamDate(SQLHandle,ParameterNumber, ParameterValue);

参数描述

SQLHandle 句柄

ParameterNumber 语句中参数出现的序号

ParameterValue 赋给的值（可以是组态王变量）

SQLSetParamDateTime

为指定的字符型参数赋日期和时间值。

语法：

[ResultCode=]SQLSetParamDateTime(SQLHandle, ParameterNumber, ParameterValue, Precision);

参数描述

SQLHandle 句柄

ParameterNumber 语句中参数出现的序号

ParameterValue 赋给的值（可以是组态王变量）

Precision 参数ParameterValue所用到的字符串的个数

SQLSetParamDecimal

为指定的整型参数赋十进制值。

语法：

[ResultCode=]SQLSetParamDecimal(SQLHandle, ParameterNumber, "ParameterValue", Precision, Scale);

参数描述

SQLHandle 句柄

ParameterNumber 语句中参数出现的序号

ParameterValue 赋给的值（可以是组态王变量）

Precision 参数ParameterValue所用到的字符串的个数

SQLSetParamTime

为指定的字符型参数赋时间值。

语法：

[ResultCode=]SQLSetParamTime(SQLHandle,ParameterNumber, ParameterValue);

参数描述

SQLHandle 句柄

ParameterNumber 语句中参数出现的序号

ParameterValue 赋给的值（可以是组态王变量）

SQLSetParamFloat

为指定的浮点型参数赋值。

语法：

[ResultCode=]SQLSetParamFloat(SQLHandle,ParameterNumber, ParameterValue);

参数描述

SQLHandle 句柄

ParameterNumber 语句中参数出现的序号

ParameterValue 赋给的值（可以是组态王变量）

 例如：

SQLSetStatement(ConnectionID, “select * from kingview where highth=?”);

SQLPrepareStatement(ConnectionID, handle);

SqlSetParamFloat(handle, 1, var1); /*1 表示第一个问号，var1 是组态王中的变量*/

SQLSetParamInt

为指定的整数型参数赋值。

语法：

[ResultCode=]SQLSetParamInt(SQLHandle,ParameterNumber, ParameterValue);

参数描述

SQLHandle 句柄

ParameterNumber 语句中参数出现的序号

ParameterValue 赋给的值（可以是组态王变量）

 例如：

SQLSetStatement(ConnectionID, “select * from kingview where agg=?”);

SQLPrepareStatement(ConnectionID, handle);

SqlSetParamInt(handle, 1, var2); /*1 表示第一个参数序号，var2 是组态王中的变量*/

SQLSetParamNull

把指定的参数赋成空。

语法：

[ResultCode=]SQLSetParamNull(SQLHandle, ParmeterNumber, ParameterType, Precision, Sclae);

SQLSetStatement

启动一个SQL语句缓存区。

语法：

[ResultCode=]SQLSetStatement(DeviceID, "SQLStatement");

参数描述

DeviceID SQLConnct()产生的连接号

SQLStatement SQL语句

例如：

SQLSetStatement(DeviceID, “Select LotNo, LotName from LotInfo”);

例如：

SQLSetStatement(DeviceID, “select Speed from kingview”);

SQLExecute(DeviceID, “BIND”, 0);

上例中，没有 SQLPrepareStatement()准备的语句调用，句柄设为 0。

例如：

SQLSetStatement(DeviceID, “select Speed from kingview”);

SQLPrepareStatement(DeviceID, handle);

SQLExecute(DeviceID, “BIND”, handle);

SQLClearStatement(DeviceID, handle);

SQLTransact

定义了一组访问指令。这组指令暂不执行，直到用SQLCommit()函数提交执行。

语法：

[ResultCode=]SQLTransact(DeviceID);

参数描述

DeviceID SQLConnct()产生的连接号

SQLUpdate

使用组态王的变量值修改数据库中的记录。

语法：

[ResultCode=]SQLUpdate(DeviceID, "TableName", "BindList", "WhereExpr");

参数描述

DeviceID SQLConnct()产生的连接号

TableName 表格名

BindList 记录体

WhereExpr 指定函数起作用行的条件

注意：如果列名是字符串，表达式必须在单引号中。

下例选择“名字”列中等于Asia的行：

名字= ’ Asia’

下例选择“年龄”列中在20和30之间的行：

年龄>=20 and 年龄<30

 例如：用组态王的当前变量更新kingview表格中所有agg=20的行。

SQLUpdate(DeviceID, “kingview”, “bind1”, “agg=20”);

SQLUpdateCurrent

使用组态王中的变量更新数据库中当前行的记录。如果数据库中有其它行与当前行（ SQLSelect 或 SQLExecute ）指定的字段值（即组态王变量值）相同，则数据库中相同项将全部被更新为当前组态王变量值。

语法：

[ResultCode=]SQLUpdateCurrent(DeviceID, "TableName");

参数描述

DeviceID：SQLConnct()产生的连接号

"TableName"：数据库的表名

Sqrt

此函数用于计算变量值的平方根，调用格式：

Sqrt(Number);

变量值的类型可为整型、模拟量、离散量，变量值为正数时，函数返回值有效，变量值为负数时，函数返回值无效，

StartApp

此函数用于启动另一窗口应用程序。为确保能启动应用程序，请在应用程序名前使用全路径。路径使用DOS名称，即在DOS下显示的路径名。

调用格式：

StartApp("命令行参数")；或 StartApp("应用程序名")；

 例如：

StartApp(“c:\programfiles\microsoftoffice\office\excel report.xls”);

启动Excel，并自动打开电子数据表"Report.XLS"。若不想自动打开,则只需：

StartApp("c:\program files\microsoft office\office\excel ");

StopApp

此函数与 StartApp 相对应，用来关闭第三方软件。

调用格式：

Void StopApp(string cmd,string type);

参数描述：

cmd：要关闭的第三方软件信息.

type:cmd 的信息类型.

返回值：无

该函数提供四种可选参数用于关闭第三方软件,其含义由 type 参数指出，说明如下:

Type 值	cmd 含义	应用举例（以简体中文操作系统下关闭组态王信息窗口为例）
1	窗口标题 caption	StopApp("信息窗口", 1);
2	类名 class	StopApp("kingmess", 2);
3	执行文件 exeFile	StopApp("KingMess.exe", 3);
4	进程 ID PID	StopApp("5068", 4);//ID 可通过任务管理器进程 PID 列获得

StrASCII

此函数返回某一指定的字符串变量首字符的ASCII值。调用格式：

IntegerResult=StrASCII(Char);

参数描述

Char字母表顺序的字符或组态王字符串变量。

Char首字符的ASCII值将返回到IntegerResult中，当此函数执行时，只有单个

字符被检测或受到影响。如果字符串变量提供给 StrASCII 字符多于一个，只有变量的首字符会被检测。

例如：

StrASCII("A");//返回 65

StrASCII("A Mixer is Running");//返回 65

StrASCII("a mixer is running");//返回 97

StrChar

此函数返回某一指定ASCII码所对应的字符。调用格式：

MessageResult=StrChar(ASCII);

参数描述

ASCII　ASCII码或“组态王”字符串变量。

ASCII变量对应的字符将返回给MessageResult。此函数的一个用处是可以不用键盘给字符串变量添加字符。

例如：

ControlString=MessageTag+StrChar(13)+StrChar(10);

将一个[CR]和[LF]加到 MessageTag 的末尾，并且传递给了 ControlString。插入 ASCII 码在 32-126 范围之外的字符对于创建外设(例如：打印机或调制解调器)的控制代码是非常有用的。

StrFromInt

此函数将一整数值转换为另一进制下的字符串表示。调用格式：

MessageResult=StrFromInt(Integer,Base);

参数描述

Integer 要转换的数。数字或组态王 的整型变量。

Base 用来转换的进制。数字或 组态王的整型变量。

Integer被转换成指定的进制，结果将存在MessageResult中。

 例如：

StrFromInt(26, 2);//返回"11010"

StrFromInt(26, 8);//返回 "32"

StrFromInt(26, 16);//返回"1A"。

StrFromReal

此函数将一实数值转换成字符串形式，该字符串以浮点数计数制表示或以指数计数制表示。调用格式：

MessageResult=StrFromReal(Real,Precision,Type);

参数描述

Real 根据指定 Precision 和 Type 进行转换，其结果保存在MessageResult中。

Precision 指定要显示多少个小数位。

Type 确定显示方式，可为以下字符之一：

"f" 按浮点数显示

"e" 按小写“e”的指数制显示。

"E"　　按大写“E”的指数制显示。


例如：

StrFromReal(263.355, 2,"f");//返回 "263.36"

StrFromReal(263.355, 2,"e");//返回 "2.63e2"

StrFromReal(263.55, 3,"E");//返回 "2.636E2"

StrFromTime

此函数将一个时间值(1969年12月31日16:00起，以秒为单位)转换成字符串。调用格式：

MessageResult=StrFromTime(SecsSince1_1_70, StringType);

参数描述

SecsSince1_1_70　　转换为指定的 StringType 类型，结果保存在 MessageResult 中。

StringType　　确定显示方式，可为以下值之一：

1　　以 Windows 控制面板相同的格式显示日期。

2　　以 Windows 控制面板相同的格式显示时间。

3　　同时显示日期和时间


例如：

StrFromTime(86400, 1);//返回 "1/2/70"

StrFromTime(86400, 2);//返回 "12:00:00 AM"

StrFromTime(86400, 3);//返回 " 1/2/70 12:00:00 AM "。

StrInStr

此函数返回SearchFor在Text中第一次出现的位置。调用格式：

IntegerResult=StrInStr (Text,SearchFor,StartPos,CaseSens);

参数描述

Text 用于查找 SearchFor 的文本，若有多个SearchFor出现，则将其第一个的位置返回给 IntegerResult。

SearchFor 查找对象文本。

StartPos 用来确定在 Text 中开始查找位置的整数。

CaseSens 用来确定此查找是否对大小写敏感(0=不，1=是)。

 例如：

StrInStr("The mixer is running", "mix", 1, 0);//返回 5

StrInStr("Today is Thursday", "day", 1, 0);//返回 3

StrInStr("Today is Thursday", "day", 10, 0);//返回 15

StrInStr("Today is Veteran's Day", "Day", 1, 1);//返回 20

StrInStr("Today is Veteran's Day", "Night", 1, 1);//返回 0。

StrLeft

此函数返回指定字符串变量的开始(或最左的)若干个字符。调用格式：

MessageResult=StrLeft(Text,Chars);

参数描述

Text 实际文本字符串或字符串变量名。

Chars 要返回的字符个数。若Chars置为0，则返回全部字符串。

例如：

StrLeft("The Control Pump is On", 3);//返回 "The"

StrLeft("Pump 01 is On", 3);//返回 "Pump"

StrLeft("Pump 01 is On", 96);//返回 "Pump 01 is On"

StrLeft("The Control Pump is On", 0);//返回 "The Control Pump is On"。

StrLen

此函数返回指定字符串变量的长度。调用格式：

IntegerResult=StrLen(Text);

参数描述

Text　实际文本字符串或字符串变量名。文本的长度(字符数)返回给IntegerTag。所有字符串变量中的字符，包括那些在屏幕上不以能显示的字符都将被计算。

例如：

StrLen("Twelve percent");//返回 14

StrLen("12%");//返回 3

StrLen("The end. [CR]");//返回 10，[CR] 是回车符- ASCII 13。

此函数将指定文字中的所有大写字母转换为小写字母。小写字母、标号、数字和其它特殊字符将不受影响。调用格式：

MessageResult=StrLower(Text);

参数描述

Text　　实际文本字符串或字符串变量名

例如：

StrLower("TURBINE");//返回 "turbine"

StrLower("22.2 Is The Value");//返回 "22.2 is the value."

StrMid

此函数从指定的位置开始，从一个字符串变量中返回指定个数的字符。此函数与它的对应函数 StrLeft() 和 StrRight() 函数稍有不同，它允许工程人员指定要从字符串变量中抽取字符串的首尾位置。调用格式：

MessageResult=StrMid(Text,StartChar,Chars);

参数描述

Text　　实际文本字符串或字符串变量名。

StartChar　　指定要抽取的首字符位置。

Chars　　指定要返回字符的全部个数。

例如：

StrMid("The Furnace is Overheating",5,7,);//返回 "Furnace"

StrMid("The Furnace is Overheating",13,3);//返回 "is "

StrMid("The Furnace is Overheating",16,50);//返回 "Overheating"

StrReplace

此函数替换或改变所提供字符串的指定部分。使用此函数能获取字符串变量并替换字符、单词或短语。调用格式：

MessageResult = StrReplace(Text,SearchFor, ReplaceWith, CaseSens, NumToReplace, MatchWholeWords);

参数描述

Text	要改变的字符串。
SearchFor	要查找并替换的字符串。
ReplaceWith	替换字符串。
CaseSens	确定查找是否大小写敏感。(0=不，1=是)
NumToReplace	确定要替换的次数。(0=全部)
MatchWholeWords	确定此函数是否要全字匹配。(0=不，1=是)

 例如：

StrReplace("In From Within","In","Out",0,1,0); //返回 "Out From Within" (只替换第一个)

StrReplace("In From Within","In","Out",0,0,0) ; // 返回 "Out From WithOut" (全部替换)

StrReplace("In From Within","In","Out",1,0,0) ; // 返回 "Out From Within" (大小写匹配的全部替换)

StrReplace("In From Within","In","Out",0,0,1) ; // 返回 "Out From Within" (全字全部替换)

StrReplace() 函数不能识别特殊字符，如 @#$%&*()。函数将它们视为分隔符。例如如，若函数 StrReplace (abc#,abc#,1234,0,1,1) 执行，将不发生替换。“#”标号被识别为一个分隔符，而非字符。

StrRight

此函数返回指定字符串变量的最末端(或最右)若干个字符。调用格式：

MessageResult=StrRight(Text,Chars);

参数描述

Text　要查找的文本。字符串或 组态王 字符串变量。

Chars　返回字符的个数。字符串或 组态王 整型变量。若Chars置为0，则将返回全部字符串。

例如：

StrRight("The Pump is On", 2);//返回 "On"

StrRight("The Pump is On", 5);//返回 "is On"

StrRight("The Pump is On", 87);//返回 "The Pump is On"

StrRight("The Pump is On", 0);//返回 "The Pump is On"。

StrSpace

此函数在字符串变量中或表达式中产生一个空格串。调用格式：

MessageResult=StrSpace(NumSpaces);

参数描述

NumSpaces　数字或整型变量，指定产生的空格数。

例如：

所有的空格用"x" 表示：

StrSpace(4);//返回 "XXXX"

"Pump" + StrSpace(1) + "Station" 的结果是： "Pumpx Station"。

StrToInt

此函数将一个由数字组成的字符串转换成一个能用作数学计算的整数值。调用格式：

IntegerResult=StrToInt (Text);

参数描述

Text函数将处理的字符串。字符串或 组态王 的字符串变量。

当此语句被计算时，函数将读出此字符串中首字符的数字值。若首字符不是一个数字(空格忽略)，则字符串的值等于零(0)。若首字符是一个数字，则函数继续读后续的字符，直到遇到一个非数字值为止。

 例如：

If Text="ABCD",then IntegerTag=0.

If Text="22.2 is the Value", then IntegerTag=22 (因为整数是数字)。

If Text="The Value is 22", then IntegerTag=0.

StrToReal

此函数将一个由数字组成的字符串转换成一个能用于数字计算的实数值。调用格式：

RealResult=StrToReal(Text);

参数描述

Text函数将处理的字符串。

当此函数被调用时，函数将读出此字符串中首字符的数字值。若首字符不是一个数字(空格忽略)，则字符串的值等于零(0)。若首字符是一个数字，则函数继续读

后续的字符，直到遇到一个非数字值为止。

 例如：

If Text="ABCD", then RealTag=0.

If Text="22.261 is the Value", then RealTag=22.261.

If Text="The Value is 22", then RealTag=0.

StrTrim

此函数删除字符串变量中无用的空格。调用格式：

MessageResult=StrTrim(Text,TrimType);

参数描述

Text　函数将处理的字符串。字符串或组态王中的字符串变量。

TrimType　删除方式，可为下列类型之一：

1　删除首部空格(第一个非空格字符的左边)

2　删除尾部空格(最后一个非空格字符的右边)

3　删除单词间单个空格外的多余空格

Text 被用来查找要删除的空白(ASCⅡ码0x9-0x01或者ox20)。

 例如：

所有的空格用 "x" 代表。

StrTrim("xxxxxThisxisxaxxtestxxxxx",1);//返回 "Thisxisxaxxtestxxxxx"

StrTrim("xxxxxThisxisxaxxtestxxxxx", 2);// 返回 "xxxxxThisxisxaxxtest"

StrTrim("xxxxxThisxisxaxxtestxxxxx", 3);//返回 "Thisxisxaxtest"

StrReplace() 函数可用于从某一指定字符串变量中消除所有的空格，用“null”简单地替换所有空格。

StrType

此函数检测字符串变量的首字符以确定其是否为某一类型。调用格式：

DiscreteResult=StrType(Text,TestType);

参数描述

Text 函数将处理的字符串或字符串变量。

TestType字符类型，确定为下列类型之一：

1 字母数字符 ('A'-'Z', 'a-z' 和 '0-9')

2 数字符 ('0'- 9')

3 字母 ('A-Z' 和 'a-z')

4 大写字母 ('A'-'Z')

5 小写字母 ('a'-'z')

6 标点字符 (0x21-0x2F)

7 ASCII 字符 (0x00 - 0x7F)

8 十六进制字符 ('A'-'F' 或 'a'-'f' 或 '0'-'9')

9 可打印字符 (0x20-0x7E)

10 控制字符 (0x00-0x1F 或 0x7F)

11 空白符(0x09-0x0D or 0x20)

若Text中首字符是由TestType指定的类型，则StrType() 函数将返回给DiscreteResult一个正值。正如在其它函数中，单个字符被检测或影响一样，若StrType() 函数的字符串变量含有一个以上字符时，只有变量的首字符将被检测。

例如：

StrType("ACB123",1);//返回 1

StrType("ABC123",5);//返回 0。

StructVarRefAddress

该函数为实现结构变量间的引用，结构变量可以引用成员数相同、成员类型相同的其它结构变量。一般用于对于多组定义相同的IO变量，定义一组内存变量，在组态王中使用这一组内存变量定义画面显示的情况。调用格式：

StructVarRefAddress(RefStructTagname, RefedStructTagname);

参数说明：

RefStructTagname：字符串型　　引用的结构变量名称（不带成员名）

RefedStructTagname：字符串型　被引用的结构变量名称（不带成员名）

例如：

电力监控系统中，有多台变压器，有电压、电流、功率等数据。要求做一幅画面，在不同时刻分别显示各变压器的电压、电流、功率值。这个可以通过引用变量来完成。

定义一个结构Transformer，其中包含三个成员变量I、V、P，分别定义结构变量：Transformer1、Transformer2，这两个个结构变量的成员I、V、P均为IO变量，定义结构变量Transformer3，其成员I、V、P均为内存变量，在定义画面的动画连接时使用Transformer3变量，可以在命令语言脚本中使用变量引用函数分别引用两个IO变量。

要显示Transformer1的数据时：

StructVarRefAddress(“Transformer3”, “Transformer1”);

要显示Transformer2的数据时：

StructVarRefAddress(“Transformer3”, “Transformer2”);

StrUpper

此函数将指定字符串变量中所有的小写字符转换成大写字符。大写字符、符号、数字以及其它特殊字符将不受影响。调用格式：

MessageResult=StrUpper(Text);

参数描述

Text 函数将处理的字符串或字符串变量。

 例如：

StrUpper("abcd");//返回 "ABCD."

StrUpper("22.2 is the value");//返回 "22.2 IS THE VALUE"

StopBackupStation

此函数用于在组态王进行网络历史数据备份合并时人为中断备份过程。

语法使用格式：

BOOL StopBackupStation(str szStationName);

参数：远程站点名称。

 例如：

StopBackupStation(“IO采集站”);

Sum

此函数为对指定的多个变量求和。语法使用格式如下：

Sum（’a1’,’a2’）；

A1、a2 为整型或实型变量。其中参数个数为 1-32 个。

当对报表指定单元格区域内的单元格进行求和运算时，显示到当前单元格内。单元格区域内出现空字符、字符串等都不会影响求和。语法使用格式如下：

Sum（‘单元格区域’）

例如：=Sum（‘a1’,‘b2’,‘r10’）； 任意单元格选择求和

=Sum(‘b1:b10’)； 连续的单元格求和

Tan

此函数用于计算变量值的正切值, 调用格式：

Tan(变量值);

例如：

Tan(45);//返回值为 1

Tan(0);//返回值为 0。

Text

此函数变量名显示一个基于指定Format_Text格式的模拟(整型或实型)变量名。调用

格式：

MessageResult=Text(Analog_Tag,Format_Text);

参数描述

Analog_Tag	要转换的模拟变量
Format_Text	转换所使用的格式

例如：

MessageTag=Text(66,"#.00");

MessageTag 是一个文字类型变量名，66 为一个整型或者实型的变量名。“#0.00”代表显示格式：

若 Analog_Tag=66, 则 MessageTag=66.00。

若 Analog_Tag=22.269, 则 MessageTag=22.27。

若 Analog_Tag=9.999, 则 MessageTag=10.00。

Time

此函数为根据给出的时、分、秒整型数，返回时间字符串，默认格式为：时：分：秒。语法使用格式如下：

Time（LONG nHour, LONG nMinute, LONG nSecond）;

例如：时、分、秒变量分别为：“$时”、“$分”、“$秒”，用“时间”来显示由以上三个整数决定的“$时间”字符串，则在命令语言中输入：时间=Time（时,分,秒)。

Trace

此函数为调试函数，即系统运行时，利用该函数将值按照指定的形式显示在信息窗口中。调用格式：

Trace(‘test = %2d’, Express);

即把表达式Express的值按照十进制整数格式输出到信息窗口中，若Express=100，信息窗口将显示“test=100”。字符串“test”也可由用户指定。关于输出格式请参见下表：

字符	类型	输出格式
D	整型	区分正负的十进制整数
X(小写)	整型	不区分正负的十六进制整数，输出为小写，例如 ‘abcdef.’
X(大写)	整型	不区分正负的十六进制整数，输出为大写，例如‘ABCDEF.’
e	双精度型	区分正负的形式为[-]*d.dddd* e[*sign*]*ddd* 的数。 *d* 为一个十进制数字，*dddd* 是一个或多个十进制数字，*ddd* 是三位十进制数字，*sign* 为+或-。 例如-1.000000e+002
E	双精度型	与 e 字符一样，只是指数用 E 表示，而不是 e，例如-1.000000E+002
F	双精度型	区分正负的形式为[-]*dddd.dddd* 的数。

		Dddd 是一个或多个十进制数字，小数点前的数字位数由取决于数据的大小，小数点后的数字位数取决于要求的精度。
S(大小写均可)	字符串型	将字符串型变量以字符串方式输出。

Trunc

通过删去小数点右边部份的方式截取一个实数。调用格式：

ResultNumericTag=Trunc(Number);

参数描述

Number 任一数字或Kingview 实型或整型变量名

此函数的结果与把一个实数变量的内容放到一个整型变量中的结果相同。

例如：

Trunc(4.3);//返回 4

Trunc(-4.3);//返回 -4。

VarRefAddress

该函数为实现普通变量间的引用，普通变量可以数据类型相同的其它变量。一般用于对于多组定义相同的IO变量，定义一组内存变量，在组态王中使用这一组内存变量定义画面显示的情况。调用格式：

VarRefAddress(RefTagName, RefedTagName);

参数说明：

RefTagName：字符串型　　引用的变量名称

RefedTagName：字符串型　被引用的变量名称

 例如：

电力监控系统中，有多台变压器，有电压、电流、功率等数据。要求做一幅画面，在不同时刻分别显示各变压器的电压、电流、功率值。这个可以通过引用变量来完成。

定义所有IO变量，Transformer1_I、Transformer1_V、Transformer1_P、Transformer2_I、Transformer2_V、Transformer2_P。定义内存变量Transformer3_I、Transformer3_V、Transformer3_P，在定义画面的动画连接时使用Transformer3_I、Transformer3_V、Transformer3_P变量，可以在命令语言脚本中使用变量引用函数分别引用两个设备的IO变量。

要显示Transformer1的数据时：

VarRefAddress（“Transformer3_I”，“Transformer1_I”）;

VarRefAddress（“Transformer3_V”，“Transformer1_V”）;

VarRefAddress（“Transformer3_P”，“Transformer1_P”）;

要显示Transformer2的数据时：

VarRefAddress（“Transformer3_I”，“Transformer2_I”）;

VarRefAddress（“Transformer3_V”，“Transformer2_V”）;

VarRefAddress（“Transformer3_P”，“Transformer2_P”）;

WindowSize

此函数用于对运行画面最大化、最小化的操作。避免了由于要使用最大化最小化按

钮在运行画面存在的蓝色目录条，与画面颜色不搭的问题，使画面显示更美观。应用此函数时，配合选中运行系统设置中的“最小化按钮”和“最大化按钮”，可以实现窗口最大化和最小化的切换。调用格式：

WindowSize(nFlag);

参数说明：

nFlag：窗口最大化、最小化控制。nFlag =0，最大化；nFlag =1，最小化

返回值：无

 例如：

运行窗口最小化：

WindowSize(1);

xyAddNewPoint

此函数用于在指定的X-Y轴曲线控件中给指定曲线添加一个数据点。

语法格式使用如下：

xyAddNewPoint ("ControlName", X, Y, Index);

参数说明：

ControlName：工程人员定义的X-Y轴曲线控件名称，可以为中文名或英文名。

X：设置数据点的x轴坐标值

Y：设置数据点的y轴坐标值

Index：给出X-Y轴曲线控件中的曲线索引号，取值范围0-7。

例如：

xyAddNewPoint ("反应罐液压-液位", 30, 20, 1);

此语句执行后在反应罐液压-液位控件中索引号为 1 的曲线上添加一个数据点，该点的坐标值为(30，20)，表示该点的反应罐液压是 30，反应罐液位是 20。

xyClear

此函数用于在指定的X-Y轴曲线控件中清除指定曲线。

语法格式使用如下：

xyClear("ControlName",Index);

参数说明：

ControlName：工程人员定义的X-Y轴曲线控件名称，可以为中文名或英文名。

Index： 给出X-Y轴曲线控件中的曲线索引号，取值范围0-7，当取值为-1时，则清除所有曲线。

例如：

xyClear ("反应罐液压-液位", 1);

此语句执行后在反应罐液压-液位控件中清除索引号为 1 的指定曲线。

亚控公司各地分支机构联系方式

北京亚控科技发展有限公司

地址：北京市海淀区知春路 113 号银网中心 A 座六层

邮政编码：100086

电话：（010）59309666

传真：（010）59309600

E-mail ： support@wellintech.com, sales@ wellintech.com

网 址：http://www.wellintech.com/

北京亚控科技发展有限公司上海分公司

地址：上海市凯旋路 3131 号明申大厦 2806~2807 室(28 层)

邮政编码：200030

电话：（021）54071618/1132

传真：（021）54071608

E-mail：salessh@ wellintech.com

北京亚控科技发展有限公司广州分公司

地址：广州市天河区天河北路 908 号高科大厦 B 栋 23A02 室

邮政编码：510630

电话：（020）38258408

传真：（020）38288201

E-mail：salesgz@ wellintech.com

北京亚控科技发展有限公司成都办事处

地址：成都市一环路南一段 22 号红瓦大厦 632 室

邮政编码：610041

电话：（028）028-85254844 85230925

传真：（028）85256152

E-mail：salescd@wellintech.com

北京亚控科技发展有限公司西安办事处

地址：西安市高新区锦业路 1 号都市之门 B 座 12 层 1208 室

北京亚控科技发展有限公司济南办事处

地址：济南市花园路 40 号火炬大厦 1106 室

邮政编码：710075
电话：（029）68596680　68596681
传真：（029）68596682
E-mail：salesxa@wellintech.com

邮政编码：250100
电话：（0531）88061305
传真：（0531）88909812
E-mail：salesjn@wellintech.com

北京亚控科技发展有限公司武汉办事处

地址：武汉武昌区珞珈山路 1 号珞珈山
大厦 B 座 907 室
邮编：430000
电话：（027）87165131
E-mail：saleswh@wellintech.com

亚控公司其它驻地机构联系方式请访问亚控公司网站。