Cat No. W09E-CN5-02

CX-Supervisor软件

软件版本 3.2

脚本语言参考

OMRON

注意

欧姆龙产品主要是适用于接受过专业培训的人员，而且仅用于手册中说明的应用目的。
下列约定用于解释本手册中的注意事项，并对其进行分类。请务必注意它们所规定的情况。

注意: 为感兴趣的人士提示如何更有效、更便捷地使用该软件产品。

注意:
表示如不注意，可能会对产品造成轻度或相对严重的受伤、损坏或错误操作。

警告:
表示如不注意，可能会导致人员严重受伤，甚至有生命危险。

商标和版权

MECHATROLINK 是 Yaskawa Corporation的注册商标
Trajexia是欧姆龙的注册商标。
EtherCAT是EtherCAT Technology Group的注册商标。
所有其他产品名称、公司名称、标志或在本手册中所提到的其他名称为他们它们各自所有者的商标。

版权

使用手册中所包含的信息不承担专利责任。此外，由于欧姆龙不断努力改进其高质量的产品，手册中的内容也会随之进行修改，恕不另行通知。在编写手册时，已尽可能考虑到所有注意事项，但对于仍然可能出现的错误或遗漏，欧姆龙将不承担任何责任。对于使用本手册中所包含的信息导致的损害也不承担任何责任。

第一部分
介绍

作为CX-Supervisor用户手册的补充材料，本参考手册介绍了脚本语言语法。其中提供CX-Supervisor脚本的语法的详细定义，当被对象和脚本使用时，它们可以驱动项目、页面、对象动作和CX-Supervisor表达式。

本参考手册中的示例使用的印刷规范如下：

- 脚本命令和保留字使用preferred case。可以是小写、大写或大小写混合。
- 点使用小写。对象使用大写。

本参考手册中使用以下词汇：

- 应用程序。一套可执行特定任务的文件，包括一个可执行文件。本参考手册使用涉及Windows应用程序的Microsoft Excel和Microsoft Word。
- 常量。脚本中的只取一个特定值的点或对象。
- 可执行。一个包含程序或命令的文件，扩展名为 '*.EXE'。
- 嵌套。在一个IF THEN ELSE/ELSEIF ENDIF结构中包含一个或多个与该结构相同的语句。
- 操作数。常量或点变量。
- 运算符。包括关系、算数、逻辑语句，比如'+', '<=' 或 'AND'.
- Or ('|'). 当有两个或两个以上相同形式的语法时，标志 '|' 用于代替'or'。
- 点类型。布尔型、整数型、实数型或文本型。
- 点变量。脚本中可以取不同值的点或对象。
- 字符串。文本形式中被引号(" ")限定的数据,可以被赋予一个点。
- '{' 和 '}' 大括号。 参数命令或错误被报告时必须插入大括号，大括号间存在空格时会报错。
- 'TRUE' 和 'FALSE'. 仅与布尔状态相关，布尔状态0为'FALSE' ，布尔状态1为 'TRUE'。

第二部分
表达式

本部分介绍脚本中表达式的使用方法。

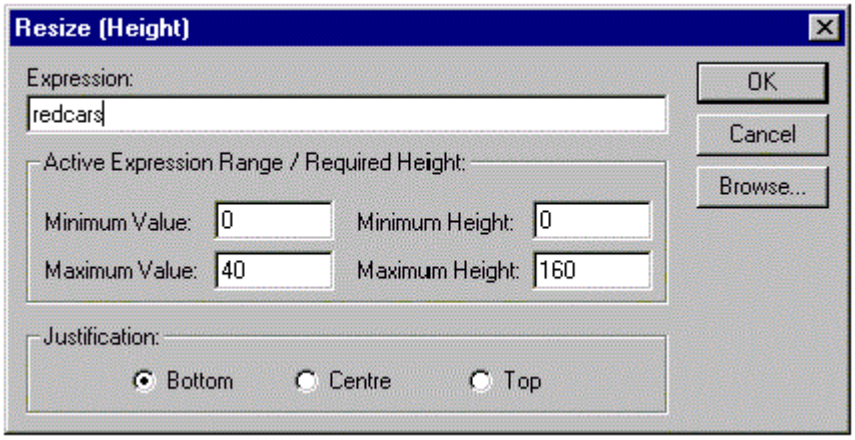
表达式包括操作数和运算符：

- 运算符是关系、算术、逻辑运算符并包括多种功能。
- 操作数是常量或点变量。

表达式可作为语句的一部分用于脚本（参考《第三部分 脚本》《第四部分 CX-Supervisor脚本语言》和《第六部分 功能和方法》）。但是，通过使用相关的表达式或数字表达式字段可以直接将表达式应用于下述动作：

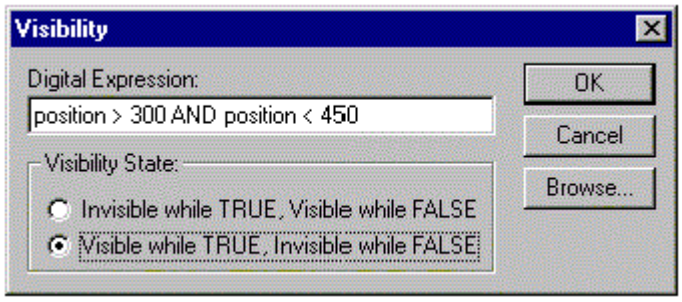
- 闪烁
- 关闭页面
- 颜色更改（模拟）
- 颜色更改（数字）
- 显示状态文本
- 显示文本点
- 显示值
- 编辑点值（模拟）
- 编辑点值（数字）
- 编辑点值（文本）
- 可用/禁用
- 水平移动
- 水平比例填充
- 调整高度
- 调整宽度
- 旋转
- 显示页面
- 垂直移动
- 垂直比例填充
- 可见性

接下来介绍一个简单表达式的示例，其中包含一个点（‘redcars’），该点通过调整高度这一对象操作附着于一个特定对象。运行时，一旦“有效表达式范围/所需高度”字段中的属性符合该点的值，当前对象将会据此调整大小。本示例是一个实数型或整数型示例，由此点的值可在指定范围内或外。在本示例中，为使表达式被满足，点‘redcars’必须落于0-40之间。

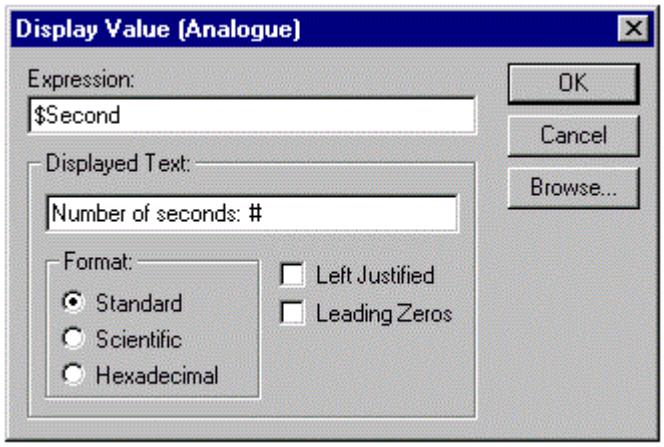


下面是一个略显复杂的表达式的示例，其中包括点‘position’的测试。如果‘position’的值多于300且少于450，即‘position’的值介于300到245之间，那么表达式被满足时，一个动作将会生效。 （在本示例中，如果表达式被满足，当前对象将会可见）。本示例为布尔型示例，由此表达式可能会被满足（‘TRUE’） 或不被满足(‘FALSE’)。布尔值总是由一个数字表达式字段返回 ，与返回一个整数或实数值的表达式字段不同。

有关本示例中使用的运算符，详见《第四部分 逻辑和算术》。



下面一个表达式的示例包括一个值点‘prompt’，包含于有‘#’标志标记的点位置。



参考CX-Supervisor用户手册以获取详细的对话框信息描述。

注意: 当布尔表达式正确时，它将会自动执行这个表达式。也就是说，每一个布尔表达式都有一个隐藏的"== TRUE"表达式。 有的时候布尔表达式很难被识别，比如 "bMyFlagPoint" 或 "BitMask & 0x80， 这时候如果 “== TRUE” 能明确的出现在表达式中， 这些布尔表达式就能被立刻识别且执行。比如: "bMyFlagPoint == TRUE" or "BitMask & 0x80 == TRUE".

注意: 当使用布尔操作符时(e.g. ==, !=, &&, ||, |)， 不要混淆对布尔操作数和非布尔操作数的测试。比如不要使用"bMyFlagPoint == 1" 或"bMyFlagPoint == 0"，而应该测试正确的布尔常数。比如对于CX-Supervisor脚本时应该使用布尔常数"TRUE" 或"FALSE" ，对于VBScript应该使用 "True" 和"False"。

注意: 当表达式正确时，条件脚本才能执行。有的时候这就会导致一些奇怪的结果。比如，当使用$Second表达式， $Second 是59或1时，它可以执行；当变为0时，就无法执行了。 任何时候当点改变时，执行条件脚本将会迫使表达式一直运行TRUE语句，比如 "$Second || TRUE"。因为当点改变时，$Second表达式将会一直进行测试，但|| TRUE则使得不管点的数值是多少，整个表达式的测试结果都为TRUE

注意: 在条件表达式中慎用数组点。表达式 "MyArray[3] == 1" 并不意味着当第三元素是1时执行表达式，而是当MyArray的任意元素改变且第三元素刚好为1时执行表达式。

注意: 在没有任何指引的情况下使用一个数组点就是指向0元素，例如 MyArray 就等于是 MyArray[0] == 1

第三部分
脚本

一个CX-Supervisor脚本时一个操作点的简单的编程语言。有各种不同类型的脚本，如对象脚本、页面脚本或是程序脚本。虽然脚本代码在这些不同的脚本中大致相同，但是接下来我们会讨论它们之间的细微的差别。

3-1 对象

如果当一个脚本是控制对象的执行时，那么这个脚本就能够影响对象的运行或者其他由脚本所实际控制的内容。

3-2 页面

页面脚本与页面上所使用的操作点和图形对象密切相关。换言之，页面脚本用于在特殊事件出现时驱动一系列功能，这些功能可以操纵页面上的图形对象。

3-3 项目

运用到程序中的脚本可以操控点。这类脚本与程序运行过程中所出现的事件相关。

第四部分
CX-Supervisor 脚本语言

本章讲述CX-Supervisor脚本语言语法，将对驱动项目、页面和对象的CX-Supervisor脚本的语法和对象及脚本所使用的CX-Supervisor表达式有详细的定义。CX-Supervisor脚本语言与第六章将学习到的脚本函数和脚本分析法相结合，是功能强大且完备、快捷灵敏的编译的程序语言。
下表时程序语言语法总览：

函数名称	函数类型	适用类型	备注
&, |, ^, <<, >>	逐位运算符	All	适用于逐位表达式
(objects)	语句	OP	明确测试或修改的对象的名称
(points)	语句	All	明确测试或修改的点的名称
+, -, *, /, %, =, ++, --	算术运算符	All	适用于算术表达式
<, >, <=, >=, ==, !=	关系运算符	All	适用于关系表达式
AND	逻辑运算符	All	适用于逻辑表达式
CALL	语句	All	调用子程序
DO LOOP WHILE UNTIL EXIT DO	语句	Scr	脚本片段将被重复
FALSE	布尔状态	Scr	适用于布尔表达式
FOR TO STEP NEXT EXIT FOR	语句	Scr	脚本片段将被重复
IFTHEN ELSE\ELSEIF ENDIF	语句	Scr	用于脚本测试
OR	逻辑运算符	All	适用于逻辑表达式
NOT	逻辑运算符	All	适用于逻辑表达式
REM	语句	Scr	备注 on line or lines of script.
RETURN	语句	Scr	终止脚本的顺序执行
SELECT CASE/ END SELECT	语句	Scr	适用于复杂测试
TRUE	布尔状态	Scr	适用于布尔表达式

“适用类型”一列指的是函数适用的脚本和语句的类型。“ALL”指脚本和语句均适用；“Scr”指仅适用于脚本；“OP”指仅适用于对象和页面。

4-1 点

4-1-1 基础点赋值

语法

```
pointname = expression
```

备注

参数	描述
pointname	被赋予数值的点名称
expression	赋予pointname的数值。表达式类型可以是布尔、整数、实数或文本

例如

```
count = 100
```

整数或实数点‘ count’ 被赋予数值100。

```
result = TRUE
```

布尔点‘result’ 被赋予语句“ TRUE” 。

```
name = "Valve position"
```

文本点‘name’ 被赋予引号内的相关的文本。

注意: 当将实数数值赋予整数点时，将使用四舍五入（趋向0）原则。也就是说，数值4.1将被赋值为数值4，数值-4.1将会被赋值为-4 。

参考:
引号的具体运用请参照《第四部分 标点》。

4-1-2 其他点赋值

语法

```
pointname = expression
```

备注

参数	描述
pointname	被赋予数值的点的名称
expression	赋予pointname的数值。表达式的类型可以是布尔、实数或整数，包括其他点，逻辑或算术表达式。 数学优先级如下: • 括号（优先级最高） • 一元操作符负号和逻辑运算符NOT • 乘法、除法和系数 • 加法和减法 • 关系运算符大于、小于、大于或等于、小于或等于 • 左移位（SHL）和右移位（SHR） • 关系运算符等于、不等于 • 逐位AND,XOR,OR • 逻辑运算符AND,逻辑运算符OR（优先级最低）

例如:

```
lift = height + rate/5.0
```

实数或整数点' lift' 被赋值为计算点' rate' 的数值除以5再加上点' height' 的数值。如有特别说明可改变优先级。

```
lift = lift - 0.2
```

实数或整数点' lift' 被赋值为计算当前点'lift' 的数值减0.2 。

```
distance = distance * time
```

实数或整数点' distance' 被赋值为计算当前点'distance' 的数值乘以点' time' 。
参考:
算术和逻辑函数的具体运用请参照《第四部分 逻辑和算术》; 优先级的运用请参照《第四部分 标点》。

4-2 逻辑和算术

4-2-1 算术运算符

语法

```
pointname = expression
```

备注

参数	描述
pointname	根据算术表达式所赋值的点的名称。
expression	赋予pointname的数值。表达式可以包含以下含点和常数的运算符: • 加法 '+'. • 减法 '-'. • 乘法 '*'. • 除法 '/'. • 系数 '%'. • 增量'++'. • 减量 '--'.

例如:

```
result = 60 + 20/5
```

实数或整数点' result' 被赋值为计算数值20除以5再加上数值60 。

```
lift = height + rate/5.0
```

实数或整数点' lift' 被赋值为计算点'lift' 的数值除以5再加上点' height' 的数值。如有特别说明可改变优先级。
参考:
优先级的具体运用请参照第四部分。

4-2-2 逐位运算符

语法

```
pointname = expression
```

或

```
IF expression
```

或

```
DO WHILE expression
```

或

```
DO UNTIL expression
```

备注

参数	描述
pointname	根据逐位运算所赋值的点的名称。
expression	赋值的pointname的数值，或是被认定为是布尔表达式。表达式可以包含以下含有点和常数的运算符： • 逐位 AND, 'BITAND' 或 '&' • 逐位 OR, 'BITOR' 或 '|'. • 逐位XOR, 'XOR' 或 '^'. • 逐位 Shift Left, 'SHL' 或 '<<'. • 逐位 Shift Right, 'SHR' 或 '>>'.

例如：

```
MSB = value & 128
```

当‘value’的二进位表示法的数值是128时，布尔点‘MSB’将运行‘TRUE’。

```
Pattern = value << 2
```

‘value’的二进位表示法的数左移两次储存在‘pattern’中。每一次左移位操作将使数值翻倍，因此两次左移位后数值翻四倍。

4-2-3 逻辑运算符

语法

```
pointname = expression
```

或

```
IF expression
```

或

```
DO WHILE expression
```

或

```
DO UNTIL expression
```

备注

参数	描述
Pointname	根据逐位运算所赋值的点的名称。

参数	描述
Expression	赋值pointname的布尔数值或构成条件语句的布尔数值。表达式包含以下含有点和常数的运算符： • 且'AND'. • 或'OR'. • 非 'NOT'.

例如：

```
flag = temp AND speed
```

布尔点‘flag’根据点‘temp’AND 点‘speed’的逻辑赋值。如果‘temp’且‘speed’都不是0，‘flag’运行1，或是“TRUE”。‘temp’或‘speed’中的任一数值为0，‘flag’将被赋值‘FALSE’或0 。

```
IF flag AND temp AND speed THEN
    flag = FALSE
ENDIF
```

当‘flag’且点‘temp’且point‘speed’均不为0时，布尔point‘flag’赋值为‘FALSE’。如果不满足条件，‘flag’将不会赋值为‘FALSE’。

参考：

IF THEN ELSE/ELSEIF ENDIF语句的具体运用请参照《第四部分 控制语句》。

4-2-4 关系运算符

语法

```
IF expression
```

或

```
DO WHILE expression
```

或

```
 DO UNTIL expression
```

备注

参数	描述
Expression	构成条件语句的数值。表达式可以包含以下含有点和常数的运算符： • 大于 '>'. • 小于 '<'. • 大于或等于'>='. • 小于或等于'<='. • 不等于 '!='. • 等于 '=='.

例如：

```
IF fuel < 0 THEN
    fuel = 0
ENDIF
```

当‘fuel’小于0时，point‘fuel’被赋值为0。当‘fuel’不小于0时，它也不会被赋予新的数值。
参考:
IF THEN ELSE/ELSEIF ENDIF语句的具体运用请参照《第四部分 控制语句》。

4-3 控制语句

4-3-1 简单条件语句

语法

```
IF condition THEN
    statementblock1
ENDIF
```

或

```
IF condition THEN
    statementblock1
ELSE
    statementblock2
ENDIF
```

备注

参数	描述
Condition	条件是使用关系、逻辑或算术符号将点和常数组合而成，用于测试。条件可以评估布尔语句‘TRUE’和‘FALSE’，实数或整数。或是文本字符串。
Statementblock1	当符合条件时所运行的一个或多个语句。
Statementblock2	当条件不符合时所运行的一个或多个语句。

例如:

```
IF fuel < 0 THEN
    fuel = 0
ENDIF
```

假设整数点‘fuel’小于0，那么它将被赋值为0 。

```
IF burner THEN
    fuel = fuel - rate
ENDIF
```

假设布尔点‘burner’是‘TRUE’，那么整数点‘fuel’被赋予一个新的数值。也可以将'IF burner == TRUE THEN'放置在第一行，将会获得相同的结果。

```
IF distance > 630 AND distance < 660 AND lift >= -3
THEN
    winner = TRUE
    burner = FALSE
ENDIF
```

假设整数点‘distance’的数值大于630且‘distance’的数值小于660（即‘distance’的数值在630到660之间）且点’lift‘大于或等于-3，那么布尔点‘winner’和‘burner’将被赋予新的数值。

```
IF burner AND fuel > 0 AND rate > 0 THEN
    fuel = fuel - rate
ELSE
    lift = 0
    altitude = 0
ENDIF
```

假设布尔点‘burner’是‘TRUE’且点‘fuel’和‘rate’的数值大于0，那么‘fuel’将被赋予一个新的数值。否则，点‘lift’和‘altitude’将被赋予新的数值。

参考:

代码布局的详细信息请参照《第四部分 标点、缩进》。

4-3-2 嵌套条件语句

语法

```
IF conditionA THEN
    statementblock1
    IF conditionB THEN
        statementblock3
    ENDIF
ELSE
    statementblock2
ENDIF
```

或

```
IF conditionA THEN
    statementblock1
    IF conditionB THEN
        statementblock3
    ELSE
        statementblock4
    ENDIF
ELSE
    statementblock2
ENDIF
```

或

```
IF conditionA THEN
    statementblock1
ELSEIF conditionB THEN
    statementblock3
ENDIF
```

或

```
IF conditionA THEN
    statementblock1
ELSE
    statementblock2
    IF conditionB THEN
        statementblock3
    ELSE
        statementblock4
    ENDIF
ENDIF
```

备注

参数	描述
conditionA	条件由点和常量构成，用关系符号、逻辑符号或算术符号测试。这个条件可以对布尔装态'TRUE'和'FALSE'、整数或实数或一个文本串求值。
conditionB	无论条件A求值成功与否，条件B都嵌入在条件A中。条件B由点和常量构成，用关系符号、逻辑符号或算术符号测试。这个条件可以对布尔装态'TRUE'和'FALSE'、整数或实数或一个文本串求值，嵌入条件语句的数量没有限制。
statementblock1	满足conditionA后进行的一个或一个以上的语句。
statementblock2	未满足conditionA后进行的一个或一个以上的语句。
statementblock3	满足conditionB后进行的一个或一个以上的语句。
statementblock4	未满足conditionB后进行的一个或一个以上的语句。

例如：

```
IF burner AND fuel > 0 AND rate > 0 THEN
    lift = lift + rate/5
ELSE
    count = 1
    IF altitude > 140 THEN
        lift = lift - 0.2
    ENDIF
ENDIF
```

假设'burner'点、'fuel'点和'rate'点已经求值成功,则'lift'点更新至最新值，这个值是‘rate’的值除以5后，与原‘lift’的值的和。
否则，需要对‘altitude’点进一步求值。如果目前‘altitude’点的值大于140，则‘lift’的值减0.2。

```
IF burner AND fuel > 0 AND rate > 0 THEN
    lift = lift + rate/5
ELSE
    IF altitude > 140 THEN
        lift = lift - 0.2
    ENDIF
ENDIF
IF burner AND fuel > 0 AND rate > 0 THEN
    lift = lift + rate/5
ELSEIF altitude > 140 THEN
    lift = lift - 0.2
ENDIF
```

这两个例子是相同的。使用ELSEIF语句能够简洁地合并ELSE语句和IF/ENDIF语句。在一个IF THEN ELSE/ELSEIF ENDIF架构中，可以允许一个以上的ELSEIF语句存在。

参考

使用缩进的详细信息请参考《第四部分 标点》。

4-3-3 Case Select

语法

```
SELECT CASE expression
    CASE expression
        statementblock1
    CASE expression
        statementblock2
    CASE expression
        statementblock3
END SELECT
```

或

```
SELECT CASE expression
    CASE expression
        statementblock1
    CASE expression
        statementblock2
    CASE ELSE
        statementblock3
END SELECT
```

备注

参数	描述
expression	表达式可能是一个点或常数，或者是一个常数的运算或有结果的点。
statementblock1	一个或一个以上只有在之前CASE表达式满足后才会运行的语句。
statementblock2	一个或一个以上只有在之前CASE表达式满足后才会运行的语句。
statementblock3	一个或一个以上只有在之前CASE表达式满足后才会运行的语句。

例如:

```
SELECT CASE colourvalue
    CASE 1
        colour (blue)
    CASE 2
        colour (green)
    CASE 3
        colour (cyan)
    CASE ELSE
        colour (0)
END SELECT
```

这个例子展示了如何根据点的值进行颜色的分配。整数点'colourvalue'被求值并与每一个case 对比直到配对成功。当一组配对成功，与CASE语句相关的一序列动作将会运行。当 'colourvalue' 值为1时，则分配给目前对象的颜色为蓝色；当'colourvalue'的值为2时，则分配给目前对象的颜色为绿色；当值为3时，则为青色。如果'colourvalue'的值分布在整数1到3范围之外,则颜色为0（黑色）。正如ELSE语句和ELSEIF语句一样，CASE ELSE语句是可选的。

```
SELECT CASE TRUE
    CASE temperature > 0 AND temperature <= 10
        colour (blue)
    CASE temperature > 10 AND temperature <= 20
        colour (green)
    CASE temperature > 20 AND temperature <= 30
        colour (red)
    CASE ELSE
        colour (white)
ENDSELECT
```

在此示例中，如同前例，除了将一个点作为条件之外，值也是条件。这个事例中布尔状态为"TRUE"且在每一个事例中整数点'temperature'都被测试。如果布尔状态为"TRUE"且'temperature'点的值在0和10之间，则目前对象的颜色被设置为蓝色。如果布尔状态为"TRUE"且'temperature'点的值在11和20之间，则目前对象的颜色被设置为绿色。如果布尔状态为"TRUE"且'temperature'点的值在21和30之间，则目前对象的颜色被设置为红色。如果没有一个CASE语句被满足，则目前对象的颜色被设置为红色L。如同ELSE和ELSEIF一样，CASE ELSE语句是可选择的。

参考

欲获得对对象应用属性的详细信息和使用颜色对象指令的详细信息，请参考《第六部分 对象指令》。欲获得调色板颜色的指定名称的详细信息，请参考《第八部分 调色板》。

4-3-4 FOR... NEXT Loop

语法

```
FOR pointname = startpt TO endpt STEP steppt
    statementblock1
NEXT
```

备注

参数	描述
pointname	作为循环计数器的pointname
startpt	pointname的最初设定,也是整个循环中第一个使用的值。
endpt	最后使用的值，当pointname超过这个值时，循环结束。
steppt	它是每一次循环后，使pointname增加的量。假设startpt比endpt值大，steppt可以向回数并且值可为负。在pointname每次循环都增加的case中，STEP的密码和变量可被省略（与添加 STEP 1相同）。

例如:

```
FOR loopcount = 0 TO 100
    Ellipse_1.vertical%fill = loopcount
NEXT
```

在此示例中, 'Ellipse_1' 被逐渐地赋了100次值。

```
FOR loopcount = 100 TO 0 STEP -5
    Ellipse_1.vertical%fill = loopcount
NEXT
```

这个例子中,Ellipse_1' 的赋值逐渐地被移除了20次 (100 times/-5)。

注意: 应谨慎使用循环语句。当它们运行时会耗费处理器的时间，由此系统的其它部分可能无法升级。

4-3-5 DO WHILE/UNTIL Loop

语法

```
DO WHILE expression
    statementblock
LOOP
```

或

```
DO
    statementblock
LOOP WHILE expression
```

或

```
DO UNTIL expression
    statementblock
LOOP
```

或

```
DO
    statementblock
 LOOP UNTIL expression
```

备注

参数	描述
expression	表达式可能是一个点或一个常数的计算，或是一个生成结果的点 。
statementblock	一个或以上的语句，它会根据表达式被执行很多次。

例如:

```
DO WHILE dooropen == TRUE
    Message ("You must shut the door before
    continuing")
LOOP
DO
    nextchar = Mid (Mystring, position, 1)
    position = position + 1
LOOP UNTIL nextchar = "A"
```

注意: 应谨慎使用循环语句。当它们运行时会耗费处理器的时间，由此系统的其它部分可能无法升级。

4-4 子程序

4-4-1 Call

语法

```
CALL subroutine (arguments)
```

备注

参数	描述
subroutine	在项目级别定义的子程序的名称。
arguments	子程序所需的参数列表，以逗号分隔。每个参数可能是一个pointname、常量、算术或逻辑表达式，或者是任何有效的合并。

例如：

```
CALL MySub ($Second, "Default", 2 + Int1)
```

4-4-2 Return

语法

```
RETURN
```

例如

```
IF limit > 1000 THEN
    RETURN
ELSE
    value = limit
ENDIF
REM final part of script
POLYGON_1.COLOUR = red
ELLIPSE_5.WIDTH = value
```

对整数点'limit'的值进行测试，如果它的值超过1000，则满足条件，执行RETURN指令，忽略所有在RETURN指令后的语句。如果整数点'limit'未超过1000，则不执行RETURN指令，运行所有在RETURN指令后的语句。

参考

欲获得使用针对Recipe validation的RETURN语句的方法，请参考CXSupervisor用户手册。

4-5 标点符号

4-5-1 指令字符串定界符

描述

可替代的字符串定界符允许字符串包含引用 " 字符。

语法

```
{Some "string" text}
```

例如：

```
Message({Error: "Invalid Function" occurred})
```

在整个字符串旁边插入的'{' 和 '}' 允许在字符串中的真正的文本包含引用，这个引用将会正常显示。他们可以在任何引用被使用的情况下使用，无论是否需要植入的引用。然而,，为了清晰起见，应优先使用引用字符。

4-5-2 缩进

例如：

```
IF burner AND fuel > 0 AND rate > 0 THEN
lift = lift + rate/5
```

```
ELSE
IF altitude > 140 THEN
lift = lift - 0.2
ENDIF
ENDIF
IF burner AND fuel > 0 AND rate > 0 THEN
    lift = lift + rate/5
ELSE
    IF altitude > 140 THEN
        lift = lift - 0.2
    ENDIF
ENDIF
```

两个例子具有相同的功能。但无论是显示了语句的构造的spaces还是tabs，它们都有助于可读性。
为清晰起见，这个例子省略了ELSEIF语句的使用。

4-5-3 多重指令

例如:

```
count = 75
result = log(count)
count = 75 : result = log(count)
```

两个例子具有相同的功能。但是在两个语句之间使用冒号可以使它们处于在同一行。

4-5-4 圆括号

例如:

```
result = 20 + 30 * 40
```

结果是1220.

```
result = (20 + 30) * 40
```

圆括号内的值先运算。结果为2000。

参考
欲知更详细的信息，请参考“第四部分，逻辑和算术，算术操作”。

4-5-5 引号

例如

```
name = "Valve position"
```

文本点 'name' 被赋了相关联的文本, 它在引号内。引号必须在这种情况下使用。

```
Message("This text to be displayed as a message.")
```

向方程传递作为参数的静态文本。

```
BlueCarsAck = IsAlarmAcknowledged("BLUEPAINT")
```

点'BlueCarsAck'被赋了一个在警报'BLUEPAINT'基础上的布尔状态。引号必须被用于一个警报的名称。

4-5-6 备注

例如:

```
REM | rem comment
```

或

```
'comment
```

备注

参数	类型	描述
Comment	---	描述性文本

例如:

```
REM The following statement adds two numbers
result = 45 + 754
result = 45 + 754  'add two numbers
```

4-6 脚本指令和表达式内的间接引用

在脚本和表达式内，使用文本点直接或间接地代替文字字符串的参数是可行的。例如，下列的每一项指令都具有相同的作用。

- 在文字上使用字符串;

```
PlayOLE("ole_1", 0)
```

- 直接使用文本点;

```
textpoint = "ole_1"
PlayOLE(textpoint, 0)
```

- 通过 '^' 符号间接使用文本点。

```
text = "ole_1"
textpoint = "text"
PlayOLE(^textpoint, 0)
```

在脚本指令内，使用文本点间接地代替point name参数是可行的。例如，下列的每一项指令都具有相同的作用。

- 直接使用一个点名称;

```
verbnumber = 0
PlayOLE("ole_1", verbnumber)
```

- 通过 '^' 符号间接使用文本点。

```
verbnumber = 0
textpoint = "verbnumber"
PlayOLE("ole_1", ^textpoint)
```

一个使用间接引用的例子
点间接引用的值可以被视为一种情况，即必须动态地改变一个与一个对象连接的pointname。如下的例子中，一个开关按钮被设置为控制四个点中之一的布尔状态。

- 四个被控制的布尔点的名称为 'motor1', 'motor2','motor3' and 'motor4'
- 文本点 'textpoint' 被用于存储被控制的布尔点的名称。
- 文本点'text'被用于储存整数点'index'的字符串值。
- 整数点'index' （范围为1-4）被用于动态改变被控制的点。
- 可以使用'^'符号和改变 'textpoint'点的内容以应用indirection，从而获取四个布尔点'motor1'、'motor2'、 'motor3'、'motor4' 中的任何一个。

比如，为了动态地改变布尔点，一个开关按钮需要按照如下的步骤进行操作。

1, 2, 3...

1. 使用间接引用，例如： ^textpoint，将开关按钮与文本点连接
2. 正如所需，将下列脚本代码与run连接。例如，点击一个按钮。
   - Text = ValueToText(index)
   - TextPoint = "motor" + text
3. ValueToText方程将点'index'的整数值转换为一个在文本点'text'中的字符串。因此，点'text'包括'1'、'2'、'3'或'4'中的任意一个。表达式'motor' + text将点'text'的内容附加在文字字符串'motor'上。因此，'textpoint'包含motor1'、'motor2'、'motor3'或'motor4'中的任何一个，这取决于'index'的值。通过更改'index'的值，决定控制哪个布尔点。例如，通过Edit Point Value (模拟) 动画。

4-7 脚本指令和表达式内的点阵列

在脚本或表达式中，直接或间接地获取在点阵列中的元素是可行的。

- 直接设定一个阵列点的值;

```
arraypoint[2] = 30
```

- 直接得出一个阵列点的值;

```
value = arraypoint[2]
```

- "使用间接引用设定一个阵列点的值;

```
textpoint = "arraypoint"
^textpoint[2] = 30
```

- 使用间接引用得出一个阵列点的值;

```
textpoint = "arraypoint"
value = ^textpoint[2]
```

使用点阵列的例子
阵列点的值可以被视为一种情况，这种情况是必须要动态地改变与一个对象连接的pointname。下面的例子中，一个开关点被预设为控制四个阵列点元素中的一个的布尔状态。
布尔阵列点'motor'被预设为包含四个元素。
整数点'index'（范围0-3）被用于动态改变被控制的点的元素。
为了动态地改变一个布尔点的元素都，开关按钮需要按照如下操作。

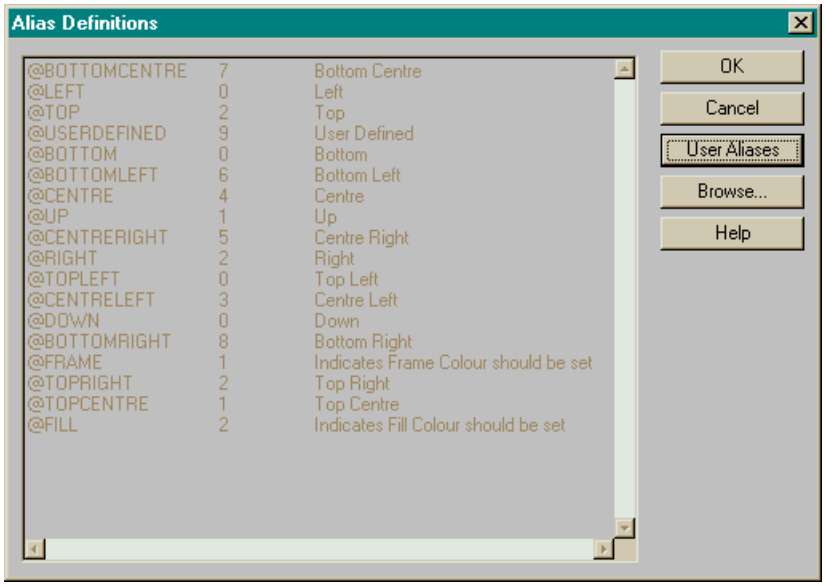
1, 2, 3...

1. 将开关按钮连接到阵列点上。比如，'motor[index]'.
2. 通过更改'index'的值，决定控制哪一个布尔点的元素。例如，通过编辑点值(模拟) 动画。

4-8 使用别名

这种简化用于宣布一个别名。也就是说定义一个文本字符串，这个文本字符串可以用于代替其他文本字符串或者在任何脚本或表达式内的数字。通过选择在"项目"菜单中的"Alias Definition..." 选项，可以显示别名定义对话框。 It 它同样可以显示，如果"Aliases..."被从脚本编辑器中选择。对话框显示用户定义的别名或者系统预设的别名，而且可以通过按User/System Alias按钮切换两种显示。

下面的图例展示了别名定义对话框，它显示了很多用户定义的别名。系统别名被预先定义且不能被编辑或添加内容。



语法：

```
@AliasNameAlias definition 'optional comment
```

备注：

参数	类型	描述
@AliasName	string	别名的字符串名称
Alias definition	string	这是一个字符串代表扩展的别名的实际文本或表达式。
' comment	string	这是一个可选择的评论。

在每一行开头的@符号发起每一个别名指令。比如文本字符串@SomePoint可以被用于表示在一个脚本或表达式中的任何charater序列。例如，它可以被定义为：

```
@SomePoint = InArray[1]
```

或甚至

```
@SomePoint = Inarray[1] + Inarray[2] / 2
```

这是一个识别点阵列中个体成员的简单方式。他也可以用于将名称和数字关联起来，比如：

```
@SecondsPerDay = 86400
```

别名定义被储存在一个位于项目目录的简单文本文件内，名称为<project name>.pre。文件的格式由任何数字或线构成，例如：

```
@Test1 = InArray[12] * 10
```

也就是说，别名的名称跟随一个@符号，别名的定义跟随一个等号（或空格）。任何跟随在撇号(')之后，且在一条线上的都被翻译为一个评论任何没有以@符号开始的线也被假定为评论。

例如

```
Declare boiler temperatures
@BoilerTemp1 = InArray[0] ' for boiler room 1
@BoilerTemp2 = InArray[1] ' for boiler room 2
```

```
@SecondsPerMinute  = 60   ' sets duration
```

别名也可以被用于创造一个复杂的脚本，比如：

```
@HYPOTENUSEsqrt(Opposite  *  Opposite  +  Adjacent  *
Adjacent) 'Calculates length of Hypotenuse
```

它们可以以如下的方式在一个脚本中使用：

```
Opposite = 8.45
Adjacent = 9.756
length = @HYPOTENUSE
```

其中，Opposite, Adjacent和length 皆为实数型点。

注意： 改变一个已经在表达式和脚本中使用的别名定义不会自动改变脚本中中的结果。应用别名的正确脚本或表达式为了应用更改，必须通过按OK按钮，被获取和重写。

第五部分 VBScript语言参考

这一部分是被称为VBScript的Microsoft Visual Basic脚本语言的语法的参考。这些功能都由Windows Scripting Host提供，默认包含Windows 2000和Windows XP。作为一本完整的用户指南，语言参考、最新版本的细节和支持请参考Microsoft，地址http://msdn.microsoft.com。

5-1 功能列表:

类型	关键词/功能
数组处理	Array Dim, Private, Public, ReDim IsArray Erase LBound, UBound
赋值	Set
注解	Comments using ' or Rem
常量	Empty Nothing Null True, False
控制流程	Do…Loop For…Next For Each…Next If…Then…Else Select Case While…Wend With
转换	Abs Asc, AscB, AscW Chr, ChrB, ChrW CBool, CByte CCur, Cdate CDbl, CInt CLng, CSng, CStr DataSerial, DateValue Hex, Oct Fix, Int Sgn TimeSerial, TimeValue
日期/时间	Date, Time DateAdd, DateDiff, DatePart DateSerial, DateValue Day, Month, MonthName Weekday, weekdayName, Year Hour, Minute, Second Now TimeSerial, TimeValue

类型	关键词/功能
声明	Class Const Dim, Private, Public, ReDim Function, Sub Property Get, Property Let, Property Set
错误处理	On Error Err
表达式	Eval Excute RegExp Replace Test
格式化字符串	FormatCurrency FormatDateTime FormatNumber FormatPercent
输入/输出	InputBox LoadPicture MsgBox
常量	Empty False Nothing Null True
数学	Atn, Cos, Sin, Tan Exp, Log, Sqr Randomize, Rnd
其他	Eval Function Execute Statement RGB Function
对象	CreateObject Err Object GetObject RegExp
运算符	Addition (+), Subtraction (-) Exporentiation (^) Modulus arithmetic (Mod) Multiplication (*), Division (/) Integer Division (\) Negation (-) String concatentation (&) Equality (=), Inequality (<>) Less Than (<), LessThan or Equal(<+) Greater Than (>) Greater Than or Equal To (>=) Is And, Or, Xor Eqv, Imp
选项	Option Explicit

类型	关键词/功能
过程	Call Function, Sub Property Get, Property Let, Property Set
四舍五入	Abs Int, Fix, Round Sgn
脚本引擎ID	ScriptEngine ScriptEngineBuildVersion ScriptEngineMajorVersion ScriptEngineMinorVersion
字符串	Asc, AscB, AscW Chr, ChrB, ChrW Filter, InStr, InStrB InStrRev Join Len, LenB LCase, UCase Left, LeftB Mid, MidB Right, RightB Replace Space Split StrComp String StrReverse LTrim, RTrim, Trim
变量	IsArray IsDate IsEmpty IsNull IsNumeric IsObject TypeName VarType

第六部分
函数和方法

这一章节介绍了可用于脚本语言的函数和方法。在大多数情况下，这可以是CX-Supervisor、VBScript或是JScript。
下列表格总体介绍了这些函数和方法。

函数名称	函数类型	类型	备注
AcknowledgeAlarm	警报命令	Scr	确认一个警报。
AcknowledgeAllAlarms	警报命令	Scr	确认所有警报。
AcknowledgeLatestAlarm	警报命令	Scr	确认最新的警报。
Acos	一元函数	All	应用一元表达式。
Asin	一元函数	All	应用一元表达式
Atan	一元函数	All	应用一元表达式
AuditPoint	数据记录命令	Scr	在CFR数据库中记录一个点值。
CancelForce	点命令	Scr	移除一个点上的强制值。
ChangeUserPassword	数据记录命令	Scr	更换用户的Windows密码。
Chr	文本命令	All	显示基于ASCII字符集上的一个字符。
ClearAlarmHistory	警报命令	All	清除警报历史。
ClearErrorLog	事件/错误命令	All	清除错误日志。
ClearLogFile	数据记录命令	Scr	清除一个数据日志文件。
ClearSpoolQueue	打印机命令	All	放弃任何处于队列中的消息或警报。
close	对象命令	Scr	关闭一个指定页面。
CloseAlarmHistory	警报命令	All	关闭当前警报历史。
CloseAlarmStatus	警报命令	Scr	关闭当前警报状态。
CloseComponent	通信命令	All	为一个PLC关闭一个组件（例如：CX Server 组件）。
CloseErrorLog	对象命令	Scr	关闭当前打开的错误日志。
CloseFile	文件命令	Scr	关闭打开的文件。
CloseLogFile	数据记录命令	Scr	关闭一个数据日志文件。
CloseLogView	数据记录命令	Scr	关闭日志查看器。
ClosePLC	PLC 命令	Scr	关闭与PLC（可编辑逻辑控制器）的通信。
colour	对象命令	OP	给一个对象指定一种颜色。
CopyArray	点命令	All	复制一个数组的内容。

函数名称	函数类型	类型	备注
CopyFile	文件命令	Scr	复制一个指定的文件。
cos	一元函数	All	应用一元函数。
DeleteFile	文件命令	Scr	删除指定的文件。
disable	对象命令	OP	禁用一个对象。
DisableGroup	点命令	All	防止一组点读取和写入。
DisablePoint	点命令	Scr	禁止与一个点的通信。
display	对象命令	Scr	显示一个指定的页面。
DisplayAlarmHistory	警报命令	Scr	显示当前警报历史。
DisplayAlarmStatus	警报命令	Scr	显示所有当前警报的警报状态。
DisplayErrorLog	事件 命令	Scr	显示当前“错误日志”。
DisplayPicture	通用命令	Scr	重新加载一个图片对象的图像。
DisplayRecipes	配方命令	Scr	浏览程序中当前的菜单。
DownloadPLCProgram	PLC 命令	All	下载指定文件到PLC。
DownloadRecipe	配方命令	Scr	下载一个指定配方。
EditFile	文件命令	All	编辑一个指定文件。
EnableAlarms	警报命令	All	启用警报功能。
EnableErrorLogging	错误命令	Scr	所有动作仅限于“错误日志”
EnableGroup	点命令	All	允许一组点读取和写入。
EnableOLE	通信命令	Scr	允许使用OLE功能。
EnablePLC	通信命令	Scr	允许使用PLC功能。
EnablePoint	点命令	Scr	启用与一个点的通信。
EnablePrinting	打印机命令	All	允许警报或信息的打印。
ExportAndViewLog	数据记录命令	Scr	导出数据日志和数据视图。
ExportLog	数据记录命令	Scr	导出数据日志。
FileExists	文件命令	All	指定一个文件的存在。
Force	点命令	Scr	锁定一个点的值。
ForceReset	点命令	Scr	设定一个点值为0。
ForceSet	点命令	Scr	设定一个点值为1。
FormatText	文本命令	All	用C语言标准函数库格式设定符号添加文本。

函数名称	函数类型	类型	备注
GenerateReport	报告命令	All	生成一个基于报告模板的报告。
GetBit	点命令	All	检索一个点的位。
GetPerformanceInfo	通用命令	All	检索内部性能和诊断值。
GetPLCMode	PLC 命令	All	检索一个PLC的模式。
GetTextLength	文本命令	All	在一个文本点中指定字符的数量。
height	对象命令	OP	指定一个对象的高度。
horizontal%fill	对象命令	OP	指定一个对象的水平填充。
InputPoint	对象命令	Scr	读取一个点的值。
IsAlarmAcknowledged	警报命令	Scr	测试一个指定的警报是否已经被确认。
IsAlarmActive	警报命令	Scr	测试一个指定的警报当前是否有效。
Left	语句	Scr	从一个字符串的左边提取字符。
log	一元函数	All	计算一个数字的自然对数。
log10	一元函数	All	计算一个数字的以10为底的对数。
LogError	错误命令	Scr	用错误日志程序记录一个错误信息。
LogEvent	错误命令	Scr	用错误日志程序记录一个事件消息。
Login	安全命令	Scr	登录一个用户到运行中的应用程序中。
Logout	安全命令	Scr	从一个运行中的应用程序中注销一个用户。
Message	文本命令	Scr	在一个消息框中输出一个字符串。
Mid	文本命令	Scr	从一个字符串中提取一个子字符串。
move	对象命令	OP	移动一个对象。
MoveFile	文件命令	Scr	移动指定的文件。
OpenComponent	通信命令	All	为一个PLC打开一个组件（例如：CX Server 组件）。
OpenFile	文件命令	Scr	打开指定的文件。
OpenLogFile	数据记录命令	Scr	打开一个数据日志文件。
OpenLogView	数据记录命令	Scr	打开数据日志查看器。
OpenPLC	PLC 命令	Scr	打开与一个PLC的通信。

函数名称	函数类型	类型	备注
OutputPoint	点命令	Scr	显示一个点的当前值。
PlayOLE	通用命令	Scr	播放一个OLE对象。
PlaySound	通用命令	Scr	播放一个声音文件。
PLCCommsFailed	PLC 命令	All	指定是否PLC通信失败。
PLCMonitor	PLC 命令	Scr	监控一个PLC。
PointExists	点命令	All	指定一个点的存在。
PrintActivePage	通用命令	Scr	打印当前有效页。
PrintFile	文件命令	Scr	打印指定文件。
PrintMessage	文本命令	All	打印消息到已配置的“警报/消息打印机”。
PrintPage	通用命令	Scr	打印指定页面。
PrintReport	报告命令	All	打印一份报告。
PrintScreen	通用命令	Scr	打印当前的显示屏幕。
PrintSpoolQueue	打印机命令	All	打印所有处于队列中的警报或消息。
Rand	通用 命令	Scr	计算一个随机数字。
Read	文件命令	Scr	从一个打开的文件中读取数据到一个点。
ReadMessage	文件命令	All	从一个外部文件中读取文本。
Right	文本命令	Scr	从一个字符串的右边提取字符。
rotate	对象命令	OP	旋转一个对象。
RunApplication	通用命令	Scr	运行指定的应用程序。
RunHelp	通用命令	Scr	运行指定的帮助文件。
SelectFile	文件命令	All	指定一个文件名称和路径。
SetBit	点命令	All	设定一个点的一个指定的位。
SetPLCMode	PLC 命令	All	设定一个PLC的模式。
SetPLCPhoneNumber	PLC 命令	All	设置一个电话号码到一个PLC。
SetupUsers	安全命令	Scr	定义用户名和登录的密码。
ShutDown	通用命令	Scr	终止CX-Supervisor。
sin	一元函数	All	应用一元表达式。
sqrt	一元函数	All	应用一元表达式。
StartAuditTrail	数据记录命令	Scr	开始审计跟踪日志。
StopAuditTrail	数据记录命令	Scr	停止审计跟踪日志。
StartLogging	数据记录命令	Scr	启动一个数据集记录。

函数名称	函数类型	类型	备注
StopLogging	数据记录命令	Scr	停止一个数据设定日志。
tan	一元函数	All	应用一元表达式。
TCAutoTune	临时控制器命令	All	启动或停止一个温度控制器自动调整操作。
TCBackupMode	临时控制器命令	All	定义一个温度控制器如何储存内部变量。
TCGetStatusParameter	临时控制器命令	All	检索温度控制器状态参数。
TCRemoteLocal	临时控制器命令	All	定义一个温度控制器的操作模式。
TCRequestStatus	临时控制器命令	All	检索温度控制器状态。
TCReset	临时控制器命令	All	重新设置温度控制器。
TCRspLsp	临时控制器命令	All	定义温度控制器的设定值模式。
TCRunStop	临时控制器命令	All	定义自动输出模式切换或手动输出模式切换。
TCSaveData	临时控制器命令	All	储存与温度控制器有关的数据。
TCSettingLevel1	临时控制器命令	All	为温度控制器执行一个设置级别功能。
TextToValue	文本命令	Scr	将一个字符串转换为一个数值点值。
UploadPLCProgram	PLC命令	All	上传在PLC中的程序到指定文件。
ValueToText	文本命令	Scr	将一个数值转换为一个文本点。
vertical%fill	对象命令	OP	指定一个对象的垂直填充。
ViewReport	报告命令	All	显示一份报告。
visible	对象命令	OP	切换一个对象的可见性。
width	对象命令	OP	指定一个对象的宽度。
Write	文件命令	Scr	写入一个值到一个打开的文件。
WriteMessage	文件命令	All	写入文本到一个外部文件。

“类型” 栏是指函数可被适用的脚本类型和表达式类型。“ All” 是指表达式和脚本都适用。“Scr”是指只有脚本适用。“OP”是指只有对象脚本和页面脚本适用。

6-1 对象命令

对象命令控制本地的CX-Supervisor图形对象，例如矩形或者线条。

注意: 对象位于CX-Supervisor本地，因此不能被访问或是命令由外部脚本语言产生，例如VBScript或Jscript。

6-1-1 Current Object

语法

```
objectcommand
```

备注

参数	描述
	"表达式可由以下命令组成，详见《第六部分 对象命令》。 • Colour 命令 • Disable 命令 • Visible 命令 • Move 命令 • Rotate 命令 • Vertical fill 命令 • Horizontal fill 命令 • Height 命令 • Width 命令 命令的内容由算术或逻辑表达式、x和y坐标或引用组成，它们由于命令不同而有所改变。Colour 命令要求一个colour标识符。

例如:

```
colour (red)
```

当前对象被指定为红色。

引用

参考:

- 第六部分, Blink ——blink 命令的使用
- 第六部分, Colour—— colour 命令的使用
- 第六部分, Disable —— disable命令的使用
- 第六部分， Height ——height命令的使用
- 第六部分, Horizontal Fill —— horizontal fill命令的使用
- 第六部分, Move —— move 命令的使用
- 第六部分, Rotate —— rotate 命令的使用
- 第六部分,Vertical Fill —— vertical fill 命令的使用
- 第六部分, Visible —— visible 命令的使用
- 第六部分, Width —— width 命令的使用

"有关动画编辑器的详细信息，请参见CX-Supervisor 用户手册。

6-1-2 Other Objects

语法

```
objectname.objectcommand
pagename.objectname.objectcommand
```

备注

参数	描述
objectname	对象的名称。对象在创建时会拥有一个一般名称，该名称之后可以被修改并赋予含义。对象名称的任何更改都会引起脚本自动更新。
objectcommand	该参数可由以下命令组成，详见《第六部分 对象命令》。 • Colour 命令 • Disable 命令 • Visible 命令 • Move 命令 • Rotate 命令 • Vertical fill 命令 • Horizontal fill 命令 • Height 命令 • Width 命令 命令的内容由算术或逻辑表达式、x和y坐标或引用组成，它们由于命令不同而有所改变。Colour 命令要求一个colour标识符。

例如

```
POLYGON_1.colour (red)
POLYGON_1.colour = red
```

指定的对象'POLYGON_1' 被设置为红色。

参考

参考:

- 对象名称详见CX-Supervisor用户手册
- 第六部分, Blink ——blink 命令的使用
- 第六部分, Colour—— colour 命令的使用
- 第六部分, Disable —— disable命令的使用
- 第六部分， Height ——height命令的使用
- 第六部分, Horizontal Fill —— horizontal fill命令的使用
- 第六部分, Move —— move 命令的使用
- 第六部分, Rotate —— rotate 命令的使用
- 第六部分,Vertical Fill —— vertical fill 命令的使用
- 第六部分, Visible —— visible 命令的使用
- 第六部分, Width —— width 命令的使用

6-1-3 Blink

语法

```
objectname.blink (colour, status)
```

备注

参数	描述
objectname	对象的名字。当脚本直接附于对象上时，不需要objectname。
colour	颜色闪烁。调色板中的有些颜色有colourID，取自颜色的名字，比如黑色或者黄色。或者，0×1000000的一个整数值可以被添加到0~65之内的号码，以选择一个颜色板条目。
status	这项参数可省略。大概等同于: TURE-打开闪光 FALSE-关掉闪光 如被省略，则假定为TRUE。

例如

```
blink (red, TRUE)
```

开始闪烁红色

```
LINE_1.blink(0xFFFF00, status)
```

对象 LINE_1 根据布尔点 ‘status’开始或结束闪烁黄色。

6-1-4 Colour

语法

```
objectname.colour (expression, context)
colour (expression, context)
```

或

```
objectname.colour (colourID, context)
colour (colourID, context)
An equals sign may be used as an alternative to
brackets:
objectname.colour = expression
colour = expression
```

或

```
objectname.colour = colourID
colour = expression
```

注意: 'colour' 或 'color'的拼写皆可。

等号也可用于其他大部分对象命令，即便在本手册中未直接提到等号。

备注

参数	描述
objectname	对象的名称。当脚本直接附于对象上时，不需要objectname。
expression	表达式可以是一个整数点，或者一个常量的计算结果，和/或产生0-16777215之间的一个整数值的点。这就是颜色的RGB值。 (形式是 0xBBGGRR).

参数	描述
colourID	调色板中的有些颜色有colourID，取自颜色的名字，比如黑色或者黄色。或者，0×1000000的一个整数值可以被添加到0~65之内的号码，以选择一个颜色板条目。
context	此参数可选并可省略。它定义了对象的哪一部分会改变颜色。可以是以下一个或多个: @FILL - 改变填充色 @FRAME - 改变框线色 如果省略则表示二者都改变。相当于 @FILL | @FRAME

例如

```
TEXT_3.colour (blue)
```

或

```
TEXT_3.colour = blue
```

对象'TEXT_3' 设置为蓝色。

```
BALL.colour (35 + 0x1000000)
```

对象'BALL' 设置为调色板中的35号。

```
BALL.colour (0xFF0000,@FILL)
```

对象'BALL' 设置为蓝色。

```
shade = tint1 + tint2
IF shade > 65 OR shade < 0 THEN
    shade = 0
ENDIF
ELLIPSE_1.colour (shade + 0x1000000)
```

根据'tint1' 和 'tint2'，点'shade' 被设置为某个值，设定值后首先需测试以确保处于0-65的范围之内。如果'shade'超出范围，则该值不能作为颜色应用于对象，并重设为0（或黑色）。'ELLIPSE_1' 被设置为阴影值的调色板。

参考

参考《第六部分 调色板》，获取颜色名称和颜色号码的详细信息。

6-1-5 Disable

语法

```
objectname.disable (expression)
```

备注

参数	描述
objectname	对象的名称。脚本直接附于对象上时，不需要objectname。
expression	表达式由点组成，并导致'TRUE' 或 'FALSE'。

例如

```
disable (TRUE)
```

此例应用于当前按钮对象的命令被禁用。

```
PUSH_8.disable (count AND flag)
```

可选对象'PUSH_8' 被禁用，如果整数点'count' 且 布尔点 'flag' 返回 "TRUE"。

6-1-6 Height

语法

```
objectname.height (expression, context)
objectname.height = expression
```

备注

参数	描述
objectname	对象的名称。脚本直接附于对象上时，不需要 objectname。
expression	是一个值、点或以像素为单位返回新的高度值的算术表达式。
context	此参数可选并可省略。它定义了对象的数据和维持不变的部分。可以是以下一种： @TOP - 使用对象顶部作为数据 @CENTRE - 使用对象中心作为数据 @BOTTOM - 使用对象底部作为数据 如被省略，则假定为 @CENTRE

例如

```
height (100)
```

或

```
height = 100
```

当前对象高度设置为100.

```
LINE_1.height (stretch/offset, @top)
```

对象 'LINE_1'的高度更改为点'stretch' 和'offset'计算出的值，保持它所在的顶部位置。

6-1-7 Horizontal Fill

语法

```
objectname.horizontal%fill (expression, context)
```

备注

参数	描述
objectname	对象的名称。脚本直接附于对象上时，不需要 objectname。
expression	算术表达式。该表达式必须返回一个介于0-100的值。返回一个有效结果时，填充从左往右。
context	此参数可选并可省略。它定义了对象从哪一侧开始填充。可以是以下一个： @LEFT - fill from the left @RIGHT - fill from the right 如被省略，则假定为@LEFT

例如

```
horizontal%fill (50)
```

此例应用于当前对象的命令为填充50%。

```
ELLIPSE_1.horizontal%fill (GAS_LEVEL, @RIGHT)
```

对象'ELLIPSE_1'被从右始进行填充，假设点'GAS_LEVEL'返回介于0-100的有效结果。

6-1-8 Move

语法

```
objectname.move (x co-ordinate, y co-ordinate)
```

备注

参数	描述
objectname	对象的名称。脚本直接附于对象上时，不需要objectname。
x co-ordinate y co-ordinate	对于移动前的对象，对象移动到的位置（像素为单位）的x和y坐标以 (x, y)的形式指定。单独的点或算术表达式的一部分可以被用作此表达式的基础。

例如

```
move (100, 200)
```

此例应用于当前对象的命令为移动到特定位置。

```
POLYGON_1.move (xpos, ypos/5)
```

对象 'POLYGON_1' 被移动到由点'xpos' 和'ypos' 除以5指定的位置。

6-1-9 Rotate

语法

```
objectname.rotate (angle, context, fixed, xcoord, ycoord)
```

备注

参数	描述
objectname	对象的名称。脚本直接附于对象上时，不需要objectname。
angle	在顺时针方向，旋转角度介于0-360.单独的点或作为算术表达式的一部分可被用作一个角度。

参数	描述
context	此参数不需要并可被省略。可以是以下一种: @TOPLEFT - 围绕对象顶部左侧进行旋转。 @TOPCENTRE -围绕对象顶部中心进行旋转。 @TOPRIGHT - 围绕对象顶部右侧进行旋转。 @CENTRELEFT - 围绕对象中心左侧进行旋转。 @CENTRE - 围绕对象中心进行旋转。 @CENTRERIGHT - 围绕对象中心右侧进行旋转。 @BOTTOMLEFT - 围绕对象底部左侧进行旋转。 @BOTTEMCENTRE - 围绕对象底部中心进行旋转。 @ BOTTOMRIGHT - 围绕对象底部右侧进行旋转。 @USERDEFINED - 以x和y指定的用户定义的点。
fixed	此参数可被省略。如果此布尔值为真，旋转起点将固定于屏幕上，即便对象被移动。否则，旋转起点与对象位置相关。
xcoord ycoord	仅在指定@USERDEFINED时需要。这些整数变量以像素为单位指定旋转起点。

例如

```
rotate (45)
```

此例应用于当前对象的命令为旋转45.

```
RECTANGLE_1.rotate(tilt, @USERDEFINED, 0, -100, 10)
```

对象'RECTANGLE_1'根据 'tilt'的值旋转,围绕点 -100, 10 与对象当前位置相关。

```
rotate (a * sin(b))
```

当前对象基于包含点 ‘a’ and ‘b’.的算术表达式的计算结果进行旋转。

6-1-10 Vertical Fill

语法

```
objectname.vertical%fill (expression, context)
```

备注

参数	描述
objectname	对象的名称。脚本直接附于对象上时，不需要objectname。
expression	算术表达式。该表达式必须返回一个介于0-100的值。返回一个有效结果时，填充从底部往顶部。
context	此参数可被省略. 可以是以下的一种: @DOWN - 向下填充对象 @UP - 向上填充对象 如被省略，则假定为@UP is assumed

例如

```
vertical%fill (50)
```

此例应用于当前对象的命令为填充50%。

```
ELLIPSE_1.vertical%fill (OIL_QUANTITY, @DOWN)
```

对象 'ELLIPSE_1'被填充，假如点 'OIL QUANTITY' 返回一个介于0-100的有效值。

6-1-11 Visible

语法

```
objectname.visible–(expression)
```

备注

参数	描述
objectname	对象的名称。脚本直接附于对象上时，不需要objectname。
expression	表达式由点组成，并导致'TRUE' 或 'FALSE'。

例如

```
visible (TRUE)
```

此例应用于当前对象的命令为可见。

```
POLYLINE_8.visible (count AND flag)
```

对象'POLYLINE_8' 可见，假如整数点'count'且布尔点 'flag' 返回 "TRUE".

6-1-12 Width

语法

```
objectname.width (expression, context)
```

备注

参数	描述
objectname	对象的名称。脚本直接附于对象上时，不需要objectname。
expression	这是一个值、点或返回一个以像素为单位的宽度值的算术表达式。
context	此参数可被省略。可以是以下的一种: @LEFT - 使用对象左侧作为数据 @CENTRE - 使用对象中心作为数据 @RIGHT - 使用对象右侧作为数据 如被省略，则假定为@CENTRE 。

例如

```
width (150)
```

当前对象宽度设置为 150.

```
LINE_1.width (squeeze/offset, @RIGHT)
```

对象 'LINE_1'的宽度更改为点'squeeze' 和 'offset'计算的值, 保持最右端点固定。

6-2 Page Commands

显示页面

语法

```
display ("pagename")
```

或

```
display ("pagename", X, Y)
```

备注

参数	描述
pagename	基于没有扩展的文件名的显示的页面名称，如：CAR.PAG的 pagename简化为 'CAR'.

例如

```
display ("CAR")
```

显示页面 'CAR.PAG'

```
textpoint = "CAR"
display(textpoint)
```

显示页面 'CAR.PAG'

```
display("CAR", 100, 200)
```

页面'CAR.PAG' 显示在自定义位置，主窗口左侧的对面100像素以及从顶部向下200像素处。

6-2-1 Close Page

语法

```
close ("pagename")
```

备注

参数	描述
pagename	基于没有扩展的文件名的要关闭的页面名称。如：CAR.PAG 的pagename简化为'CAR'.。要关闭的pagename 当前必须是打开状态。

注意: 'close'操作将造成不加载页面，包括所有对象、ActiveX控件和脚本。关闭指令后要注意不要尝试去访问这些内容。

注意: 在页面中包含'close'指令的脚本将被关闭的地方，该指令应该是脚本的最后指令，因为它将结束加载脚本。

例如

```
close("CAR")
```

页面 'CAR.PAG' 被关闭。

```
textpoint = "CAR"
close(textpoint)
```

页面'CAR.PAG' 被关闭。

6-3 一般命令

6-3-1 Exponential

描述

计算被提高到乘方的值的数学函数。

语法

```
result = Exp (value, exponent)
```

备注

参数	类型	描述
result	整数	接收返回的结果的点名称，该结果是被提高到乘方的值。
value	整数	提高的数字。
exponent	整数	提高数字的乘方

例如

```
MSBMask = Exp (2, 15)
```

在此示例中， 'MSBMask'被赋值 215, 即 32,768.

6-3-2 PlayOLE

描述

在一个OLE2对象上开始一个OLE动词或'method'。动词号码据对象而定，请参考对象的文档。此函数已不太常用，因为现在的大多数对象是 ActiveX对象。

语法

```
returnstate = PlayOLE("objectname",OLEVerbNumber)
```

备注

参数	类型	描述
returnstate	布尔型	若函数成立则返回状态为1，否则为0。
objectname	字符串	将被播放的OLE对象的标识符。
OLEVerbNumber	实数	对 应用来说，动词编号有特定意义。例如： 0: 指定动作在终端用户在对象的容器中双击对象时发生。对象决定此动作(常见 'edit'或 'play'). -1: 为编辑或查看而引导对象展示出来。经常会给一些其他定义对象的动词起别名。 -2: 为在一个与对象的容器分离出的窗口中编辑而引导对象打开。 -3: 造成对象从查看中移走它的用户接口。只能应用于代替被激活的对象。 正编码指定对象特定的动词。

例如

```
PlayOLE("ole_1",0)
```

使用对象的主动作，对象'ole_1' 被播放。

6-3-3 DisplayPicture

描述

为图片对象重新加载图片。

语法

```
returnstate = DisplayPicture("objectname", filename)
```

备注

参数	类型	描述
returnstate	布尔型	若函数成立则返回状态为1，否则为0。
objectname	字符串	将要加载和显示的位图对象的标识符。
filename	字符串	将要显示的位图的filename。可以是一个常数（内含引用）或一个文本点。

例如

```
DisplayPicture("Bitmap_1","C:\Application\Floorplan1.
bmp")
```

对象"Bitmap_1" 将加载和展示Floorplan1 位图。

```
DisplayPicture("Bitmap_2", txtFileName)
```

位图"Bitmap_2"将加载和展示储存在 txtFileName文本点的文件名称。

6-3-4 PlaySound

描述

使用标准Windows声道和声卡驱动来播放Windows .WAV 声音文件。

语法

```
returnstate = PlaySound("soundfile")
```

备注

参数	类型	描述
returnstate	布尔型	若函数成立则返回状态为1，否则为0。
soundfile	字符串	声音文件被播放的路径。

例如

PlaySound("c:\noise.wav")

播放声音文件 "c:\noise.wav" 。

6-3-5 Rand

描述

返回一个任意整数，介于0到指定界限值之间。

语法

```
pointname = Rand(upperlimit)
```

备注

参数	类型	描述
upperlimit	整数	Rand函数可生成的最大负整数或正整数。 f
pointname	整数点	包含Rand函数返回的整数的点。

例如

```
randomnumber = Rand(upperlimit)
```

范围0到上限内的任意整数倍返回并包含于点 'randomnumber'. 。最大上限为32767.

注意: 如果 'upperlimit'是负数，那么范围是0到负数。

6-3-6 RunApplication

描述

要求正在操作的系统运行一个新程序。它将以一个分离的进程来运行，并且RunApplication不会等待应用程序启动。特定文件名称必须可执行，即有 .EXE、 .COM 或 .BAT扩展名。

语法

```
returnstate = RunApplication("executable")
```

备注

参数	类型	描述
returnstate	布尔型	若函数成立则返回状态为1，否则为0。
executable	字符串	可执行文件的路径名称。

例如

```
RunApplication("c:\myprog.exe")
```

可执行文件 c:\myprog.exe被运行。

6-3-7 RunHelp

描述

调用Windows Help引擎并加载一个帮助文件，展示特定的主题号码。

语法

```
returnstate = RunHelp("helpfile",helpindex)
```

Remarks

参数	类型	描述
returnstate	布尔型	若函数成立则返回状态为1，否则为0。
helpfile	字符串	将被运行的 helpfile的路径名称。
helpindex	整数	如被正在运行的帮助文件定义的，为帮助主题编索引。

例如

```
RunHelp("c:\myhelp.hlp",0)
```

帮助文件 c:\myhelp.hlp被运行，展示主题 0 。

6-3-8 SetLanguage

描述

改变显示中文本的语言。这将重新加载程序文件夹（带有 .LNG扩展名）中的系统语言文件，以及应用程序文件夹中的用户定义文本（带有.USL扩展名）。此函数与用户双击并更改“语言设置...”选项的效果相同。

语法

```
SetLanguage("language name")
```

备注

参数	类型	描述
language name	字符	要设定的语言名称，必须与带有".lng"扩展名的相关文件的文件名称相同。标准选项是English, Czech, Danish, Deutsch, Español, Finnish, French, Italiano, Nederlands (België), Norwegian, Português, Slovenija 和 Swedish。此外， "Default"将加载设计者的默认语言。

例如

```
SetLanguage("Español")
```

在此例中，将加载西班牙语文件。

```
SetLanguage("Default")
```

在此例中，语言将会恢复由应用设计者指定的默认状态。I

6-3-9 GetPerformanceInfo

描述

读出性能监视器和诊断对话框显示的性能和诊断属性的值。

语法

```
returnvalue = GetPerformanceInfo(PLC, Point, "Property Name")
```

备注

参数	类型	描述
returnstate	布尔型	若函数成立则返回状态为1，否则为0。
PLC	字符串	如果被指定，则为获得属性的PLC的名称。如果属性不是 PLC属性，则指定空字符串 "".
Point	字符串	如果被指定，则为获得属性的点的名称。如果不是点属性，则指定空字符串"".

参数	类型	描述
Property Name	字符串	显示的属性名称. 必须和显示的属性名称一致，如果PLC和point都是空字符串，那么属性将会显示“Summary”。

例如:

```
GetPerformanceInfo("", "", "Performance Index")
```

在这个例子里，将显示 Summary Performance Index.

```
GetPerformanceInfo("", "", "Processing Time (ms)")
```

I在这个例子里， 将显示CPU Time 的运行时间。

```
GetPerformanceInfo("MyPLC", "", "Actual CPS")
```

在这个例子里， 将显示 'MyPLC'每秒所运行的实际字符 。

```
GetPerformanceInfo("", "MyPoint", "Read Callbacks")
```

在这个例子里, 将显示point 'MyPoint'的回调内容。

6-3-10 ShutDown

描述

关闭 CX-Supervisor 应用

语法

```
returnstate = ShutDown()
```

备注

参数	类型	描述
returnstate	布尔型	如果函数正确，返回指令为 '1' ，否则返回指令为’0’。

例如:

```
ShutDown()
```

CX-Supervisor运行时操作被终止。

6-4 通信命令

6-4-1 CloseComponent

语法

```
Returnstate = CloseComponent(ComponentName, PLCName)
```

备注

参数	类型	描述
returnstate	布尔型	如果函数正确，返回指令为 '1' ，否则返回指令为’0’。
ComponentName	文本	包含所要关闭的组件名称的文本点或文本常数。
PLCName	文本	包含所要关闭的PLC附带的组件的名称的文本点或文本常数

例如:

```
CloseComponent("PLC Data Monitor", "MyPLC")
```

在这个例子里， 监控PLC ’MyPLC’ 的PLC Data Monitor组件被关闭。

```
Component = "Performance Monitor"
PLC = "PLC06"
OK = CloseComponent(Component, PLC)
```

在这个例子里，监控PLC ’PLC06’的Performance Monitor组件被关闭。’OK’用于确认该操作是否成功进行。

6-4-2 EnableOLE

语法

```
returnstate = EnableOLE(pointname)
```

备注

参数	类型	描述
returnstate	布尔型	Returnstate is '1' if the function is successful, or '0' otherwise.
Pointname	布尔点	A Boolean point that holds the required enable/ disable state.

例如

```
EnableOLE(result)
```

依据点‘result’的数值，OLE函数被激活。如果结果是‘TRUE’，那么OLE被启用，如果结果是‘FALSE’，那么OLE被禁用。

```
EnableOLE(TRUE)
```

在没有使用包含明确数值的点的情况下，OLE函数也能被直接激活。

6-4-3 EnablePLC

语法

```
returnstate = EnablePLC(pointname)
```

备注

参数	类型	描述
returnstate	布尔型	如果函数正确，返回指令为 '1' ，否则返回指令为’0’.
Pointname	布尔点	含有所需启用/禁用状态的布尔点

例如

```
EnablePLC(result)
```

依据point‘result’的数值，PLC函数被激活。如果result是‘TRUE’，那么PLC被激活，如果result是‘FALSE’，那么PLC无效。

```
EnablePLC(TRUE)
```

在没有使用包含明确数值的point的情况下，OLE函数也能被直接激活。

6-4-4 OpenComponent

语法

```
Returnstate = OpenComponent(ComponentName, PLCName)
```

备注

参数	类型	描述
returnstate	布尔型	如果函数正确，返回指令为 '1' ，否则返回指令为'0' 。
ComponentName	文本	包含所要打开的组件名称的文本 point 或文本常数。
PLCName	文本	包含所要关闭的PLC附带的组件的名称的文本point或文本常数

例如:

```
OpenComponent("PLC Data Monitor", "MyPLC")
```

在这个例子中，监控PLC 'MyPLC'的PLC Data Monitor组件被打开。

```
Component = "Performance Monitor"
PLC = "PLC06"
OK = OpenComponent(Component, PLC)
```

在这个例子里，Performance Monitor 控制 PLC 'PLC06' 的打开。 'OK' 用于确认该操作是否成功进行。

6-5 点命令

6-5-1 CancelForce

语法

```
returnstate = CancelForce(pointname)
```

备注

参数	类型	描述
returnstate	布尔型	若函数成立则返回状态为1，否则为0。
pointname	点	Name of point. If the point is an array point then all elements within the array have the CancelForce command applied.

例如

```
CancelForce(point1)
```

赋予点 'point1' 的数值上的Force将被取消。

引用

参考PLC操作手册获取Force Set和Force Reset的详细描述。

6-5-2 CopyArray

语法

```
CopyArray (SourceArray, DestArray)
```

备注

参数	类型	描述
SourceArray		需要复制的源数组point的名称。
DestArray		所复制的数组point的名称。

例如:

```
InitArray (DestArray, 0)
```

首先初始化 'DestArray'.

```
SourceArray [0] = 1
SourceArray [1] = 2
SourceArray [2] = 3
```

然后，初始化 'SourceArray' 为 {1, 2, 3}.

```
CopyArray (SourceArray, DestArray)
```

最后，复制源数组'SourceArray'的内容至目标数组'DestArray'。
这两个数组的规格并没有必要相同，比如，如果'DestArray'包括20个元素，只有元素[0], [1] 和 [2]一一对应数值1,2,3，其他的元素都未被改变。也就是说，如果'DestArray'比'SourceArray'小，比如'DestArray'只包含两个元素，那么元素[0]和[1]将一一对应数值1和2。

注意: 'CopyArray' 允许不同类型的数组，例如，布尔数组可以被复制成实数数组。唯一的限制是文本数组不可以被复制成数值数组，反之亦然。

6-5-3 DisableGroup

语法

```
returnstate = DisableGroup(groupname)
```

备注

参数	类型	描述
returnstate	布尔型	若函数成立则返回状态为1，否则为0。
groupname	文本	Name of the group containing the points to disable.

例如:

```
DisableGroup("<Default>")
```

所有属于 <Default> 组的点都是无效的，因此阻止了数值的读/写。

6-5-4 DisablePoint

语法

```
returnstate = DisablePoint(pointname)
```

备注

参数	类型	描述
returnstate	布尔型	如果函数正确，返回指令为 '1' ，否则返回指令为'0' 。
Pointname	点	将被禁用的点的名称

例如

```
DisablePoint(point1)
```

点'point1'被禁用，因此阻止值被读写。

注意: 这对通信的最优化非常有用。.

6-5-5 EditPoint

语法

```
EditPoint(BoolPoint, Caption, OffText, OnText)
```

或

```
EditPoint(AnalogPoint, Caption, MinValue, MaxValue, Keyboard)
```

或

```
EditPoint(TextPoint, EchoOff, Keyboard)
```

备注

参数	类型	描述
BoolPoint	点	将要编辑的布尔point的名称
Caption	文本	编辑指令的文本说明
OffText	文本	布尔状态0的文本描述
OnText	文本	布尔状态1的文本描述
AnalogPoint	点	将要编辑的整数或实数的名称
MinValue	整数/实数	输入的最小数值
MaxValue	整数/实数	输入的最大数值
Keyboard	布尔型	在屏幕键盘显示TRUE
TextPoint	点	将要编辑的文本点的名称
EchoOff	布尔型	为保障安全，如果输入没有被重复将显示TRUE

例如:

```
EditPoint(bFlag, "Select ON or OFF", "ON", "OFF")
```

对话框显示编辑布尔点'bFlag'至'ON'或'OFF'，并附有说明"Select ON or OFF"。

```
EditPoint(nValue, "Enter a new value", 0.000000, 9999.000000, FALSE )
```

对话框显示编辑模拟点 'nValue'的数值在0-9999之间，并附有不使用屏幕键盘的说明 "Enter a new value"。

```
EditPoint(txtMessage, "Set Text to", FALSE ,FALSE )
```

对话框显示编辑文本点'txtMessage'，并附有说明"Set Text to"重复输入，并不显示屏幕键盘。

6-5-6 EnableGroup

语法

```
returnstate = EnableGroup(groupname)
```

备注

参数	类型	描述
returnstate	布尔型	如果函数正确，返回指令为 '1' ，否则返回指令为'0'。
groupname	文本	包含有效points的组名

例如

```
EnableGroup("<Default>")
```

所有属于 '<Default>' 组的数值都是有效的，因此能够允许数值的读/写。

6-5-7 EnablePoint

语法

```
returnstate = EnablePoint(pointname)
```

备注

参数	类型	描述
returnstate	布尔型	Returnstate is '1' if the function is successful, or '0' otherwise.
pointname	文本	Name of point to be enabled.

例如

```
EnablePoint(point1)
```

点 'point1'被启用，因此允许值被读写。

6-5-8 Force

语法

```
returnstate = Force(pointname)
```

备注

参数	类型	描述
returnstate	布尔型	若函数成立则返回状态为1，否则为0。
pointname	点	Name of point to have force state applied. If the point is an array point then all elements within the array have the Force command applied.

例如

Force(point1)

点'point1' 在目前的语句中被锁定，也就是说，如果当前的数值被设定为1，那么除非CancelForce命令移除锁定指令，否则‘point1’的数值无法更改。

6-5-9 ForceReset

语法

```
returnstate = ForceReset(pointname)
```

备注

参数	类型	描述
returnstate	布尔型	如果函数正确，返回指令为 '1' ，否则返回指令为’0’。
pointname	点	点的名称。如果点是一个数组点，那么ForceReset命令适用于数组中的所有元素。

例如

```
ForceReset(point1)
```

布尔点 'point1'的值被设定为 'FALSE'.
引用
参考PLC操作手册获取ForceSet和ForceReset的详细信息。

6-5-10 ForceSet

语法

returnstate = ForceSet(pointname)

备注

参数	类型	描述
returnstate	布尔型	如果函数正确，返回指令为 '1' ，否则返回指令为'0'。
pointname	点	点的名称。如果点是一个数组点，那么ForceReset命令适用于数组中的所有元素。

例如

```
ForceSet(point1)
```

布尔点 'point1' 的数值被设定为 'FALSE'。
引用
参考PLC操作手册获取ForceSet和ForceReset的详细信息。

6-5-11 GetBit

语法

```
returnpoint = GetBit(pointname,bit)
```

备注

参数	类型	描述
pointname	实数/整数	这是获取位数值的点的名称。将使用点的数值和间接运算。
bit	整数	这将明确哪个位将获取数值
returnpoint	布尔型	包括返回数值 'TRUE' 或'FALSE'。

例如

```
    pointname = 256;
returnpoint = GetBit(pointname,8)
```

点 'returnpoint' 包含语句 'TRUE'.

6-5-12 InitialiseArray

语法

InitArray (arrayname, value)

备注

参数	类型	描述
arrayname		点阵列的名称
value		赋予数组中所有元素的数值

例如:

```
InitArray (MyArray, 0)
```

在这个例子中,，数组'MyArray'中的所有元素均被设定为0。

6-5-13 InputPoint

语法

```
returnstate = InputPoint(pointname, returnflag)
```

备注

参数	类型	描述
returnstate	布尔型	如果函数正确，返回指令为 '1' ，否则返回指令为'0' 。
pointname	点	被读取数据的点的名称。
returnflag	点	当数值从 PLC返回时，被设定为.'TRUE'的可选择的布尔点。

例如:

```
InputPoint(point)
returnflag = FALSE
InputPoint(point, returnflag)
```

请求读取当前点 'point' 的数值。在第二个例子中，当数值从PLC返回时，返回标志被设定为'TRUE'。

注意: 数值不会立刻返回 - 在InputPoint命令脚本下使用返回的数值是不可能的。相反，数值将由"On Condition"脚本获取，且脚本上将会显示表达式'returnflag = TRUE'。

6-5-14 OutputPoint

语法

```
returnstate = OutputPoint(pointname)
```

备注

参数	类型	描述
returnstate	布尔型	如果函数正确，返回指令为 '1' ，否则返回指令为'0' 。
pointname	点	被更新的点。

例如

```
OutputPoint(result)
```

点 'result' 将更新当前的数值。

注意: 如果点处于"forced"指令中，与PLC关联的点的数值将不会被改变。

6-5-15 PointExists

语法

```
returnpoint = PointExists(pointname)
```

备注

参数	参数	描述
pointname	字符串	pointname字符串。文本包括 点的名称。
returnpoint	点	包括返回值的布尔点。

例如:

```
PointName="Testpoint"
Exists=PointExists(PointName)
```

如果有叫做'TestPoint'的点存在，那么布尔点 'Exists'被设定为 'TRUE'。

Note: "PointName" 是一个能够被设定为任何字符串数值的文本点。

6-5-16 SetBit

语法

```
returnstate = SetBit(pointname,bit,value)
```

备注

参数	类型	描述
returnstate	布尔型	如果函数正确，返回指令为 '1' ，否则返回指令为'0'。
pointname	整数/实数	这是获取位的点的名称。将使用点的数值和间接运算。
bit	点	这明确了所要设定的位
value	布尔型	这明确了所要设定位的点的值

例如:

```
testpoint = 0;
SetBit(testpoint,4,TRUE)
```

点'testpoint' 包含值 16。

6-6 PLC 命令

6-6-1 ClosePLC

语法

```
returnstate = ClosePLC("plcname")
```

备注

参数	类型	描述
returnstate	布尔型	如果函数正确，返回指令为 '1' ，否则返回指令为'0'。
plcname	字符串	将要打开的 PLC 的名字. 如果PLC是通过通信组件获取的，例如Omron CX-Communications Control，这个参数控件名称和PLC名称之间应该用一个点分隔，比如 "OMRONCXCommunicationsControl.controlPLC".

例如:

```
ClosePLC("controlPLC")
```

名为controlPLC的PLC被关闭了。在它被再次打开之前，不会再有其他的通信产生。

6-6-2 DownloadPLCProgram

语法

```
returnstate = DownloadPLCProgram(plcname, filename, processed)
```

备注

参数	类型	描述
returnstate	布尔型	如果函数正确，返回指令为 '1' ，否则返回指令为'0'。
plcname	字符串	下载的PLC程序的名称
filename	字符串	磁盘中下载至PLC的文件名。如果驱动和路径不明确，当前的目录是假定的，可能和实际的目录不一样。如果文件名明确为“”，那么用户在使用时创建了一个文件名。
processed	布尔型	当运行完成时，处理完成被设定为显示标志“TRUE”。

例如

```
DownloadPLCProgram("controlPLC", "Prog01.bin", done)
```

T程序储存在当前下载至PLC的目录'controlPLC'下的文件 'Prog01.bin'中。 在继续运行之前，脚本将等待之多五秒钟以供完成运行。

注意: 当语句被执行时，程序的运行并不会立刻完成。处理完成的标志’done‘设定为程序完成一段时间之后出现。因此，使用上传完成并建立一个包含上传完成后执行On Condition 脚本的代码和处理完成标志作为表达式（例: ‘done’）的语句。

注意: 当PLC是‘STOP’模式时，这个命令才能被执行。参照第六章GetPLCMode或SetPLCMode获取更多信息。

6-6-3 GetPLCMode

语法

```
mode = GetPLCMode("plcname")
```

备注

参数	类型	描述
mode	字符串	包含当前PLC模式的文本点 。模式可能为 'STOP', 'DEBUG', 'RUN','MONITOR' 和 'UNKNOWN'。
plcname	字符串	PLC的名称

例如:

```
currentmode = GetPLCMode("controlPLC")
```

在这个例子里，当前PLC'controlPLC'的模式储存在点'currentmode'中。

6-6-4 OpenPLC

语法

```
Returnstate = OpenPLC("plcname", processed)
```

备注

参数	类型	描述
returnstate	布尔型	如果函数正确，返回指令为 '1' ，否则为'0' 。
plcname	字符串	即被打开的PLC的名称。如果PLC是通过通信组件获取的，例如Omron CX-Communications Control，这个参数控件名称和PLC名称之间应该用一个点分隔，比如 "OMRONCXCommunicationsControl.controlPLC".
processed	布尔型	当运行完成时，处理完成被设定为显示标志“TRUE” 。

例如

```
OpenPLC("controlPLC", doneopen)
```

名叫controlPLC 的PLC被打开以供通信。

注意:

当语句被执行时，程序的运行并不会立刻完成。处理完成的标志’done‘设定为程序完成一段时间之后出现。因此，使用上传完成并建立一个包含上传完成后执行On Condition 脚本的代码和处理完成标志“processed”作为表达式的语句（这通常更为有效）。

6-6-5 PLCCommsFailed

语法

```
returnstate = PLCCommsFailed("plcname")
```

备注

参数	语法	描述
returnstate	布尔型	如果函数正确，返回指令为 '1' ，否则为'0'。.
plcname	字符串	被核查的PLC的名称。

例如:

```
IsFailing = PLCCommsFailed ("controlPLC")
```

当名叫controlPLC的PLC不通信的时候，点 IsFailing 被设定为正确。否则，它被设定为错误。

注意:当被命名的PLC通信超时错误出现时，这个函数循环执行，知道PLC进行正常的通信。

6-6-6 PLCMonitor

语法

```
returnstate = PLCMonitor("plcname")
```

备注

参数	类型	描述
returnstate	布尔型	如果函数正确，返回指令为 '1' ，否则为'0' 。

参数	类型	描述
plcname	字符串	要监视的PLC的名称

例如

```
PLCMonitor("controlPLC")
```

PLC的监视对话框已经启用，名为“controlPLC”。该对话框可以用于检查PLC状态，更改模式等。

6-6-7 SetPLCMode

语法

```
returnstate = SetPLCMode("plcname", mode, processed)
```

备注

参数	类型	描述
returnstate	布尔型	当函数成立时为1，否则为0
plcname	字符串	PLC的名称
mode	字符串	新PLC模式的值。有效模式为‘停止’、‘调试’、‘运行’和‘监视’。
processed	布尔型	当操作已经实际完成时，处理参数被设为'TRUE’

例如

```
SetPLCMode("controlPLC", "STOP", done)
```

在此示例中，叫做'controlPLC'的PLC模式被更改为 "STOP".

注意: 执行该语句后，模式可能不会被立即更改。 操作完成后，处理标志将随后设置为 'done' 。 因此，如果使用要求完成操作的语句创建On Condition 脚本，脚本中包含模式设定后所要执行的代码，将以处理标志作为表达式 (如'done')。

6-6-8 SetPLCPhoneNumber

语法

```
Returnstate = SetPLCPhoneNumber("plcname", numbertext)
```

备注

参数	类型	描述
returnstate	布尔型	当函数成立时为1，否则为0
plcname	字符串	要更改号码的PLC的名称
numbertext	字符串	PLC的新电话号码

例如

```
SetPLCPhoneNumber("controlPLC", "01234 987654")
```

PLC的电话号码被更改为所需值。

6-6-9 UploadPLCProgram

语法

```
returnstate = UploadPLCProgram(plcname, filename,
processed)
```

备注

参数	类型	Descripti描述on
returnstate	布尔型	当函数成立时为1，否则为0
plcname	字符串	上传程序所来自PLC的名称
filename	字符串	磁盘上要上传程序至的文件名。若驱动和路径未指定，则文件会创建在当前目录下，可能与应用程序目录不同。若文件名指定为“”，则运行时会提醒用户。
processed	布尔型	当操作已经实际完成时，处理参数被设为 'TRUE'

例如

```
UploadPLCProgram("controlPLC", "Prog01.bin", done)
```

在PLC的程序中， ‘controlPLC’被上传到当前目录下的文件‘Prog01’中。在继续操作前，脚本需等待最多5秒以确认动作成功。

注意: 执行该语句后，操作可能不会立即完成。操作完成后，处理标志将随后设置为‘done'。 因此，如果使用需要已完成操作的语句创建了条件脚本，脚本中包含了上传完成后索要执行的代码，将以处理标志作为表达式（如'done'）。

注意: 该指令只能用于PLC处于'stop'模式下。 参考第六部分的 GetPLCMode 或 SetPLCMode 以获取更多信息。

6-7 温度控制器命令

6-7-1 TCAutoTune

语法

```
returnstate = TCAutoTune(TController,mode)
```

备注

参数	类型	描述
returnstate	布尔型	当函数成立时为1，否则为0
TController	字符串	代表温度控制器名称的字符串
mode	点	该点在TC自动调谐指令发出时，用以说明操作模式，以及定义要执行的操作。 0: 指出自动调谐操作将被停止。 1: 该模式由E5*K支持，并用于将操作极限周值的操纵量变化范围设置为40%。 2: 用于启动自动调谐操作。

例如

```
temp1 = TCAutoTune("e5ak",temp2)
```

6-7-2 TCBackupMode

语法

```
returnstate = TCBackupMode(TController,mode)
```

备注

参数	类型	描述
returnstate	布尔型	当函数成立时为1，否则为0
TController	字符串	代表温度控制器名称的字符串
mode	点	该点用以说明操作模式，以及定义温度控制器存储内部变量的方法。 0: 在该模式下变量被存储在RAM和EPROM中。 1: 在该模式下变量只被存储在RAM中。

例如

```
temp1 = TCBackupMode("ea5k",temp2)
```

6-7-3 TCGetStatusParameter

语法

```
returnstate = TCGetStatusParameter(TController,paramID,value)
```

备注

参数	类型	描述
returnstate	布尔型	当函数成立时为1，否则为0.
TController	字符串	代表温度控制器名称的字符串

参数	类型	描述
paramID	点	该点用以说明所需参数，范围为0-22 0: ControlMode. 1: Output. 2: InputShiftDelay (Bool) E5*F, E5*X, E5*J. 3: DisplayUnit. 4: PIDConstantDisplay (Bool) E5*F, E5*X, E5*J. 5: OutputType. 6: CoolingType. 7: Output2. 8: Alarm1. 9: Alarm2. 10: InputType (Integer) E5*F, E5*X, E5*J. 11: OperationMode. 12: BackupMode. 13: AutoTuneMode. 14: OverFlow (Bool) E5*F, E5*X, E5*J. 15: UnderFlow (Bool) E5*F, E5*X, E5*J. 16: SensorMalfunction (Bool) E5*F, E5*X, E5*J. 17: ADConvertorFailure (Bool) E5*F, E5*X, E5*J. 18: RAMAbnormality (Bool) E5*F, E5*X, E5*J. 19: RAMMismatch (Bool) E5*F, E5*X, E5*J. 20: StatusWordsOnly (Bool) E5*K only (TRUE indicates valid words below). 21: Status0 (word) E5*K only. 22: Status1 (word) E5*K only.
value	实数点 或整数点	返回状态参数值。详细信息参考上述参数ID。

例如

```
temp1 = TcGetStatusParameter("e5ak",temp2,temp3)
```

6-7-4 TCRemoteLocal

语法

```
returnstate = TCRemoteLocal(TController,mode)
```

备注

参数	类型	描述
returnstate	布尔型	当函数成立时为1，否则为0
TController	字符串	代表温度控制器名称的字符串.

参数	类型	描述
mode	点	该点用以说明操作的模式，以及定义温度控制器的运作模式。 0: 指代温度控制器在远程模式中。 1: 指代温度控制器在本地模式中。

例如

```
temp1 = TCRemoteLocal("e5ak",temp2)
```

注意: 该指令以前被称为 TCOperationalMode.

6-7-5 TCRequestStatus

语法

```
returnstate = TCRequestStatus(Tcontroller, returnflag)
```

备注

参数	类型	描述
returnstate	布尔型	若函数成立则返回状态为1，否则为0。
TController	字符串	代表温度控制器名称的字符串
returnflag	点	该点用以说明返回已经返回并且可供TCGetStatusParameter使用。

例如

```
temp1 = TCRequestStatus("e5ak", temp2)
```

Note: 状态信息不会立即返回——因为不能在与指令在同一脚本下访问状态信息。因此，状态信息应该在具有表达式 "returnflag == TRUE" 的"On Condition" 脚本中被访问。

6-7-6 TCRspLsp

语法

```
returnstate = TCRspLsp(Tcontroller,mode)
```

备注 –

参数	类型	描述
returnstate	布尔型	当函数成立时为1，否则为0
TController	字符串	代表温度控制器名称的字符串
mode	点	该点说明了温度控制器所用的操作模式，并定义了定位点。 0: 指代远程定位点模式 1: 指代本地定位点模式

例如

```
temp1 = TCRspLsp("e5ak",temp2)
```

Note: 该命令之前被叫作TCSetpoint。

6-7-7 TCRunStop

语法

```
returnstate = TCRunStop(TController,mode)
```

参数	类型	描述
returnstate	布尔型	若函数成立则返回状态为1，否则为0
TController	字符串	代表温度控制器名称的字符串
mode	点	该点用以说明操作的模式，以及定义自动输出模式切换或手动输出模式切换。 0: 指代手动输出模式切换 1: 指代自动处处模式切换

例如

```
temp1 = TCRunStop("e5ak",temp2)
```

注意: 该命令之前被叫作TCModeShift

6-7-8 TCSaveData

语法

```
returnstate = TCSaveData(TController)
```

备注

参数	类型	描述
returnstate	布尔型	若函数成立则返回状态为1，否则为0
TController	字符串	代表温度控制器名称的字符串

例如

```
temp1 = TCSaveData("e5ak",temp2)
```

6-7-9 TCSettingLevel1

语法

```
returnstate = TCSettingLevel1(TController)
```

备注

参数	类型	描述
returnstate	布尔型	若函数成立则返回状态为1，否则为0
TController	字符串	代表温度控制器名称的字符串

例如

```
temp1 = TCSettingLevel1("e5ak")
```

6-7-10 TCReset

语法

```
returnstate = TCReset(TController)
```

备注

参数	类型	描述
returnstate	布尔型	若函数成立则返回状态为1，否则为0.
TController	字符串	代表温度控制器名称的字符串

例如

```
temp1 = TCReset("e5ak")
```

6-8 警报命令

6-8-1 AcknowledgeAlarm

语法

```
returnstate = AcknowledgeAlarm("alarmname")
```

备注

参数	类型	描述
returnstate	布尔型	若函数成立则返回状态为1，否则为0
alarmname	字符串	警报的标识

例如

```
AcknowledgeAlarm("temphigh")
```

警报'temphigh' 确认。

参考:

参考CX-Supervisor用户手册获取更多关于警报的详细信息。

6-8-2 AcknowledgeAllAlarms

语法

```
returnstate = AcknowledgeAllAlarms()
```

备注

参数	类型	描述
returnstate	布尔型	若函数成立则返回状态为1，否则为0。

例如

```
AcknowledgeAllAlarms()
```

所有警报确认。

参考:

参考CX-Supervisor用户手册获取更多关于警报的详细信息。

6-8-3 AcknowledgeLatestAlarm

语法

```
returnstate = AcknowledgeLatestAlarm()
```

备注

参数	类型	描述
returnstate	布尔型	若函数成立则返回状态为1，否则为0

例如

```
AcknowledgeLatestAlarm()
```

最新警报的确认具有最高优先权.

参考

参考CX-Supervisor用户手册获取更多关于警报的详细信息。

6-8-4 ClearAlarmHistory

语法

```
returnstate = ClearAlarmHistory()
```

备注

参数	类型	描述
returnstate	布尔型	若函数成立则返回状态为1，否则为0.

例如

```
ClearAlarmHistory()
```

该警报历史窗口和记录被清除。

参考

参考CX-Supervisor用户手册获取更多关于警报的详细信息。

6-8-5 CloseAlarmHistory

语法

```
returnstate = CloseAlarmHistory()
```

备注

参数	类型	描述
returnstate	布尔型	若函数成立则返回状态为1，否则为0

例如

```
CloseAlarmHistory()
```

该警报历史窗口被关闭。

参考:

参考CX-Supervisor用户手册获取更多关于警报的详细信息。

6-8-6 CloseAlarmStatus

语法

```
returnstate = CloseAlarmStatus()
```

备注

参数	类型	描述
returnstate	布尔型	若函数成立则返回状态为1，否则为0。

例如

```
CloseAlarmStatus()
```

当前警报窗口被关闭。

参考:

参考CX-Supervisor用户手册获取更多关于警报的详细信息。

6-8-7 DisplayAlarmHistory

语法

```
returnstate = DisplayAlarmHistory()
```

备注

参数	类型	描述
returnstate	布尔型	若函数成立则返回状态为1，否则为0

例如

```
DisplayAlarmHistory()
```

该警报历史窗口已显示。

参考:

参考CX-Supervisor用户手册获取更多关于警报的详细信息。

6-8-8 DisplayAlarmStatus

语法

```
returnstate = DisplayAlarmStatus()
```

备注

参数	类型	描述
returnstate	布尔型	若函数成立则返回状态为1，否则为0

例如

```
DisplayAlarmStatus()
```

当前警报状态已显示。

参考:

参考CX-Supervisor用户手册获取更多关于警报的详细信息。

6-8-9 EnableAlarms

语法

```
EnableAlarms (flag, "message")
```

备注

参数	类型	描述
flag		若标志设置为‘TRUE’，则警报记录已启用，若设置为'FLASE'，则记录停用。
message		记录在警报中的文本信息指示出状态的更改。

例如

```
EnableAlarms (TRUE, "Alarm logging enabled")
```

参考:

参考CX-Supervisor用户手册获取更多关于警报的详细信息。

6-8-10 IsAlarmAcknowledged

语法

```
pointname = IsAlarmAcknowledged("alarmname")
```

备注

参数	类型	描述
pointname	布尔型点	要在已确认警报的测试上赋值的布尔型点的名称。
alarmname	字符串	警报的标识。

例如

```
acknowledged = IsAlarmAcknowledged("temptoohigh")
```

若'temptoohigh'警报当下已确认，则该点'acknowledged'被赋值为布尔型状态"TRUE"。若该警报当下未确认，则赋值为"FLASE"。

参考

参考CX-Supervisor用户手册获取更多关于警报的详细信息。

6-8-11 IsAlarmActive

语法

```
pointname = IsAlarmActive("alarmname")
```

备注

参数	类型	描述
pointname	布尔型点	要在已确认警报的测试上赋值的布尔型点的名称。
alarmname	字符串	警报的标识。

例如

```
active = IsAlarmActive("temptoohigh")
```

若'temptoohigh'警报当下活跃，则该点'active'被赋值为布尔型状态 "TRUE"。若该警报当下不活跃，则赋值为"FLASE"。

参考

参考CX-Supervisor用户手册获取更多关于警报的详细信息。

6-9 File Commands

6-9-1 CloseFile

语法

```
returnstate = CloseFile(pointname)
```

备注

参数	类型	描述
returnstate	布尔型	若函数成立则返回状态为1，否则为0
pointname	布尔型	包含文件所需状态——当文件关闭时是否应去除空格——的布尔型点。

例如

```
CloseFile(status)
```

当前打开文件被关闭。若布尔型点"status"设置为"TRUE"，则从文件中跳过每行结尾处空格。

```
CloseFile(FALSE)
```

注意: 若从文件中跳过空格，则会大大减小文件大小，但需要较长时间关闭文件。若空格在网络驱动一次用在多个系统中，则不能从文件中跳过。

在该示例中，当前打开文件被关闭，且文件中未跳过任何空格。

6-9-2 CopyFile

语法

```
returnstate = CopyFile("sourcename", "destname")
```

备注

参数	类型	描述
returnstate	布尔型	若函数成立则返回状态为1，否则为0。
sourcename	字符串	要复制的文件的路径名称。可能包含“*”通配符。
destname	字符串	要复制文件到的目标路径名称。若路径名称不存在，则将自动创建。

例如

```
CopyFile("c:\autoexec.bat", "c:\autoexec.old")
```

文件"c:\autoexec.bat" 被复制到文件"c:\autoexec.old".

```
CopyFile("c:\logging\*.dlv", "a:\backup")
```

在“C:\logging”中的数据记录文件（以dlv结尾）被复制到驱动A下的目录“\backup”中。

6-9-3 DeleteFile

语法

returnstate = DeleteFile("filename")

备注

参数	类型	描述
returnstate	布尔型	若函数成立则返回状态为1，否则为0。
Filename	字符串	要删除文件的路径名称。

例如

```
DeleteFile("c:\pagename.pag")
```

文件“c:\pagename.pag”被删除。

6-9-4 EditFile

语法

```
returnstate = EditFile("filename")
```

备注

参数	类型	描述
returnstate	布尔型	若函数成立则返回状态为1，否则为0。
Filename	字符串	要编辑文件的路径名称

例如

```
EditFile("C:\report3.txt")
```

```
FileExists
```

语法

```
returnpoint = FileExists (filename)
```

备注

参数	类型	描述
filename	字符串	文本字符串包含文件名。
returnpoint	点	包含返回值的布尔点。

例如

```
FileName = "TEST.TXT"
Exists = FileExists(FileName)
```

如果一个名为'C:\TEST.TXT' 的文件存在，布尔点'Exists'则被设定为 'TRUE' 。

注意: "FileName" 是一个文本型点，它能被设定为任何字符串值。

6-9-5 MoveFile

语法

```
returnstate = MoveFile("sourcename", "destname")
```

备注

参数	类型	描述
returnstate	布尔型	若函数成立则返回状态为1，否则为0。
sourcename	字符串	要移动的文件的路径名。
destname	字符串	移动目的地的路径名。

例如

```
MoveFile("c:\autoexec.bat", "c:\autoexec.old")
```

文件 "c:\autoexec.bat" 被移动到文件 "c:\autoexec.old"中。

6-9-6 OpenFile

语法

```
returnstate = OpenFile("filename")
```

备注

参数	类型	描述
returnstate	布尔型	若函数成立则返回状态为1，否则为0。
Filename	字符串	要打开的文件的路径名。

例如

```
OpenFile("c:\filename")
```

文件"c:\filename.csf"已打开，并能通过Read() 和 Write() 脚本命令访问。一次只能打开一个文件。如果文件不存在，则创建该文件。文件可以分享（例如位于网络驱动器中，并能同时被几个正在运行的CX-Supervisor应用程序访问——这可以被应用于数据交换）。

注意: 扩展名".csf" 一直加在文件名中，因此它不能被指定为参数的一部分。

6-9-7 PrintFile

语法

```
returnstate = PrintFile("filename")
```

备注

参数	类型	描述
returnstate	布尔型	若函数成立则返回状态为1，否则为0。
Filename	字符串	要打印的文件的路径名。

例如

```
PrintFile("c:\autoexec.bat")
```

文件"c:\autoexec.bat" 被送往当前配置的打印机。
具有文本参数的脚本命令能使用引号内的文字字符串或文本点。

Note: CX-Supervisor使用OLE的注册信息（文件扩展关联）来决定如何打印一个文件。它调用了与一个指定文件扩展名有关的父程序，命令应用程序开始最小化，并传递“打印” 命令。例如，如果文件扩展名.txt与Notepad关联，那么Notepad就被调用来打印文件。

6-9-8 Read

语法

```
returnstate = Read(RecordId, pointname, ...)
```

备注

参数	类型	描述
returnstate	布尔型	若函数成立则返回状态为1，否则为0。
RecordId	整数型	进入文件的索引。.
Pointname	点	点的名称要用从打开的文件中读取的数据来更新。

例如

```
Read(1, value)
```

点“value”被赋有从当前打开文件中读取的值，文件使用值为1的索引文件。

```
ReadOK = Read(indexno, value1, value2, value3)
```

点‘value1’, ‘value2’, ‘value’ 使用作为索引文件的索引号值。通过或失败的状态都储存在‘ReadOK’中。

注意: 建议尽可能使用小于1024的RecordId，以优化文件访问（记录0到1023被缓存）。

6-9-9 ReadMessage

语法

```
returnstate = ReadMessage ("filename", offset, textpoint, noofchars)
```

备注

参数	类型	描述
returnstate	布尔型	若函数成立则返回状态为1，否则为0。
Filename	字符串	要读取的文件的路径名。

参数	类型	描述
Offset	整数型	从文件开头（字符中）的偏移说明从哪里开始读取。
Textpoint	文本点	保存从文件中读取字符的文本点。
Noofchars	整数型	从文件中读取的字符数。

例如

```
ReadMessage ("C:\CX-SUPERVISOR\TESTFILE.TXT", 0, TextPoint, 20)
```

前20个字符是从文件"C:\CX-SUPERVISOR\TESTFILE.TXT" 中读取的，并且储存在点‘TextPoint’里。

注意: 文本点能容纳256个字符，因此最多256个字符能从文件中读取出来。

6-9-10 SelectFile

语法

```
filename = SelectFile (filter, path)
```

备注

参数	类型	描述
Filename		返回文本字符串。如果从OpenFile通讯对话框中选择了OK，则包含完全限定的文件名，包括驱动器和路径；否则包含空字符串。
Filter	字符串	可选参数。如果省略，将显示所有文件。如果路径被指定，例如设置为“ ”，则必须提供此参数。指定“文件类型”列表使用的过滤器字符串。字符串应包含1个或多个过滤器，用'|'（通道）字符分开，并以2个字符结束，例如“||”。每个过滤器应有一些用户文本和1个或多个用分号分隔的文件规格。除了在用户文本中，否则不应使用空格。
Path	字符串	可选参数。指定最初显示的路径。如果省略，对话框将显示当前工作目录。

例如

```
TFile = SelectFile()
```

将显示'File Open' 对话框，显示当前工作目录中的所有文件。用户的选择将存储在tFile中。

```
TFile = SelectFile("Text Files (*.txt)|*.txt||")
```

将显示'File Open' 对话框，仅显示当前工作目录中带有.txt扩展名的文件。

```
TFile = SelectFile("Text Files (*.txt;*.csv)|*.txt;*.csv||")
```

将显示 'File Open'对话框，显示当前工作目录中具有.txt或.csv扩展名的文件。

```
TFile = SelectFile("Text Files (*.txt;
*.csv)|*.txt;*.csv|Document Files (*.doc)|*.doc||")
```

在本示例中， 'Files of type' 过滤器有两种选择：一是显示文本文件（例如 .txt 和.csv 文件)，二是显示文档文件（只能是.doc 文件）。

```
TFile = SelectFile("", "C:\WINDOWS")
```

显示'File Open'对话框，展示在"C:\WINDOWS"目录里的所有文件。

6-9-11 Write

语法

```
returnstate = Write(RecordId, pointname, ...)
```

备注

参数	类型	描述
returnstate	布尔型	若函数成立则返回状态为1，否则为0。
RecordId	实数型	进入文件的索引。
Pointname	点	点的名称要用从打开的文件中读取的数据来更新。

例如

```
WroteOK = Write(indexno, $Second)
```

使用作为进入文件的索引的索引号值，将点'$Second'写入到当前打开的文件中。通过或失败状态储存在‘WroteOK’中。

```
Write(2, $Second, $Minute, $Hour)
```

点'$Second'，'$Minute'，'$Hour' 被写入到当前打开的文件中，使用值2作为文件的索引。
注意：建议尽可能使用小于1024的RecordId，以优化文件访问（记录0到1023被缓存）。

6-9-12 WriteMessage

语法

```
returnstate = WriteMessage("filename", offset, "text",
linefeed)
```

备注

参数	类型	描述
returnstate	布尔型	若函数成立则返回状态为1，否则为0。
filename	字符串	要写入的文件的路径名。
offset	整数	从文件开头的偏移（以字符为单位），表示开始写入的位置。如果偏移量为-1，那么消息将附加到文件的结尾。
text	字符串	要写入文件的文本。
linefeed	布尔型	应附加表示回车和换行符的标志。

例如

```
WriteMessage("C:\CX-SUPERVISOR\TESTFILE.TXT",      0,
"Hello World", TRUE)
```

文本‘ Hello World’写在‘C：\ CX-SUPERVISOR \ TESTFILE.TXT’ 文件的开头，并附加回车和换行符，将后面的文本移动到下一行的开头。

注意: 当文本写入文件时，它将覆盖可能存在于此位置的任何现有文本。

6-10 配方命令

6-10-1 DisplayRecipes

语法

```
returnstate = DisplayRecipes()
```

备注

参数	类型	描述
returnstate	布尔型	若函数成立则返回状态为1，否则为0。

例如

```
DisplayRecipes()
```

显示当前配方。

参考

有关配方的详细信息，请参阅CX-Supervisor用户手册。

6-10-2 DownloadRecipe

语法

```
returnstate = DownloadRecipe("recipename")
```

备注

参数	类型	描述
returnstate	布尔型	若函数成立则返回状态为1，否则为0。
recipename	字符串	要下载的配方名称。

例如

```
DownloadRecipe("recipe1")
```

配方‘recipe1’下载完毕。

参考

有关配方的详细信息，请参阅CX-Supervisor用户手册。

6-10-3 UploadRecipe

语法

```
returnstate = UploadRecipe("recipename", processed)
```

备注

参数	类型	描述
returnstate	布尔型	若函数成立则返回状态为1，否则为0。

参数	类型	描述
recipename	字符串	要上传的配方的名称。
processed	布尔型	当操作完成时，标记被设置为正确。

例如

```
UploadRecipe("recipe1",done)
```

配方‘recipe1’被上传，并且当完成上传时，点‘done’ 被设置为True。

参考

有关配方的详细信息，请参阅CX-Supervisor用户手册。

6-11 报告命令

6-11-1 GenerateReport

语法

```
returnstate =
GenerateReport(ReportTemplateFile,ReportOutputFile)
```

备注

参数	类型	描述
returnstate	布尔型	若函数成立则返回状态为1，否则为0。
ReportTemplateFile	字符串	报告模板文件的路径名。
ReportOutputFile	字符串	报告输出文件的路径名。

例如

```
GenerateReport("report3.txt","output.txt")
```

ReportTemplateFile report3.txt 包含预定义的一组点名称和文本，排版完全适应于报告阅读器。包含字符中包含的点名称是报告中所需的数据的CX-Supervisor名称。

可以在项目/运行时间设置/报告设置对话框中更改包含字符，但是一旦设置，必须对项目生成的所有报告进行修改。

模板文件可以使用任何ASCII文本编辑器编写，例如文本文件（.TXT），富文本文件（.RTF）或超文本文件（.HTML）。

处理报告模板，使用当前值动态替换点名称，并另存为output.txt。

6-11-2 PrintReport

语法

```
returnstate = Printreport(ReportTemplateFile)
```

备注

参数	类型	描述
returnstate	布尔型	若函数成立则返回状态为1，否则为0。

参数	类型	描述
ReportTemplateFile	字符串	报告模板文件的路径名。
ReportOutputFile	字符串	报告输出文件的路径名。

例如

```
PrintReport("report3.txt")
```

报告模板被处理，使用当前值动态替换点名称，并打印到默认Windows打印机。

6-11-3 ViewReport

语法

```
returnstate = ViewReport(ReportTemplateFile)
```

备注

参数	类型	描述
returnstate	布尔型	若函数成立则返回状态为1，否则为0。
ReportTemplateFile	字符串	报告模板文件的路径名。

例如

```
ViewReport("report3.txt")
```

6-12 文本命令

6-12-1 BCD

语法

```
result = BCD (value)
```

备注

参数	类型	描述
Value		要转换为二进制编码十进制（BCD）的数字。
result		包含BCD表示法的值的字符串。

例如

```
BCDStr = BCD(39)
```

在本示例中，‘BCDstr’ 包含 ‘00111001’。

6-12-2 Bin

语法

```
result = Bin (value)
```

备注

参数	类型	描述
Value		要转换为二进制数的数字。
result		包含二进制表示法的值的字符串。

例如

```
BStr = Bin (20)
```

在本示例中，‘Bstr’ 包含‘10100’。

6-12-3 Chr

语法

```
result = Chr (value)
```

备注

参数	类型	描述
Value		将扩展ASCII的值转换为一个字符。
result		包含单字符表达式的值的字符串。

例如

```
Char = Chr(65)
```

在本示例中，‘Char’包含‘A’。

6-12-4 FormatText

语法

```
textpoint = FormatText ("formattext", expression, ...)
```

备注

参数	类型	描述
textpoint	文本点	保留格式化文本的文本点。
formattext	字符串	插入结果表达式的文本（带有适当的格式字符）。
expression	整数型/实数型	插入到格式化文本的值或表达式。

例如

```
TextPoint = FormatText ("Boiler temperature is %ld
degrees.", BoilerTemp)
```

‘ BoilerTemp’点的值插入到由格式化字符（%ld）标记的位置的指定文本中，然后存储在点‘ TextPoint’中。
如果’BoilerTemp’的值为57，则存储在‘TextPoint’中的结果文本如下：

```
"Boiler temperature is 57 degrees."
TextPoint = FormatText ("Boiler %ld temperature is %ld
degrees.", BoilerNo, BoilerTemp)
```

‘BoilerNo’点的值插入在第一个‘%ld’标记处，并且‘BoilerTemp’点的值插入在第二个‘ %ld’标记处，结果字符串存储在点‘TextPoint’中。
如果‘BoilerNo’的值为7，‘BoilerTemp’的值为43，则存储在‘TextPoint’中的结果文本如下：

```
"Boiler 7 temperature is 43 degrees."
```

注意: 格式化字符是标准的'C'格式化字符（由C语言sprintf函数使用）。 一些常用的类型是:

- %ld. 插入整数值;
- %f. 插入十进制值。前缀带小数点和数字来控制位置（例如‘%.2f’为2个小数位）;
- %s. 插入字符串;
- %IX. 插入十六进制值（大写的十六进制字符，例如‘FFFF’）;
- %lx. 插入十六进制值（小写的十六进制字符，例如‘ffff’）;
- %c. 插入字符（可用于将值转化为字符，例如插入控制字符）。

使文本向左对齐，并使用宽度字段（例如，'%-6ld'用于插入与6个字符宽的字段向左对齐的值）。

参考

更复杂的表达式（例如控制对齐，小数位数，数字基数等）也是可能的。有关‘sprintf’ 函数使用的格式的完整详细信息，请参考任意C语言参考书。

6-12-5 GetTextLength

语法

```
value = GetTextLength (textpoint)
```

备注

参数	类型	描述
textpoint	文本点	这是经过文本长度计算的点。
returnpoint	整数型/实数型	这是保存返回值的点。

例如

```
textpoint = "Hello World"
count = GetTextLength (textpoint)
```

计算'textpoint'中的字符数，并将点'count'设置为值11。

6-12-6 Hex

语法

```
result = Hex (value)
```

备注

参数	类型	描述
Value		要转换为十六进制数的数字。
Result		包含十六进制表示法的值的字符串。

例如

```
HStr = Hex (44)
```

在本示例中，'Hstr' 包含 '2C'。

6-12-7 Left

语法

```
lefttext = Left(textpoint,noofchars)
```

备注

参数	类型	描述
textpoint	文本	包含要处理的字符串的文本点。
noofchars	整数	从字符串开头提取的字符数。
lefttext	文本	包含指定字符范围的文本点。

例如

```
textpoint = "abcdefgh"
lefttext = Left(textpoint,3)
```

文本点'lefttext'包含字符串'abc'。

6-12-8 Message

语法

```
Message("message")
```

备注

参数	类型	描述
message	字符串	包含消息盒中显示的文本字符串。

例如

```
Message("this is a message")
```

消息'this is a message'在消息框中显示。

6-12-9 Mid

语法

```
midtext = Mid(textpoint,offset,noofchars)
```

备注

参数	类型	描述
textpoint	文本	包含要处理的字符串的文本点。
offset	整数	要包含在提取中的字符串中第一个字符的基于零的索引。
noofchars	整数	从字符串中提取的字符数。
midtext	文本	包含指定字符范围的文本点。

例如

```
textpoint = "abcdefgh"
midtext = Mid(textpoint,3,2)
```

文本点'midtext'包含字符串'de'。

6-12-10 PrintMessage

语法

PrintMessage ("message")

备注

参数	类型	描述
message	字符串	包含发送至打印机的文本字符串。

例如

```
PrintMessage ("Print this message")
```

消息'print this message'被打印至经配置的 'Alarm/message printer'，如果以页面模式操作，进入排队状态，否则打印机被EnablePrinting命令禁用。

参考

参考CX-Supervisor用户手册以获取配置 'Alarm/message printer'的详细信息。

6-12-11 Right

语法

```
righttext = Right(textpoint,noofchars)
```

备注

参数	类型	描述
textpoint	文本	包含即将被控制的字符串的文本点。
noofchars	整数	从字符串提取的字符数。
righttext	文本	包含特定字符范围的文本点

例如

```
textpoint = "abcdefgh"
righttext = Right(textpoint,3)
```

文本点 'righttext'包含字符 'fgh'.

6-12-12 TextToValue

语法

```
valuepoint = TextToValue(textpoint)
```

备注

参数	类型	描述
textpoint	文本	包含即将被转换为数字的字符串的文本点。
valuepoint	整数	包含字符串转换后返回的值的点。

例如

```
textpoint = "10"
valuepoint = TextToValue(textpoint)
```

值10被赋予点 'valuepoint'.

```
textpoint = "10.34"
realpoint = TextToValue(textpoint)
```

实数值10.34被赋予点'realpoint'.

6-12-13 ValueToText

语法

```
textpoint = ValueToText(value)
```

备注

参数	类型	描述
value	整数	被代替为文本点的数字。点名称也是有效参数。
textpoint	文本点	包含转换为字符串的值的文本点。

例如

```
textpoint = ValueToText(10)
```

值10 被输入一个字符串且被赋予文本点 'textpoint'.

```
value = 10
textpoint = ValueToText(value)
```

与前例具相同效果。

6-13 事件/错误命令

6-13-1 ClearErrorLog

语法

```
ClearErrorLog()
```

例如

```
ClearErrorLog()
```

错误列表被清除，记录被删除。

6-13-2 CloseErrorLog

语法

```
returnstate = CloseErrorLog()
```

备注

参数	类型	描述
returnstate	布尔型	若函数成立则返回状态为1，否则为0。

例如

```
CloseErrorLog()
```

所有当前记录的错误的列表被关闭。

6-13-3 DisplayErrorLog

语法

```
returnstate = DisplayErrorLog()
```

备注

参数	类型	描述
returnstate	布尔型	若函数成立则返回状态为1，否则为0。

例如

```
DisplayErrorLog()
```

所有当前记录的错误的列表显示在对话框中。

6-13-4 EnableErrorLogging

语法

```
returnstate = EnableErrorLogging(pointname)
```

备注

参数	类型	描述
returnstate	布尔型	若函数成立则返回状态为1，否则为0。
pointname	布尔型	一个布尔点。

例如

```
EnableErrorLogging(flag)
```

基于布尔点 'flag'，错误日志记录被启用。如果 'flag' 是 'TRUE', 那么错误日志记录被启用；否则被禁用。

6-13-5 LogError

语法

```
returnstate = LogError("message", priority)
```

备注

参数	类型	描述
returnstate	布尔型	若函数成立则返回状态为1，否则为0。
message	字符串	包括显示在错误记录中的文本字符串。
priority	整数	赋予错误的优先权 0 - 低 1 - 中 2 - 高

例如

```
LogError("This is an error", 1)
```

消息 'This is an error' 在错误记录中显示为中优先级错误。

6-13-6 LogEvent

语法

```
returnstate = LogEvent("message")
```

备注

参数	类型	描述
returnstate	布尔型	若函数成立则返回状态为1，否则为0。
message	字符串	包含显示在错误记录中的文本字符串。

例如

```
LogEvent("this is an event")
```

消息 'this is an event'在错误记录中显示为事件 。

6-14 打印机命令

6-14-1 ClearSpoolQueue

语法

```
returnstate = ClearSpoolQueue()
```

备注

参数	类型	描述
returnstate	布尔型	若函数成立则返回状态为1，否则为0。

例如

```
ClearSpoolQueue()
```

排队等候发送至CX-Supervisor Alarm/Message打印机的任何消息（典型的是打印警报）都被舍弃。

6-14-2 EnablePrinting

语法

```
returnstate = EnablePrinting(flag)
```

备注

参数	类型	描述
returnstate	布尔型	若函数成立则返回状态为1，否则为0。
flag	布尔型	0禁用，1 启用。

例如

```
EnablePrinting(FALSE) - Disables printing
EnablePrinting(TRUE) - Enables printing
```

当警报打印被禁用，任何新消息都将储存但不会被打印。当警报打印重新启用，任何未处理的消息将会被打印（如果按行执行模式），或被添加到当前页（如果是页面模式）。

6-14-3 PrintActivePage

语法

```
returnstate = PrintActivePage(flag)
```

备注

参数	类型	描述
returnstate	布尔型	若函数成立则返回状态为1，否则为0。
flag	布尔型	说明在打印前，打印启动对话框是否显示的标志。

例如

```
PrintActivePage(TRUE)
```

当前激活的页面已发送至打印机。标志 'TRUE'说明打印对话框已显示；反之则未显示。

6-14-4 PrintPage

语法

```
returnstate = PrintPage ("pagename", flag, printheaderfooter)
```

备注

参数	类型	描述
returnstate	布尔型	若函数成立则返回状态为1，否则为0。
pagename	字符串	将被打印的页面的名称
flag	布尔型	说明在打印前，打印启动对话框是否显示的标志。
printheaderfooter	布尔型	可选。控制页眉和页脚是否包含打印输出细节的标志。

例如

```
PrintPage("page1", TRUE)
```

CX-Superviso页面被发送至打印机。标志'TRUE'说明打印对话框首先显示，以进行打印机设置；如果指定为 'FALSE' 而非'TRUE' ，那么不会显示打印对话框，页面被直接打印。

6-14-5 PrintScreen

语法

```
returnstate = PrintScreen(flag)
```

备注

参数	类型	描述
returnstate	布尔型	若函数成立则返回状态为1，否则为0。
flag	布尔型	说明在打印前，打印启动对话框是否显示的标志。

例如

```
PrintScreen(FALSE)
```

打印所有当前查看中的CX-Supervisor页面。标志'FALSE'说明打印对话框未显示；标志'TRUE'使打印对话框显示，并允许用户配置或选择打印机。

6-14-6 PrintSpoolQueue

语法

```
returnstate = PrintspoolQueue()
```

备注

参数	类型	描述
returnstate	布尔型	若函数成立则返回状态为1，否则为0。

例如

```
PrintSpoolQueue
```

排队等候发送至CX-Supervisor Alarm/Message打印机的任何消息（典型的是打印警报）都将立刻打印。

6-15 安全命令

6-15-1 Login

语法

```
returnstate = Login(username, password)
```

备注

参数	类型	描述
returnstate	布尔型	若函数成立则返回状态为1，否则为0。
username	文本	用户名登陆的可选参数。如省略，会出现登陆对话框。
password	文本	用户登录密码的可选参数。如使用，必须指定用户名，即便空白，即 ""。如省略，会出现登陆对话框。

例如

```
Login()
```

为用户入口显示登陆对话框。

```
Login("Designer", "Designer")
```

使用匹配密码默认'Designer'用户被自动记录。

参考

参考CX-Supervisor用户手册以获取登陆的更多详细信息。

6-15-2 Logout

语法

```
returnstate = Logout()
```

备注

参数	类型	描述
returnstate	布尔型	若函数成立则返回状态为1，否则为0。

例如

```
Logout()
```

用户注销。

参考

参考CX-Supervisor用户手册以获取注销的更多详细信息。

6-15-3 SetupUsers

语法

```
returnstate = SetupUsers()
```

备注

参数	类型	描述
returnstate	布尔型	若函数成立则返回状态为1，否则为0。

例如

```
SetupUsers()
```

为用户入口显示使用者设定对话框。

参考

参考 CX-Supervisor用户手册以获取设置和修改用户详细信息的更多内容。

6-15-4 ChangeUserPassword

语法

```
ChangeUserPassword("username","old", "new")
```

备注

参数	类型	描述
username	字符串	需要更改密码的用户
old	字符串	现有密码
new	字符串	新密码

例如

```
ChangeUserPassword("Fred Smith","fred1", "fred2")
```

ChangeUserPassword将更改‘Fred Smith’s’ Windows登陆密码‘fred1’ 至‘fred2’.

参考

参考CX-Supervisor用户手册以获取设置和修改用户详细信息的更多内容。

6-16 数据记录命令

6-16-1 AuditPoint

语法

```
AuditPoint("pointname")
```

备注

参数	类型	描述
pointname	字符串	要记录到CFR数据库的点的名称。

例如

```
AuditPoint("MyInteger")
```

此命令将引起 'MyInteger'的值被记录到CFR数据库中。

6-16-2 ClearLogFile

语法

```
ClearLogFile("datasetname")
```

备注

参数	类型	描述
datasetname	字符串	要清除的数据集名称，如同文本点或常量。

例如

```
ClearLogFile("Process 1")
```

此命令将从活跃（最新）的记录文件中清除所有数据，并添加一个 'Clear Event' 标志。

6-16-3 CloseLogFile

语法

```
returnstate = CloseLogFile("datasetname")
```

或

```
returnstate = CloseLogFile("databaselink")
```

备注

参数	类型	描述
returnstate	布尔型	若函数成立则返回状态为1，否则为0。
datasetname	文本	要关闭的数据集的名称，如同文本点或常量。
databaselink	文本	链接到关闭的数据集的名称，如同文本点或常量。

例如

```
CloseLogFile("Process 1")
```

此命令将关闭活跃的记录文件中的数据集。该数据集的记录自动停止。

6-16-4 CloseLogView

语法

```
CloseLogView("datasetname")
```

备注

参数	类型	描述
datasetname	文本	要关闭的数据集的名称，如同文本点或常量。

例如

```
CloseLogView("Process 1")
```

此命令将关闭正在显示已命名的数据集的数据记录查看器。

6-16-5 ExportAndViewLog

语法

```
ExportAndViewLog ("datasetname", "item list",
"format", file, outputfile)
```

或

ExportAndViewLog ("datasetname", TextArray, "format", file, outputfile)

备注

参数	类型	描述
datasetname	文本	要关闭的数据集的名称，如同文本点或常量。
item list	字符串	数据记录中要输出的项和/或组的列表，用逗号分隔。或用"*"输出全部。
TextArray	字符串数组	有一个被指定为1或多个元素的数组规模的文本点。每个元素都包窜一个项或组名称。.
format	字符串	"CSV" 或 "Text"指定输出格式。包括以下内容跟随的后缀'-': B 执行破损信息 D 执行记录数据 T 执行记录时间 M 执行记录毫秒 G 不将 'On Change' 数据分为一组
file	整数	要输出文件的号码，其中0是最新（活跃）文件，1是之前的文件等。
outputfile	字符串	输出文件的文件名称。包括全称路径，如果不存在将会自动创建。

上述所有参数都可选也可省略，假设没有更进一步的参数（即指定必须包括 'format', 'datasetname' 和 'item list'，但'file' and 'output'可省略）。

例如

```
ExportAndViewLog("Balloon", "*")
```

或

```
ExportAndViewLog("Balloon",
"Altitude,Fuel,Burning,Lift,Group 1", "CSV-BDTM", 0,
"output")
```

或

```
ItemList[0] = "Altitude"
ItemList[1] = "Fuel"
ItemList[2] = "Burning"
ItemList[3] = "List"
ItemList[4] = "Group 1"
ExportAndViewLog("Balloon", ItemList, "CSV-BDTM", 0,
"output")
```

所有这些命令将为已命名的数据集到输出文件，而以指定的格式（如同每个ExportLog）输出指定文件中的全部数据。然后它使用Windows文件关联，可以启动一个适当的查看器来显示文件。

6-16-6 ExportLog

语法

```
ExportLog ("datasetname", "item list", "format", file,
outputfile)
```

或

```
ExportLog ("datasetname", TextArray, "format", file,
outputfile)
```

备注

参数	类型	描述
datasetname	文本	要关闭的数据集的名称，如同文本点或常量。
item list	字符串	数据记录中要输出的项和/或组的列表，用逗号分隔。或用"*"输出全部。
TextArray	字符串数组	有一个被指定为1或多个元素的数组规模的文本点。每个元素都包窜一个项或组名称。.
format	字符串	"CSV" 或 "Text"指定输出格式。包括以下内容跟随的后缀'-': B 执行破损信息 D 执行记录数据 T 执行记录时间 M 执行记录毫秒 G 不将 'On Change' 数据分为一组
file	整数	要输出文件的号码，其中0是最新（活跃）文件，1是之前的文件等。
outputfile	字符串	输出文件的文件名称。包括全称路径，如果不存在将会自动创建。

上述所有参数都可选也可省略，假设没有更进一步的参数（即指定必须包括 'format', 'datasetname' 和 'item list'，但'file' and 'output'可省略）。

例如

```
ExportLog("Balloon", "*")
```

或

```
ExportLog("Balloon",
"Altitude,Fuel,Burning,Lift,Group  1"  "CSV-BDTM",  0,
"output")
```

或

```
ItemList[0] = "Altitude"
ItemList[1] = "Fuel"
ItemList[2] = "Burning"
ItemList[3] = "List"
```

```
ItemList[4] = "Group_1"
ExportAndViewLog("Balloon", ItemList, "CSV-BDTM", 0,
"output")
```

所有这些命令将为已命名的数据集到输出文件，而以指定的格式输出指定文件中的全部数据。

6-16-7 OpenLogFile

语法

```
returnstate = OpenLogFile("datasetname")
```

或

```
returnstate = OpenLogFile("databaselink")
```

备注

参数	类型	描述
returnstate	布尔型	可选。若函数成立则返回状态为1，否则为0。
datasetname	文本	要打开的数据集的名称，如同文本点或常量。
databaselink	文本	要打开的数据集链接的名称，如同文本点或常量。

例如

```
OpenLogFile("Balloon")
```

此命令将打开记录文件，并准备开始记录。当函数耗费磁盘空间时，它无法被频繁调用。

6-16-8 OpenLogView

语法

```
OpenLogView("datasetname", "item list", sessionfile)
```

或

```
OpenLogView("datasetname", TextArray, sessionfile)
```

备注

参数	类型	描述
datasetname	文本	要打开的数据集的名称，如同文本点或常量。
item list	字符串	要查看的数据集中的项和/或组列表。用逗号隔开。
TextArray	字符串数组	有一个被指定为1或多个元素的数组规模的文本点。每个元素都包窜一个项或组名称。.
sessionfile	字符串	会话信息文件的可选文件名称。通过会话设置（如存储在会话文件中的窗口位置、大小、颜色、网格选项等）来显示数据记录查看器。如被省略，可以使用之前的设置。

例如

```
OpenLogView("Balloon",
"Altitude,Fuel,Burning,Lift,Group 1")
```

或

```
ItemList [0] = "Altitude"
ItemList [1] = "Fuel"
ItemList [2] = "Burning"
ItemList [3] = "Lift"
ItemList [4] = "Group 1"
OpenLogView("Balloon", ItemList)
```

所有这些命令将打开数据记录查看器，并加载热气球记录文件，以及显示已命名的项。

```
OpenLogView("Balloon", ItemList, "C:\Program Files\Omron\CX-SUPERVISOR\App\MySessionInfo.txt")
```

此命令将打开上面提到的数据记录查看器和热气球记录文件，但是数据记录查看器将总是出现在同一位置，并带有同样的设置-并非最新一次的设置。

6-16-9 StartAuditTrail

语法

```
returnstate = StartAuditTrail()
```

备注

参数	类型	描述
returnstate	布尔型	可选。若函数成立则返回状态为1，否则为0。

例如

```
StartAuditTrail()
```

在选择目标（即Microsoft Access o或SQL）的基础上，此命令将开始所有项的审计追踪记录，这些项均已配置为记录入审计追踪数据库。默认的是，如果某一数据已经存在，数据将被追加到审计追踪数据库，否则将创建一个新的数据库。 'Audit Trail Configuration'对话框可用来配置如何将审计追踪数据记录到Microsoft Access 或SQL 数据库。

6-16-10 StopAuditTrail

语法

```
StopAuditTrail()
```

例如

```
StopAuditTrail()
```

此命令将停止当前审计追踪记录并关闭审计追踪数据库。

6-16-11 StartLogging

语法

```
returnstate = StartLogging("datasetname")
```

or

```
returnstate = StartLogging("databaselink")
```

备注

参数	类型	描述
returnstate	布尔型	可选择。若函数成立则返回状态为1，否则为0。
datasetname	文本	以文本点或常数打开的数据集的名称。
databaselink	文本	以文本点或常数开始记录的数据库链接名称。

例如

```
StartLogging("Process 1")
```

这条命令将会开始记录被命名的数据集下的所有条目。关闭的文件将会被自动打开。

6-16-12 StopLogging

语法

```
returnstate = StopLogging("datasetname")
```

或

```
returnstate = StopLogging("databaselink")
```

备注

参数	类型	描述
returnstate	布尔型	可选择。若函数成立则返回状态为1，否则为0。
datasetname	text	以文本点或常数打开的数据集的名称
databaselink	text	以文本点或常数开始记录的数据库链接名称。

例如

```
StopLogging("Process 1")
```

这条命令将会停止记录被命名的数据集下的所有条目。

6-17 Database Commands

6-17-1 DBAddNew

描述

添加一个新的记录至记录集。如果记录集以'只读'的锁定形式打开，函数将不能成功运行。

语法

```
returnstate = DBAddNew(level)
```

备注

参数	类型	描述
returnstate	布尔型	可选择。若函数成立则返回状态为1，否则为0。
level	文本	指定连接级别的文本点或常数。应当是一个字段或记录集级别。

例如

```
Result = DBAddNew("Northwind.Order Details")
```

使用记录集连接级别添加一个新记录，该记录有来自属性类型'Add'相关联的全部字段的值。函数正确时，点'Result'设置为正确。

```
DBAddNew("Northwind.Order Details.OrderID")
DBAddNew("Northwind.Order Details.ProductID")
DBAddNew("Northwind.Order Details.Quantity")
DBAddNew("Northwind.Order Details.UnitPrice")
DBUpdate("Northwind.Order Details")
```

使用一个记录集连接级别，每一个所需字段都将加入新的记录中，新的记录使用多重调用函数DBAddNew()。当记录完成后，调用DBUpdate()函数添加记录。

注意: 通过记录集级别使用DBAddNew()时，记录集的配置必须符合这种类型函数的运行。也就是说，创建新记录时必须使用含有主键和‘no null’值的字段。但

注意: 在记录集级别中使用时，含有属性类型‘Add’的记录集相关的所有字段豆浆杯添加（类似调用DBAddNew()），并且记录被更新（类似调用DBUpdate()）。与'Add'属性相关 的点可以是数组点，因此能够在一次操作中增加多个记录。

注意: 当使用字段级别连接是，在通过调用DBExecute() 命令 "CancelUpdate"以调用DBUpdate() 函数之前，操作可能在任何阶段被取消。

注意: 只有属性类型为'Add'的字段可以添加到Recordset。 调用DBUpdate（）时的关联点的数值将用于创建记录。

注意: 根据ADO提供程序，添加的记录在重新查询记录集之前可能不可见。请参阅DBExecute，参数Requery获取更多信息。

6-17-2 DBClose

描述

关闭连接或记录集。关闭连接将自动关闭与其关联的所有记录集。可以通过选择适当的级别来单独关闭记录集。

语法

```
returnstate = DBClose(level)
```

备注

参数	类型	描述
returnstate	布尔型	可选择。若函数成立则返回状态为1，否则为0。
level	文本	指定连接级别的文本点或常量。应该是一个字段或记录集级别。

例如

```
Result = DBClose("Northwind.Order Details")
```

关闭 'Order Details' 记录集

```
Result = DBClose("Northwind")
```

关闭与Northwind数据库的连接以及被打开的其他任何记录集

6-17-3 DBDelete

描述

从当前记录位置删除指定数量的记录。此函数仅在记录集级别工作。如果记录集使用“只读”的锁定状态打开，此函数将失效。

语法

```
returnstate = DBDelete(level, quantity)
```

备注

参数	类型	描述
returnstate	布尔型	若函数成立则返回状态为1，否则为0。
level	文本	明确连接级别的文本点或常数。应当是一个字段或记录集级别。
quantity	int	要删除的记录集的数量。

例如

```
Result = DBDelete("Northwind.Order Details", 10)
```

删除记录集中的接下来的10个记录。

```
DBMove("First")
Result = DBDelete("Northwind.Order Details", 10)
```

删除前10个记录。

6-17-4 DBExecute

描述

执行数据库（DBExecute）函数允许执行其他命令，并允许通过执行新命令进行之后的扩展，而无需创建更多新的数据库函数。

语法

```
return = DBExecute(level, command, parameter)
```

备注

参数	类型	描述
return		若函数成立则返回状态为1，否则为0。除了返回记录数字的“Find”和“FindNext”命令（如果找到或未找到），其他将当前记录设置为EOF并返回-1。
level	文本	明确连接级别的文本点或常量，这取决于指定的命令。
command	文本	要执行的命令。 可能是下面列出的命令之一。
parameter	文本	命令参数只对某些命令是必需的。 对于"Connection"，此参数应该保存新的连接字符串。 对于“Find”和“FindNext”，此参数应为搜索条件。 对于“Source”，这是记录集源。 对于"Filter"，这是记录集过滤器。

例如

```
Pos = DBExecute("Northwind.Order Details", "Find",
"UnitPrice > 14.00")
```

从当前位置开始，查找满足指定条件的下一条记录。 有效的搜索条件包括：“ProductName LIKE'G *”通配符搜索查找以“G” 开头，且“Quantity = 5”，“Price> = 6.99”的所有记录。 只允许使用单个搜索值，使用带有“AND”或“OR”的多个值不能使用。

```
DBExecute("Connection1.Recordset1", "Source",
"Table2")
```

修改记录集源以打开与配置不同的表格。

```
DBExecute("Northwind.Shippers", "Filter",
"CompanyName = 'United Package'")
```

应用过滤器以仅显示公司名称为“United Package”的记录。

```
DBExecute("Northwind.Shippers", "Filter", "")
```

取消现有的过滤器（通过传递一个空字符串）

DBExecute命令

命令	连接级别	描述
Connection	连接	修改连接字符串。
BeginTrans	连接	开始一个新的事务。
CommitTrans	连接	保存任何待定的更改和结束当前事务。
RollbackTrans	连接	取消所做的任何更改和结束事务。
CommitTransAll	连接	保存所有更改并全部结束事务。
RollbackTransAll	连接	取消所有更改并全部结束事务。
TransCount	连接	返回待定的事务的数量。
Requery	记录集	重新运行记录集查询。
CancelUpdate	记录集	取消 DBAddNew 操作
Find	记录集	在记录集中查找指定的条件。
FinNext	记录集	组合 DBMove("Next"), DBFind() 操作
Source	记录集	修改记录集源。
Filter	记录集	将过滤器应用于记录集。
Save	记录集	以XML格式保存记录集。

6-17-5 DBGetLastError

描述

返回数据库提供程序生成的最后一个错误字符串，并将其显示在消息框中。

语法

```
returnstate = DBGetLastError(level, display)
```

备注

参数	类型	描述
returnstate	文本	来自提供程序的错误消息
level	文本	明确连接级别的文本点或常量。 这必须是连接级别。
display	布尔型	可选标志。 默认情况下，DBGetLastError将在消息框中显示提供程序错误消息。 将此标志设置为FALSE可阻止此动作。

例如

```
DBGetLastError("Northwind")
```

或

```
DBGetLastError("Northwind", TRUE)
```

上述两行都将获得并显示Northwind连接的最后一个错误。

```
ErrMsg = DBGetLastError("Northwind", FALSE)
```

Northwind连接发生的最后一个错误是存储文本点'ErrMsg'，而不显示消息框。

6-17-6 DBMove

描述

DBMove函数允许通过移动记录集中的'当前记录'的位置来浏览记录集。 当首先打开记录集时，第一个记录是当前记录。

语法

```
returnstate = DBMove(level, direction, position)
```

备注

参数	类型	描述
returnstate	布尔型	若函数成立则返回状态为1，否则为0。
level	文本	明确连接级别的文本点或常量。 这必须是记录集级别。

参数	类型	描述
direction	文本	指示移动到的位置的文本字符串。 可能是以下之一： "First" "Last" "Next" "Previous" "Position" "FirstPage" "LastPage" "NextPage" "PreviousPage" "Page" "Bookmark"
position	整数/实数	只有在使用“位置”，“页面”和“书签”方向时，才需要此可选参数。 当与“位置”和“页面”一起使用时，此参数必须是整数，并且是要移动到的记录或页码。 当与“书签”一起使用时，此参数必须是实数。

例如

```
DBMove("Northwind.Order Details", "First")
```

转到记录集中的第一个记录。

```
pos = 3
DBMove("Northwind.Order Details", "Position", pos)
```

转到记录集的第三条记录。

```
DBMove("Northwind.Order Details", "Page", 6)
```

转到记录集的第六页。

注意: 书签从函数数据库属性‘DBProperty’返回，它们使您能够返回到‘marked’记录，即使记录已被添加或已删除。

注意: 一些提供程序不支持以"Previous"方向移动，即光标是'Forward-Only'。 一些'Forward-Only'提供者允许移动"First"，而一些是严格仅前向的提供程序，即，记录集必须被有效地重新询问一个关闭再打开的组合操作，以将光标重置为记录集的开始。 一些支持移动"Previous"的提供程序不支持移动到"Position"。 但是，为了保持一致性，CX-Supervisor确保所有操作（"Bookmarks"除外）将适用于与任何提供程序的任何连接，但是在设计使用'Forward-Only'光标的应用程序时，需要牢记 可能会有一些杂技在幕后进行。 有关如何检查有效的光标类型的详细信息，请参阅DBSupports（）。

注意: 只有在提供程序特别支持的情况下，书签才能工作。

6-17-7 DBOpen

描述

打开连接或记录集。 打开连接将自动打开与其关联的所有记录集，这杯标记为自动打开。 通过选择适当的级别可以独立打开记录集。

语法

```
returnstate = DBOpen(level)
```

备注

参数	类型	描述
returnstate	布尔型	若函数成立则返回状态为1，否则为0。
level	文本	明确连接级别的文本点或常量。 这必须是记录集级别。

例如

```
DBOpen("Northwind")
```

打开与Northwind数据库的连接，并自动打开任何设置为“在连接时打开”的记录集。

```
done = DBOpen("Northwind.Order Details")
```

只打开某一个指定的记录集。

6-17-8 DBProperty

描述

返回请求的属性。 此函数在记录集和字段级别上操作。 返回值的类型取决于请求的属性。

语法

```
returnstate = DBProperty(level, property)
```

备注

参数	类型	描述
returnstate	布尔型	若函数成立则返回状态为1，否则为0。
level	文本	明确连接级别的文本点或常量。 这必须是记录集级别。
property	文本	要获取的属性的名称。 有关详细信息，请参阅“记录集属性”和“字段属性”表。

例如

```
Page = DBProperty("CSV.Result", "CurrentPage")
```

获取CSV.Result 记录集的当前页面。

```
FieldSize = DBProperty("Northwind.Customers.Address",
"Size")
```

获取 'Address' 字段的大小。

注意: 记录集只有在打开时才返回有效的属性。

记录集属性

一个记录集的属性如下:

属性	描述	返回类型
"CurrentRecord	"Current cursor position	整数
"RecordCount	"Number of records in the Recordset.	整数

属性	描述	返回类型
"Bookmark	"Record marker.	实数
"PageCount	"Number of pages in the Recordset.	整数
"PageSize	"Number of records in a page.	整数
"CurrentPage	"Page in which the cursor position resides.	整数
"Source	"Command or SQL that created the Recordset.	文本
"Sort	"Field name(s) the Recordset is sorted on.	文本
"FieldCount	"Number of fields(columns) in the Recordset.	整数
"BOF	"Current position is at the start of the Recordset.	布尔型
"EOF	"Current position is at the end of the Recordset.	布尔型

字段属性

一个字段的属性如下:

属性	描述	返回类型
"Value	"Value of the field at the current position.	如字段类型
"Name	"Name of the Field.	字符串
"Type	"The fields data type.	字符串
"Size	"Maximum width of the field.	整数

6-17-9 DBRead

描述

从记录集中读取记录到相关联的点，如果关联点是数组点，则读取整个记录页。此函数在记录集和字段两个级别上操作。在字段级别，来自记录集当前位置的相关列值将被复制到点（被复制的元素数=点中元素的数量，在字段级别不应用分页）。

语法

```
returnstate = DBRead(level, reset)
```

备注

参数	类型	描述
returnstate	布尔型	若函数成立则返回状态为1，否则为0。
level	文本	明确连接级别的文本点或常量。 这必须是记录集级别。
reset	布尔型	此参数是可选的，可以省略。 如果省略或TRUE，当读取完成时，记录光标将重置到读取前的位置。

例如

```
DBRead("Northwind.Customers")
```

从 'Customers' 中读取记录的下一页。

```
DBRead("Northwind.Customers", FALSE)
```

从 'Customers' 记录集中读取记录的下一页，将光标留在下一条记录。

```
DBRead("Northwind.Customers.Address")
```

读取地址字段。 如果它是一个数组点，则从后续记录中读取地址，直到数组被填充。

注意: 如果读取操作与后续写入操作相结合，则可以使用重置 = TRUE，即能够读入一组记录 - 重置光标，修改一些字段，然后将更改写回到Recordset。

注意: 使用重置= FALSE将使下一组记录的开始的地方成为当前位置。 如果提供程序仅支持向前移动光标，或者您只想逐步浏览某个页面的记录，则此选项可能会有益。

6-17-10 DBSchema

描述
是读取图式结果或属性或启动新图式条件的问题命令。此函数只有在图式级别可操作。

语法

```
return = DBSchema(level, command, parameters...)
```

备注

参数	类型	描述
return		命令返回的值。 对于某些命令，如“RecordCount”这是一个整数值，对于其他命令，这是一个文本值。
level	文本	明确连接级别的文本点或常量。 这必须在图式级别下进行。
command	文本	该命令必须是以下之一: • "Read" - 将图式页面传输到关联点 • "Set" - 允许修改图式详细信息 • "Type" - 返回当前图式类型 • "Criteria"- 返回当前图式条件 • "Filter" - 返回当前图式过滤器 • "RecordCount" - 返回当前图式中的记录数 • "PageCount" - 返回当前图式中的页数 • "CurrentPage" - 返回当前图式页面

参数	类型	描述
parameters		一些命令需要1个或多个额外的参数。"Read"采用类型为整数的可选参数'Page Number'。 如果没有提供'Page Number'，则该函数在首次调用时返回第1页，并在每次后续调用时自动返回下一页图式，在返回所有页面时循环回到开始。 "Set" 需要三个文本参数，分别为图式'Name', 'Criteria' 和'Filter'。

例如

```
NumberOfRecords = DBSchema("Invoice.Data Types", "RecordCount")
```

读取图式中的记录的数量。

```
DBSchema("Invoice.Data types", "Read", 2)
```

将图式页面2的结果读入相关点。

```
DBSchema("Invoice.Data Types", "Set", "Columns", "COLUMN_NAME", "")
```

设置新的图式以返回列目名称。

6-17-11 DBState

描述

报告指定级别是否处于请求状态。

语法

```
return = DBState(level, state)
```

备注

参数	类型	描述
return	布尔型	若函数成立则返回状态为1，否则为0。
level	文本	明确连接级别的文本点或常量。 这可以是在连接或记录集级别上进行。
state	鑷囨湅	请求的状态必须是 "Open" 或 "Closed"

例如

```
State = DBState("Invoice", "Closed")
```

检查连接 "Invoice" 当前是否已关闭。

```
State = DBState("Northwind.Customers", "Open")
```

检查记录集"Customers" 当前是否已打开。

6-17-12 DBSupports

描述

如果指定的记录集支持所请求的操作，则返回TRUE。

语法

```
return = DBSupports(level, operation)
```

备注

参数	类型	描述
return	布尔型	若函数成立则返回状态为1，否则为0。
level	文本	明确连接级别的文本点或常量。 这可以是在连接或记录集级别上进行。
operation	文本	请求的操作可以是以下之一: "AddNew" "Bookmark" "Delete" "Find" "MovePrevious" "Update"

例如

```
Result = DBSupports("CSV.Recordset1", "Delete")
```

检查是否可以在“Recordset1”中删除记录

注意: 如果不支持“MovePrevious”操作，则只支持'Forward-Only'光标移动。

6-17-13 DBUpdate

描述

更新正在记录集中添加的记录。 与DBAddNew结合使用以提交新记录。

注意: 只有在字段级别使用DBAddNew时才需要DBUpdate。 当在记录集级别使用DBAddNew时，不需要另外的DBUpdate，因为这是自动执行的。

语法

```
returnstate = DBUpdate(level)
```

备注

参数	类型	描述
return	布尔型	若函数成立则返回状态为1，否则为0。
level	文本	明确连接级别的文本点或常量。 这可以是在连接或记录集级别上进行。

例如

```
DBAddNew("Northwind.Order Details.OrderID")
DBAddNew("Northwind.Order Details.ProductID")
DBAddNew("Northwind.Order Details.Quantity")
DBAddNew("Northwind.Order Details.UnitPrice")
DBUpdate("Northwind.Order Details")
```

每个必需字段都将通过多次调用DBAddNew（）添加到新记录。 当记录完成时，通过调用DBUpdate（）函数将其添加到记录集。

6-17-14 DBWrite

描述

这一函数将相关点记录在记录集中，并能同时作用于记录级别和字段级别。在记录级别，相关点的值会从当前记录开始写入记录集中（每个点的相关值会分页储存）。在字段级别，相关点的值会从当前位置开始写入记录集中。写入的元素数=点中的元素数。若记录集设为'Read Only'，该函数将无法正常运行。

语法

```
return = DBWrite(level, reset)
```

备注

参数	类型	描述
return	布尔型	若有一特定级别处于指定状态为1，否则为0。
level	文本	明确连接级别的文本点或常数。可以是连接级别或记录级别。
reset	布尔型	该参数可选择也可省略。若省略则值TRUE，在写入完成后，光标将复位至之前的位置。

例如

```
DBWrite("Northwind.Customers")
```

将所有点的值写入关联的顾客字段。

```
DBWrite("Northwind.Customers.Address", FALSE)
```

将点的值写入地址栏，并使光标停留在下一记录处。

6-18 串行端口函数

6-18-1 InputCOMPort

描述

为接收ASCII文本信息设置的一系列通信端口。所有接收到的信息都将储存在文本点中。若信息成功接收则布尔标记将设为真。用户在收到信息后可自行重设此标志以显示新接收的信息。这一函数在接受多条信息时，仅需每次在终端字符接收后开启一次。

语法

```
ReturnState = InputCOMPort(PortNumber, Message,
MessagePresent, MaxLength)
```

备注

参数	类型	描述
ReturnState	布尔型	成功则值为真，否则为假。
PortNumber	整数	经过SetupCOMPort函数设置并通过OpenCOMPort打开过的端口号码。

参数	类型	描述
message	文本	储存通过端口接收的ASCII文本信息的文本点。
MessagePresent	布尔型	表示信息已成功接收的布尔点。
MaxLength	整数	可选。输入终止前传输的最大长度。应用于接收某一确定且无终端字符的长度包的情况中。

例如:

```
bState = InputCOMPort(1, Msg, bTransmission)
```

6-18-2 OutputCOMPort

描述

通过指定的串行通信端口发送ASCII文本信息

语法

```
ReturnState = OutputCOMPort(PortNumber, Message)
```

备注

参数	类型	描述
ReturnState	布尔型	成功则值为真，否则为假。
PortNumber	整数	经过SetupCOMPort函数设置并通过OpenCOMPort打开过的端口号码。
message	文本	储存通过端口接收的ASCII文本信息的文本点。

例如:

```
bState = OutputCOMPort(1, Msg)
```

6-18-3 CloseCOMPort

描述

关闭个人计算机上指定的串行通信端口。端口在关闭前必须经过设置并处于开启状态。

语法

```
ReturnState = CloseCOMPort(PortNumber)
```

备注

参数	类型	描述
ReturnState	布尔型	成功则值为真，否则为假。
PortNumber	整数	经过SetupCOMPort函数设置并通过OpenCOMPort打开过的端口号。

例如:

```
bState = CloseCOMPort(1)
```

6-18-4 OpenCOMPort

描述

打开个人计算机上的串行通信端口以传输或接收数据。端口在打开前必须经过设置。

语法

```
ReturnState = OpenCOMPort(PortNumber)
```

备注

参数	类型	描述
ReturnState	布尔型	成功则值为真，否则为假。
PortNumber	整数	经过SetupCOMPort函数设置过的端口号码。

例如:

```
bState  = OpenCOMPort(1)
```

6-18-5 SetupCOMPort

描述

对个人计算机上的串行通信端口进行设置以传输或接收数据。

语法

```
ReturnState = SetupCOMPort(PortNumber,
ConfigurationString, HandShaking, TerminationChar,
ControlCharFlag, TermMode)
```

备注

参数	类型	描述
ReturnState	布尔型	成功则值为真，否则为假。
PortNumber	整数	表示需要的波特率，校验以及数据位和停止位的数目的字符串。
HandShaking	整数	请求的信号交换协议。有效值为: 0 - None 1 - XonXoff 2 - RTS 3 - RTS & XonXoff
TerminationChar	整数	表示信息结束的字符。
ControlCharFlag	布尔型	表示接收信息中应该忽略的控制字符的标记。

参数	类型	描述
TermMode	整数	可选。表示终端字符使用方法的标记。 @ONINPUT（或值1）InputComPort函数识别的终端字符。若省略则该值为默认值。 @ONOUTPUT（或值2）OutputComPort函数附加的终端字符。 @ONINPUT|@ONOUTPUT（或值3）以上两者。

例如:

```
bState  = SetupCOMPort(2, "9600,N,8,1", 0, 0x0D, TRUE)
```

6-19 ActiveX 函数

6-19-1 GetProperty

描述

获取OLE对象的属性值并将其储存在点中。

语法

```
propertyvalue = GetProperty(object, property, ...)
```

备注

参数	类型	描述
propertyvalue	n/a	值的属性。其类型取决于属性的类型。
object	文本	要获取其属性的OLE对象名称。
property	文本	要获取的属性的名称。
---	n/a	属性参数的数值

例如

```
OLE1Height = GetProperty("OLE1", "Height")
```

读出'OLE1'这一OLE对象的'Height'属性，并将其储存在'OLEHeight'这一点中。

```
DM100Value = GetProperty("CXComms1", "DM", 100)
```

读出'CXComms1'这一OLE对象的'DM'属性（参数为100），并将其存储在'DM100VALUE'这一点中。

6-19-2 PutProperty

描述

Puts a value stored in a point into the property of an OLE object.

语法

```
PutProperty(object, property, ..., value)
```

备注

参数	类型	描述
object	文本	要改变其属性的OLE对象名称。
property	文本	要写入的属性的名称。
---	n/a	属性参数的数值。
value	n/a	写入属性的值。其类型取决于属性的类型。可以为数字。

例如

```
PutProperty("OLE1", "Left", NewLeftValue)
```

将储存在NewLeftValue点中的值写入OLE对象'OLE1'的'Left'属性中。

```
PutProperty("CXComms1", "DM" 10, NewValue)
```

将储存在NewValue点中的值写入OLE对象'CXComms1'的'DM'属性中（参数为10）。

```
PutProperty("Gauge1", "Value", 25.2)
```

将对象'Gauge1'的值记为25.2

6-19-3 Execute

描述

执行一个 OLE 对象的方法。

语法

```
Execute(object, method, ...)
```

备注

参数	类型	描述
object	文本	OLE对象的名称。
method	文本	要执行方法的名称。
---	n/a	方法参数的数值。

例如

```
Execute("OLE1", "Start")
```

调用 'OLE1'对象的 'Start'方法。

```
Execute("CXComms1", "OpenPLC", "MyPLC")
```

调用OLE对象'CXComms1'的'OpenPLC'方法，方法中带有一个'MyPLC'文本参数。

6-19-4 ExecuteVBScript

描述

可通过创建别名让Visual Basic脚本实现联机执行。这一函数需要用到Windows脚本宿主。Windows脚本宿主所支持的函数及更详细内容请见第五部分。

语法

```
@VBSCRIPT
@ENDSCRIPT
```

例如

```
@VBSCRIPT
```

```
        OLE1.LEFT = Point("PointName")
    @ENDSCRIPT
```

该Visual Basic脚本会将点‘PointName’点的值记在OLE对象‘OLE1’的‘Left’属性中。

6-19-5 ExecuteJScript

描述

可通过创建别名让Java脚本实现联机执行。Windows脚本宿主所支持的函数及更详细内容请见附录C。

语法

```
@JSCRIPT
@ENDSCRIPT
```

例如

```
@JSCRIPT
    Point("PointName") = OLE_1.Height;
@ENDSCRIPT
```

该Java脚本会将OLE对象‘OLE1’的'Height'属性值记在点'PointName'中。

注意: Java脚本中无法使用{or}字符。若想使用，可将脚本存入文本文档后使用ExecuteJScriptFile函数。

6-19-6 ExecuteVBScriptFile

描述

执行储存在文本文档中的Visual Basic脚本。该函数必须用到Windows脚本宿主。支持的函数详见第五部分。

语法

```
returnstate = ExecuteVBScriptFile(scriptfile)
```

备注

参数	类型	描述
returnstate	布尔型	若函数成立则返回状态为1，否则为0。
scriptfile	文本	要执行Visual Basic脚本的文档名称。

例如

```
returnstate = ExecuteVBScriptFile("c:\vbscript.txt")
```

执行储存到"c:\vbscript.txt”的 Visual Basic脚本。

6-19-7 ExecuteJScriptFile

描述

将Java脚本储存到要执行的文本文档中。该函数必须用到Windows脚本宿主。支持的函数详见附录C。

语法

```
returnstate = ExecuteJScriptFile(scriptfile)
```

备注

参数	类型	描述
returnstate	布尔型	若函数成立则返回状态为1，否则为0。

参数	类型	描述
scriptfile	文本	要执行Java脚本的文档名称。

例如

```
returnstate = ExecuteJScriptFile("c:\jscript.txt")
```

执行储存到"c:\jscript.txt"的Java脚本。

6-19-8 GenerateEvent

描述

这一命令仅用于通过CX-Supervisor通信控制进行远程连接（可参考用户手册第十五部分，连接远程应用程序）。该命令可以使服务器将未请求的数据发送回客户端。该数据可由客户端的OnEvent处理器接收。

参数数据可根据客户端需要的数据类型进行设置。

语法

```
returnstate = GenerateEvent(param1, param2, param3)
```

备注

参数	类型	描述
returnstate	布尔型	若函数成立则返回状态为1，否则为0。
param1	文本	可选。要发送的数据参数。
param2	文本	可选。要发送的数据参数。
param3	文本	可选。要发送的数据参数。

例如

```
returnstate = GenerateEvent ("Archive", "", "")
```

向客户端应用程序发送'Archive'事件，强制客户端执行特定的存档操作。没有用到第二和第三个参数。

```
returnstate = GenerateEvent ("[Alarm Set]", "Boiler
alarm", "95.5")
```

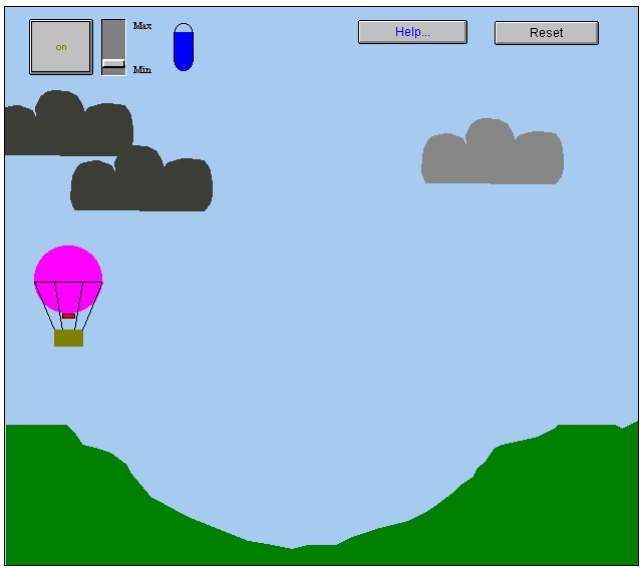
向客户端应用程序发送事件，该事件为'已将热水器报警器的过程值设为95.5'。

第七部分
脚本示例

本部分为脚本的示例应用程序。该脚本为执行基本命令的典型脚本。我们将分两部分展开讲解，首先整体讲解，再逐行讲解。

7-1 热气球脚本

下方的脚本用于一款简易的游戏。



玩家要将热气球停在右侧的平台上，并通过大/小滑块控制飞行。点击重置按钮会清除当前游戏并开始新游戏。点击开/关按钮开始游戏。
当热气球在空中时，云彩会在水平方向上缓慢移动并有轻微的颜色变化。在任意时刻点击帮助会弹出特殊帮助页面。点击帮助页面的关闭按钮会继续回到游戏。蓝色的仪表显示的是已消耗的和剩余的燃料量。
这一项目中包含三个页面脚本和一个对象。三个页面脚本有不同的启用时间间隔，分别为10毫秒，100毫秒和1000毫秒。
启用时间间隔为10毫秒页面脚本用于确定云彩的位置以及每片云彩的移动速度。启用时间间隔为1000毫秒页面脚本用于确定热气球对于不同条件的反应。
启用时间间隔为100毫秒页面脚本确定游戏的整体设置，即根据玩家输入的指令移动气球位置。页面脚本如下：

```
IF burner AND alt > 400.0 THEN
    burner = FALSE
ENDIF
IF burner THEN
    fuel = fuel - rate
    IF fuel < 0.0 THEN
        fuel = 0.0
        burner = FALSE
```

```
    ENDIF
ENDIF

IF burner AND fuel > 0.0 AND rate > 0.0 THEN
    lift = lift + rate/5.0
ELSE
    IF alt > 140.0 THEN
        lift = lift - 0.2
    ENDIF
ENDIF
IF lift < -10.0 THEN
    lift = -10.0
ENDIF
alt = alt + lift
IF alt <= 140.0 THEN
    IF distance>630.0 AND distance<660.0 AND lift>=-
    3.0 THEN
        winner = TRUE
        burner = FALSE
    ENDIF
    IF lift < -3.0 then
        crash = TRUE
        burner = FALSE
    ENDIF
    lift = 0.0
ENDIF

speed = (alt-140.0 )/100.0
IF speed < 0.0 then
    speed = 0.0
ENDIF

distance = distance + speed
```

以下为对脚本的逐行描述。

```
IF burner AND alt > 400.0 THEN
    burner = FALSE
ENDIF
```

若燃烧机处于开启状态，则布尔点'burner'设为'TRUE'。随后若气球的高度，即点‘alt’超过400，则燃烧器关闭。点‘alt’以像素为单位，值介于140到1000之间，因此值为400时即为高400像素。

```
IF burner THEN
    fuel = fuel - rate
    IF fuel < 0.0 THEN
        fuel = 0.0
        burner = FALSE
    ENDIF
ENDIF
```

若燃烧机处于开启状态，则燃料量随热气球上升的速度的增大而减少。上升速度，即点‘rate’，能够通过移动滑块进行设置。若点‘fuel’的值低于零，则燃料耗尽，燃烧机关闭。

```
IF burner AND fuel > 0.0 AND rate > 0.0 THEN
    lift = lift + rate/5.0
ELSE
```

```
        IF alt > 140.0 THEN
            lift = lift - 0.2
        ENDIF
    ENDIF
```

若燃烧机处于开启状态并仍有剩余燃料，且上升速率大于零（即热气球起飞），则点'lift'以上升速率的五分之一的速度增长，热气球上升。否则热气球下降，点'lift'以0.2的速度减少。

```
    IF lift < -10.0 THEN
        lift = -10.0
    ENDIF
```

点'lift'点的最小值为-10.

```
    alt = alt + lift
```

热气球的高度随点'lift'增大而增长。

```
    IF alt <= 140.0 THEN
        IF distance>630.0 AND distance<660.0 AND lift>=-
        3.0 THEN
            winner = TRUE
            burner = FALSE
        ENDIF
```

若热气球回到地面（即点'alt'为140）且落到了平台上（即点'distance'显示的热气球的位置为630到660像素），同时下降速度不快（由点'lift'决定），则游戏胜利。

```
      IF lift < -3.0 then
        crash = TRUE
        burner = FALSE
      ENDIF
```

若热气球回到地面（即点'alt'点为140）且下降速度不快（由点'lift'点决定），则游戏失败。

```
    lift = 0.0
    ENDIF
```

点'lift'重设。

```
    speed = (alt-140.0 )/100.0
    IF speed < 0.0 then
        speed = 0.0
    ENDIF
```

点'speed'根据热气球高度进行计算。

```
    distance = distance + speed
```

点'distance'根据速度进行计算。

第八部分
调色板

本章节主要介绍颜色调色板。颜色可通过其名字和代码进行区分。下表即为颜色名称和代码的参照表。颜色名称长度若超过一个词则单词间用下划线或连字符连接。特定的颜色可在开发环境的当前工作中进行修改，但修改的颜色并不能保存在页面或项目中。若想保存修改，需要到项目菜单下的常规设置子菜单中的颜色调色盘内进行修改。
在16色位的屏幕分辨率中（详细信息请查询Microsoft Windows说明书），16到65号颜色能够通过16种基本颜色显示出来。在更高的颜色分辨率中颜色不会通过基本颜色的混淆来表现。

No.	Colour	No.	Colour
0	黑色	12	紫色
1	蓝色	13	橄榄色
2	绿色	14	深灰色
3	青色	15	浅灰色
4	红色	16	浅绿色
5	洋红色	17	浅蓝色
6	黄色	18	灰白色
7	白色	19	灰色
8	深蓝色	20	樱桃红
9	深绿色	21	银色
10	蓝绿色	22	苹果色
11	棕色	23	橘色
		24-65	未使用

附录A OPC通信控制

本附录包含可用组件属性的列表，并提供了Visual Basic脚本接口的详细信息。这些属性可以在运行时使用Visual Basic脚本命令设置，例如：
OMRONCXOPCCommunicationsControl1.ServerNodeName = "\\NAME"
脚本接口定义了OPC通信控制的Visual Basic脚本接口。有关运行Visual Basic脚本的更多信息，请参阅ExecuteVBScript脚本函数。

A.1 组件属性

属性标题	示例	描述
DisplayErrors	True False	当设置为True时，对象将显示任何错误的消息框。如果设置为False，则不显示错误消息。
ProjectName		包含客户端设置的.OPC文件的名称。
ServerComputerName	"MyPC	"这是带有OPC服务器的PC的名称。
ServerName		要连接的OPC服务器的名称。例如OMRON.OpenDataServer.1
ServerProjectName		可选文件名，如果指定，则会导致OPC服务器使用指定的文件（如果服务器支持）。

A.2 脚本接口

脚本接口定义了OPC通信控制的方法。

A.3 函数

Value	获取和设置OPC项值的函数。
Read	读取OPC项的值的函数。
Write	写入OPC项的值的函数。

A.3.1 Value

读取或写入OPC项的值。
示例1 - 读取值：

```
intVal =
 OMRONCXOPCCommunicationsControl1.Value
("MyGroup", "BoilerTemp")
```

在本示例中，OPC组中名为“ MyGroup”的OPC项“ BoilerTemp”将从OPC服务器中读取，并存储在“ intVal”中。
示例2 - 写入值：

```
OMRONCXOPCCommunicationsControl1.Value("MyGroup",
"BoilerTemp") = 50
```

在此示例中，值50将被写入OPC项‘BoilerTemp’。

注意: 'Value'是默认属性，所以假设省略。因此，以下示例是相同的：

```
intVal =
OMRONCXOPCCommunicationsControl1.Value("MyGroup",
"BoilerTemp")
```

并

```
intVal = OMRONCXOPCCommunicationsControl1 ("MyGroup",
"BoilerTemp")
```

A.3.2 Read

读取OPC项的值。

同步读取示例：

```
intVal =
 OMRONCXOPCCommunicationsControl1.Read
("MyGroup", "BoilerTemp")
```

在本示例中，OPC组中名为“ MyGroup”的OPC项‘ BoilerTemp’将从OPC服务器中读取，并存储在‘intVal’中。脚本将等待读取操作完成，然后继续执行下一行。这与‘Value’方法的操作相同。

A.3.3 Write

写入OPC项的值。

同步写入示例：

```
OMRONCXOPCCommunicationsControl1.Write
"MyGroup", "BoilerTemp", NewValue
```

在本示例中，‘NewValue’将被写入OPC组中称为“ MyGroup”的OPC项‘BoilerTemp’中。脚本将等待写入操作完成，然后继续执行下一行。这与‘Value’方法的操作相同。

附录B
CX-Server通信控制I

当选择项目设置 - >高级设置选项“ 允许通过‘CXServer’控制对PLC进行高级脚本访问” 选项时，将自动创建CX-Server通信控制以允许脚本访问CX-Server函数。此ActiveX控件总命名为“ CXServer”（没有任何连字符），并且可以始终从任何脚本中使用。
本附录包含脚本接口中可用的组件属性和方法的列表。

B.1 Functions

Value	在PLC中获取和设置存储器区域的函数。此函数允许使用逻辑名称。如果使用数组，则返回第一个元素。
Values	在PLC中获取和设置存储器区域的函数。此函数允许使用逻辑名称。如果使用数组，那么将返回一个带有所有值的SAFEARRAY。
SetDefaultPLC	设置默认PLC的函数。这主要在项目包含多个PLC时使用。
OpenPLC	打开特定的PLC进行通信。
ClosePLC	关闭特定PLC。
Read	读取PLC点的值的函数。
Write	写入PLC点的值的函数。
ReadArea	从PLC读取存储器块的函数。
WriteArea	将存储器块写入PLC的函数。
RunMode	读取/写入PLC当前模式的函数。
TypeName	读取PLC类型的函数（例如CQM1H）。
IsPointValid	检查点名称是否有效。
PLC Memory Functions	A, AR, C, CIO, D, DM, DR, E, EM, G, GR, H, IR, LR, SR, ST, T, TC, TK, W. 用于获取和设置PLC中的存储区的函数。
ListPLCs	属性。保存在项目文件中配置的所有PLC名称的列表。此属性为只读。
ListPoints	属性。保存在项目文件中配置的所有点名称的列表。此属性为只读。
IsBadQuality	检查点当前是否指示“ 质量差” 。
ClockRead	读取PLC Clock
ClockWrite	设置PLC Clock
RawFINS	使原始FINS命令发送到指定PLC的函数。

Active	返回指定PLC的连接状态的函数。
TCGetStatus	返回指定温度控制器的设备状态的函数。
TCRemoteLocal	将指定温度控制器切换到远程或本地模式的函数。
SetDeviceAddress	设置PLC网络、节点、单元号和IP地址。
SetDeviceConfig	设置设备配置的任何元素。
GetDeviceConfig	获取设备配置的任何元素。
UploadProgram	从PLC上传程序。
DownloadProgram	将程序下载到PLC。
Protect	保护程序存储器（或解除保护）。
LastErrorString	发生的最后一个错误的描述。

B.2 Value

从PLC读取地址的值，或者将值写入PLC中的地址中。此函数允许逻辑名称。
示例1 - 使用逻辑名从PLC读取值。

```
intVal = CXServer.Value("BoilerTemp")
```

或

```
intVal = CXServer ("BoilerTemp")
```

在这些示例中，与“ BoilerTemp”相关联的PLC地址将从PLC中读取并存储在“ intVal”中。“ Value”是默认属性，不必指定。
示例2 - 使用逻辑名称将值写入PLC。

```
CXServer.Value("BoilerTemp") = 50
```

或

```
CXServer ("BoilerTemp") = 50
```

在这些示例中，值50将被写入与“ BoilerTemp”相关联的PLC地址。“ Value”是默认属性，不必指定。

B.3 Values

从PLC读取值的数组，或将值的数组写入PLC。此函数允许逻辑名称。如果使用数组，那么将返回一个带有所有值的SAFEARRAY。
示例1 - 使用逻辑名从PLC读取值数组。

```
SomeArray  = CXServer.Values("BoilerTemps")
```

示例2 - 使用逻辑名称将值数组写入PLC。

```
CXServer.Values("BoilerTemps") = SomeArray
```

B.4 SetDefaultPLC

'SetDefaultPLC'函数可用于通知脚本解析器，当特定的PLC已被设置为默认值。一旦设置了默认PLC，则不需要（通过某些函数）指定PLC名称。例如，

```
CXServer.SetDefaultPLC("MyPLC")
intVal = CXServer.Value("BoilerTemp1")
CXServer.Value("BoilerTemp1") = 75
intVal = CXServer.Value("DM50")
```

上面的每个" Value"函数将访问PLC中称为" MyPLC"的数据。

注意: 如果项目中只有一个PLC，则不需要调用"SetDefaultPLC"函数。项目中的第一个PLC将自动设置为默认PLC。

B.5 OpenPLC

打开PLC进行通信。如果未指定PLC，则打开默认PLC。
示例1:

```
CXServer.SetDefaultPLC("MyPLC")
CXServer.OpenPLC()
CXServer.DM(100) = 10
CXServer.DM(50) = 10
```

示例2:

```
CXServer.OpenPLC("MyPLC")
CXServer.DM(100) = 10
```

B.6 ClosePLC

关闭之前打开的PLC。如果未指定PLC，则默认PLC关闭。
示例:

```
CXServer.ClosePLC("MyPLC")
```

B.7 Read

读取PLC点的值的函数。
同步读取示例:

```
intVal = CXServer.Read("MyPLC", "MyPoint", 0)
```

在本例中，点'MyPoint'将从PLC的'MyPLC'中读取并存储在'intVal'中。由于'0'参数，脚本将等待读取操作完成，然后继续执行下一行。这与'Value'方法的操作相同。

注意: 如果PLC未打开，则此命令将使其打开，然后在读取完成后关闭。如果要执行多个读取或写入操作，则先使用OpenPLC命令，执行所有读取和写入，然后（如果需要）使用ClosePLC命令关闭PLC，这样的速度更快，效率更高。

B.8 Write

写入PLC点值的函数。
同步写入示例:

```
CXServer.Write("MyPLC", "MyPoint", NewValue, 0)
```

在此示例中，‘NewValue’将写入PLC中名为‘MyPLC’的点“ MyPoint”。由于‘0’参数，脚本将等待写入操作完成，然后继续执行下一行。这与‘Value’ 方法的操作相同。

注意: 如果PLC未打开，则此命令将使其被打开，然后在写入完成后关闭。如果要执行多个读取或写入操作，则首先使用OpenPLC命令，执行所有读取和写入，然后（如果需要）使用ClosePLC命令关闭PLC，这样的速度更快，效率更高。

B.9 ReadArea

从PLC读取指定的存储器块。
同步读取的示例:

```
MyVariant = CXServer.ReadArea("MyPLC/DM0", 12, vbString)
MyVariant = CXServer.ReadArea("BoilerTemp",10, vbInteger)
MyVariant = CXServer.ReadArea("BoilerTemp", 20)
```

在第一个示例中，DM0到DM11将被读取为来自‘MyPLC’的字符（字符串的一部分），并将存储在‘MyVariant’中。第二个示例演示了为开始地址使用逻辑名也是可行的，并且可以使用任何VB变量类型（例如vbInteger）。第三个例子显示VB Variant类型参数是可选的 - 如果没有指定，则假定为vbInteger。脚本将等待读取操作完成，然后继续执行下一行。

注意: 如果从CX-Supervisor脚本访问，则返回类型应使用以下整数值:

常量	值	描述
vbEmpty	0	未初始化（默认）
vbNull	1	不包含有效数据
vbInteger	2	整数子类型
vbLong	3	长子类型
vbSingle	4	单子类型
vbSingle	5	双子类型
vbCurrency	6	货币子类型
vbDate	7	日期子类型
vbString	8	字符串子类型
vbObject	9	对象
vbError	10	错误子类型
vbBoolean	11	布尔子类型
vbVariant	12	变量（仅用于变量数组）
vbDataObject	13	数据访问对象
vbDecimal	14	小数子类型
vbByte	17	字节子类型

常量	值	描述
vbArray	8192	数组

B.10 WriteArea

将一块存储器写入PLC中的指定区域。

同步写入示例:

```
MyString = "TestString"
CXServer.WriteArea "MyPLC/DM50", 10, MyString
Dim newValue(2)
newValue(1) = 0
newValue(2) = 1
CXServer.WriteArea "BoilerTemp",2,newValue
```

在第一个示例中，‘MyString’的内容将被写入DM50到DM54。‘MyString’中的任何附加数据将被忽略（即如果‘MyString’的长度为15个字符，则前10个字符将被写入DM50到DM54，其余5个字符将被忽略 - {注意：每个PLC地址保存2字符}）。第二个示例显示可以使用逻辑名。脚本将等待写入操作完成，然后继续执行下一行。

B.11 RunMode

读取PLC的当前操作模式（停止/编程，调试，监视，运行），其中0 =停止/编程模式，1 =调试模式，2 =监视模式，4 =运行模式。

示例

```
intMode = CXServer.RunMode("MyPLC")
```

在此示例中，操作模式将从 ‘MyPLC’ 读取并存储在‘ intMode’中。如果 ‘ MyPLC’ 处于'Monitor'模式，那么‘intMode’将设置为值2。

B.12 TypeName

读取PLC的PLC型号名称（例如C200H，CQM1H，CVM1等）。

示例

```
strPLCType = CXServer.TypeName("MyPLC")
```

在本示例中，PLC模型类型将从‘MyPLC’读取，并存储在‘ strPLCType’中。

B.13 IsPointValid

检查是否在CX-Server项目文件中定义了点名称。

示例

```
bValid = CXServer.IsPointValid("MyPoint")
bValid = CXServer.IsPointValid("MyPoint", "MyPLC")
```

在这两个示例中，如果已定义点“ MyPoint”，则布尔变量bValid设置为True。

B.14 PLC Memory Functions

(A, AR, C, CIO, D, DM, DR, E, EM, G, GR, H, IR, LR, SR, ST, T, TC, TK, W)

所有PLC记忆函数（例如A，AR，D，DM等）以完全相同的方式工作。 以下示例使用DM函数来获取和设置PLC中DM地址的值。
示例1

```
intVal = CXServer.DM(100)
```

在本例中，DM100的内容将从PLC中读取并存储在'intVal'中。

注意: 这些示例假定CX-Server项目文件中只有一个PLC，或者使用‘SetDefaultPLC’函数来选择所需的PLC。有关在项目中使用多个PLC的脚本的详细信息，请参阅‘SetDefaultPLC’函数。

示例2

```
CXServer.DM(100) = 75
```

在此示例中，值75将写入PLC中的DM100。
位寻址，即从各个存储器位访问数据，也由这些存储器区域支持：IR，AR，HR和CIO。

示例3

```
bVal = CXServer.IR("100.2")
```

在这个例子中，位IR100.2的状态（即IR100的位2）将从PLC中读取并存储在‘bVal’中（例如，‘bVal’将被设置为TRUE或FALSE）。
示例4

```
CXServer.IR("100.2") = True
```

在本示例中，PLC中的位IR100.2（即IR100的位2）将被设置为True。请注意，是否使用引号是可选的，但是需要区分100.1和100.10。

B.15 ListPLCs

保存配置在项目文件中所有PLC名称的列表。此属性为只读。

示例

```
Dim arrayOfPLCs
Dim nUbound, nLbound
arrayOfPLCs = CXServer.ListPLCs
nLbound = LBound(arrayOfPLCs)
nUbound = UBound(arrayOfPLCs)
For Count = nLbound To nUbound
    MsgBox arrayOfPLCs(Count)
Next
```

在本示例中，项目中配置的PLC名称列表存储在'arrayOfPLCs'中，每个都显示在消息框中。

B.16 ListPoints

保存配置在项目文件或PLC中所有点名称的列表。此属性为只读。

示例

```
Dim arrayOfPoints
Dim nUbound, nLbound
arrayOfPoints = CXServer.ListPoints(sPLC)
nLbound = LBound(arrayOfPoints)
nUbound = UBound(arrayOfPoints)
```

```
For Count = 1 To UBound(arrayOfPoints)
    MsgBox arrayOfPoints (Count)
Next
```

在本示例中，指定在文本点sPLC中的为PLC名称配置的点列表存储在‘arrayOfPoints’中，并且每个都显示在消息框中。

示例2

```
arrayOfPoints = CXServer.ListPoints
```

如果在没有参数的情况下使用ListPoints，则所有PLC的点将被返回。

B.17 IsBadQuality

检查点当前是否指示“质量差”。

示例

```
Dim bBad
bBad = CXServer.IsBadQuality("MyPLC", "MyPoint")
```

注意: 在质量未知的情况下，IsBadQuality将返回True。例如，其中没有发生与点的过往通信。

B.18 ClockRead

读取PLC时钟的函数

示例

```
Dim NewDate
NewDate  = CXServer.ClockRead("PLC1")
'  dates  can  be  manipulated  via  standard  VBScript
methods (FormatDateTime,  DatePart etc.)
TextBox1 = NewDate      ' this uses a Microsoft Forms
Text Box to convert date to string
TextPoint1  = TextBox1 'this writes the date string to
a CX-Supervisor text point
```

B.19 ClockWrite

设置PLC时钟的函数。日期的理想格式为“dd / mm / yyyy hh：mm：ss”。

示例

```
Dim NewDate
'set  time/date  value  here  using  standard  VBScript
methods (Date, Time, Now, CDate etc.)
NewDate = Now   ' This example sets the time to the
current PC time
CXServer.ClockWrite "PLC1",  NewDate
```

B.20 RawFINS

此函数可将原始FINS命令发送到指定的PLC。此功能仅适用于熟悉Omron FINS协议的高级用户。

VBScript示例

```
Dim sFINS
Dim sResponse
sFINS = "0501"
sResponse = CXServer.RawFINS(sFins, sPLC)
txtFINSResponse = sResponse 'txtFINSResponse is a CX-
Supervisor point.
```

B.21 Active

返回一个指定的PLC的连接状态

VBScript 示例

```
bActive = CXServer.Active("MyPLC") ' bActive is a CX-
Supervisor point
```

在此示例中，已连接的状态可从 'MyPLC'读出，并存储在CX-Supervisor点'bActive'中。如果 'MyPLC'被连接，'bActive' 将被设置为 True。

B.22 TCGetStatus

为指定的温度控制器返回状态数据:

示例:

```
Dim bTCStatusResponse
bTCStatusResponse = CXServer.TCGetStatus("E5AK")
'Heating output is bTCStatusResponse(21)
'Cooling output is bTCStatusResponse(22)
'Alarm 1 output is bTCStatusResponse(23)
'Alarm 2 output is bTCStatusResponse(24)
'Alarm 3 output is bTCStatusResponse(25)
'Stopped status is bTCStatusResponse(28)
'Remote status is bTCStatusResponse(30)
```

在此示例中，设备状态作为一数组的字节正在从"E5AK"读出。来自温度控制器的回应，作为一数组的字节被存储在bTCStatusResponse.

B.23 TCRemoteLocal

TCRemoteLocal命令将为指定的温度控制器执行Remote/Local命令。

示例 - 在此例中， "E5AK"设备被设置为本地模式:

```
'Set the device to local mode
CXServer.TCRemoteLocal "E5AK", 1
```

示例- 在此例中， "E5AK" 设备被设置为远程模式:

```
'Set the device to remote mode
CXServer.TCRemoteLocal "E5AK", 0
```

B.24 SetDeviceAddress

此函数可以被用来设置设备地址（网络号码，节点号，单元号以及Ethernet IP地址）的关键元素。号码的范围是0-255，-1代表“忽略此参数”。此函数仅面向高级用户。

注意：此方法不能说明或证明通过的数据，可以通过会阻止设备通信的无效数据。要注意确保所有通过的数据均为有效。当PLC打开并通信时不能使用此方法。

示例:

```
NetworkNum = 1
NodeNum = 2
UnitNum = -1
iPAddress = "10.0.0.1"
bValid    =    CXServer.SetDeviceAddress(    "PLC1",
NetworkNum, NodeNum, UnitNum, IPAddress)
```

注意: 如果没有错误被删除，返回布尔值bValid被设置为True。然而，对正在使用的PLC来说，这并非意味着使用的所有参数都是有效或适合的。

B.25 SetDeviceConfig

这是一个可以被用来设置 CX-Server设备配置的任何元素的函数。所有数据都以文本形式传递。此函数仅面向高级用户。

注意: 此方法不能说明或证明已通过的数据，可以通过会组织设备通信的无效数据。要注意确保所欲通过的数据均为有效。当PLC打开并通信时不能使用此方法。

示例:

```
Device = "PLC1"
Section = "NET"
Entry = "IPADDR"
Setting = "10.0.0.1"
bValid = CXServer.SetDeviceConfig  Device,  Section,
Entry, Setting
```

注意: 如果没有错误被删除，返回布尔值bValid被设置为 True。然而，对正在使用的 PLC 来说，这并非意味着使用的所以参数都是有效或适合的。

目前只支持下列的Section, Entry 和 Setting参数值组合:

- Section = "ADDRESS", Entry = "DNA", Setting = "0"..Setting = "255" - 可以用来设置网络号码
- Section = "ADDRESS", Entry = "DA1", Setting = "0"..Setting = "255" - 可以用来设置节点号
- Section = "ADDRESS", Entry = "UNIT", Setting = "0"..Setting = "255" -t 可以用来设置单元号
- Section = "ADDRESS", Entry = "IPADDR", Setting = "0.0.0.0"..Setting = "255.255.255.255" - 可用来设置Ethernet IP 地址

其他参数值也可发挥作用，不过应该仅限于欧姆龙设备。

B.26 GetDeviceConfig

此函数可用来读取CX-Server 设备配置的任何元素。所有数据都以文本形式传递（或接收）。此函数仅面向高级用户。

示例:

```
Dim Setting
Device = "PLC1"
Section = "NET"
Entry = "IPADDR"
Setting = CXServer.GetDeviceConfig  Device,  Section,
Entry
```

当前支持的参数值与SetDeviceConfig方法中介绍的相同。

B.27 UploadProgram

UploadProgram函数可读取PLC的程序。程序以二进制的形式被读取，并存储在一个用户指定文件。 此函数不可在其他PLC通信时同时使用。如有需要，项目和PLC将自动被打开。此函数仅面向高级用户。

示例：

```
Dim SourceFile
Dim DestinationFile
Sourcefile = ""
DestinationFile = "c:\test1.bin"
CXServer.UploadProgram       "PLC1",       SourceFile,
DestinationFile, 1, 0
```

第一个参数是PLC名称。

第二个参数是原文件名称。为了更新当前程序，该参数应该是空白字符串，但是也可能被设置为存储卡的根目录下的一个文件名称。比如"Example.obj"。

迪桑参数是存储程序的本地文件的名称。 '.bin'的文件扩展是二进制文件的标志。

注意：第四和第五个参数被保留，分别为1和0.

B.28 DownloadProgram

DownloadProgram函数可被用来向PLC文件写入程序。此函数不能在其他PLC通信时使用。如有需要，项目和PLC将自动被打开。此函数仅面向高级用户。

注意: 注意使用此函数，以确保程序写入对程序所被下载至的PLC有效。

示例：

```
bValid        =CXServer.DownloadProgram        "PLC1",
"c:\test2.bin", "", 1, 0
```

第一个参数是PLC名称。

第二个参数是原文件名称。为了更新当前程序，该参数应该是空白字符串，但是也可能被设置为存储卡的根目录下的一个文件名称。比如"Example.obj"。

迪桑参数是存储程序的本地文件的名称。 '.bin'的文件扩展是二进制文件的标志。

注意: 第四和第五个参数被保留，分别为1和0.

B.29 Protect

Protect函数可用来保护（或移除保护）PLC程序存储器。此函数不能再其他PLC通信时使用。如有需要，项目和PLC将自动被打开。此函数仅面向高级用户。

例1
(对CS系列PLC 设置保护)

```
Dim SetProtection
Dim PasswordString
Dim PasswordNumber
EnableProtection = true
```

```
PasswordString = "Password"
PasswordNumber = 0
CXServer.Protect       "PLC1",       EnableProtection,
PasswordString, PasswordNumber
```

例2 (对C 系列PLC不设置保护)

```
Dim SetProtection
Dim PasswordString
Dim PasswordNumber
EnableProtection = false
PasswordString = ""
PasswordNumber = 12345678
CXServer.Protect       "PLC1",       EnableProtection,
PasswordString, PasswordNumber
```

此命令的参数依次为:

- PLC - PLC的名称.
- EnableProtection - 值为真设置密码保护，为假则不设置。
- PasswordString - 密码是字符串。对于CS系列PLCs，字符串至多有8个字符。对于CV PLCs，字符串至多有8个字符，并包括十六进制数，如，"12345678". 对于C 系列 PLCs，字符串至多有4个字符，包括一个十六进制数，如 "1234".
- PasswordNumber - 当前仅用于C和CV系列PLCs,并且仅当密码字符串空白是使用。在这些情况下，它仅是一个代表4或8位密码的值的数字。请注意密码以十六进制字符串输入CX-Programmer， (如同上面的PasswordString参数), 比如说，十进制的值1234相当于十六进制的"04d2"密码字符串。

另外C系列PLC 注意:对于C系列，PLC程序需要代码（应用程序第一行） ，以能够进行密码设置/发表，这也与密码值相适应。

```
e.g.  LD AR10.01
FUN49 0 0 #1234  (#1234 - password value in Hex)
```

当设定密码时，这个值被使用而非被传递，-即无需在意密码字符串或数字。但当禁用密码保护时，必须要提供正确的密码。

B.30 LastErrorString

此属性可以被设置或读出，是最后一次发生的错误的文本描述。如果没有错误发生，将显示空白。

例如:

```
txtError = CXServer.LastErrorString
CXServer.LastErrorString = ""
```

Appendix C JScript 功能

本附录提供可用于ExecuteJScript 和 ExecuteJScriptFile脚本函数的JScript功能的摘要。这些功能由Windows 脚本宿主提供，Windows 98, Windows ME, Windows 2000 和 Windows XP 都默认包括Windows脚本宿主，也可以从 Internet Explorer 4.0及以上版本安装。对于Windows 95和Windows NT, Windows脚本宿主可从 Microsoft的网站上自行免费下载。

欲了解最新版本的详细信息与获取支持，请联系Microsoft，网址为http://msdn.microsoft.com/scripting

种类	关键词/功能
Array Handing	Array join, length, reverse, sort
Assignments	Assign (=) Compound Assign (OP=)
Booleans	Boolean
Comments	/*…*/ or //
Constants / Literals	NaN null true, false Infinity undefined
Control flow	break continue for for..in if…else return while
Dates and Time	Date getDate, getDay, getFullYear, getHours, getMilliseconds, getMinutes, getMonth, getSeconds, getTime, getTimezoneOffset, getYear, getUTCDate, getUTCDay, getUTCFullYear, getUTCHours, getUTCMilliseconds, getUTCMinutes, getUTCMonth, getUTCSeconds, setDate, setFullYear, setHours, setMilliseconds, setMinutes, setMonth, setSeconds, setTime, setYear, setUTCDate, setUTCFullYear, setUTCHours, setUTCmillisecinds, setUTCMinutes, setUTCMonth, setUTCSeconds, toGMTString, toLocaleString, toUTCString, parse, UTC
Declarations	function new this var with
Function Creation	Function arguments, length

种类	关键词/功能
Global Methods	Global escape, unescape eval isFinite, isNaN parseInt, parseFloat
Maths	Math abs, acos, asin, atan, atan2, ceil, cos, exp, floor, log, max, min, pow, random, round, sin, sqrt, tan, E, LN2, LN10, LOG2E, LOG10E, PI, SQRT1_2, SQRT2
Numbers	Number MAX_VALUE, MIN_VALUE NaN NEGATIVE_INFINITY, POSITIVE_INFINITY
Object Creation	Object new constructor, prototype, toString, valueOf
Operators	Addition(+), Subtraction (-) Modulus arithmetic (%) Multiplication (*), Division (/) Negation (-) Equality (==), Inequality (!=) Less Than (<), Less Than or Equal To (<=) Greater Than (>) Greater Than or Equal To (>=) Logical And (&&), Or (||), Not (!) Bitwise And (&), Or (|), Not (~), Xor (^) Bitwise Left Shift (<<), Shift Right (>>) Unsigned Shift Right (>>>) Conditional (?:) Comma (,) delete, typeof, void Decrement (--), Increment (++)
Objects	Array Boolean Date Function Global Math Number Object String
Strings	String charAt, charCodeAt, fromCharCode indexOf, lastIndexOf split toLowerCase, toUpperCase length

附录D 淘汰功能

本附录提供一些已经淘汰以及从标准文档中删除的功能的摘要。以下将介绍详细信息，以帮助用户仍然使用这些功能维护旧的项目。这些功能不用于开发新的解决方法，因为它们能够支持的下列功能也许在下一个或将来的版本中删除。

D.1 Windows NT, Windows ME, Windows 98 and Windows 95

这款产品将不会安装在这些操作系统上，推荐更新Windows 版本。

D.2 Sleep

描述

指定的一段时间内暂停脚本的执行。

语法

```
Sleep (duration)
```

备注

参数	类型	描述
Duration	- - -	继续之前等待的毫秒数。

例如

```
Sleep (1000)
```

CX-Supervisor等待 1 秒钟。

注意: 应谨慎使用睡眠语句，因为当脚本睡眠时，系统的其他一些部分可能会停止更新。该语句也使用多线程处理，这意味着PLC通信类的任务会同时发生，结果难以预料。

注意: 在完善的设计中，不应该要求 Sleep() 语句的事件驱动系统的使用。总是要思考，当一个条件发生时，睡眠之后的语句是否应该在它们自己的脚本内执行。

注意: 操作系统件的间隔各不相同。Windows NT (和 2000) 期限每10ms被检查一次，所以 'Sleep(100)'实际上是否能够暂停100-109.99之间的任何一段时间，取决于该语句什么时候开始。Windows 98 (和ME)的间隔是55ms，所以 'Sleep(100)'实际上暂停了110 (2 乘以 55)到164.99 毫秒(将近3 乘以55)。因此，睡眠语句在不同的使应用OS独立的操作系统上，也是表现不同的。

作为一个拖慢时间进程的语句，睡眠语句应放弃使用，原因如下:

注意:

- 实际的时间延迟取决于上面描述的OS
- 总存在0到1间隔的错误，取决于动作开始时间。
- 无法保证频率，因为OS也许会忙碌或处理其他进程。

D.3 DDE Commands

DDE作为交换数据的方式，已经被淘汰多年了。实际上，它的继承者OLE自动化也已成为往事。DDE已被证明是一种落后的科技，在本地操作系统和Microsoft Excel等工具方面，都存在无法修复的存储漏洞。现在，我们应该使用CX-Supervisor通信控制以取代这一技术。

下述DDE脚本命令已被淘汰，不再使用。

D.3.1 DDEExecute

语法

```
returnstate = DDEExecute(channel, {command})
```

备注

参数	类型	描述
returnstate	布尔型	若函数成立则返回状态为1，否则为0。
channel	整数点	包含DDEInitiate()命令的返回值的整数点。在DDEInitiate()命令下，应用于channel的服务器和主题参数必须打开或报错。
command	字符串	被 channel中指定的服务器应用程序承认的命令。

例如

```
channelname = DDEInitiate("Excel", "Sheet1.xls")
DDEExecute(channelname,
{[OPEN("C:\EXCEL\WORK\SHEET2.XLS")]})
```

由整数点'channelname'指定，路径 'C:\EXCEL\WORK' 中的文件'SHEET2.XLS'在Microsoft Excel中打开。 文件'SHEET1.XLS'已经在 Microsoft Excel中打开了。

D.3.2 DDEInitiate

语法

```
channel = DDEInitiate("server", topic")
```

备注

参数	类型	描述
channel	整数点	包含 DDEInitiate()命令返回值的整数点。
server	字符串	作为DDE服务器，包含支持DDE的应用程序。通常，这是应用的*.EXE可执行文件，该文件没有文件扩展名。 运行时，服务器应用必须打开，否则无法返回值或出现报错。

参数	类型	描述
topic	字符串	包括被服务器应用程序承认的主题的名称。 通常,一个主题是一个应用里的一份文件。 在运行时，主题必须打开，否则无法返回值或出现报错。主题可以空白，这允许在进行特定连接之前远程打开文件。主题名称 'System' 可用来查找出服务器应用中其他可用的主题。不过，是否支持这一主题取决于服务器应用。

例如

```
channelname = DDEInitiate("Excel", "Sheet1.xls")
```

整数点'channelname'有可以链接到应用程序Microsoft Excel 的DDE链接，该应用程序有可执行文件名称'EXCEL.EXE'运行， 也可链接到Excel中的'SHEET1.XLS' 。

D.3.3 DDEOpenLinks

语法

```
returnstate = DDEOpenLinks(channel)
```

备注

参数	类型	描述
returnstate	布尔型	若函数成立则返回状态为1，否则为0。
channel	整数点	包含DDEInitiate()命令的返回值的整数点。在DDEInitiate()命令下，应用于channel的服务器和主题参数必须打开或报错。

例如

```
channelname = DDEInitiate("Excel", "Sheet1.xls")
DDEOpenLinks(channelname)
```

DDEOpenLinks命令允许点开始数据传输，这些点已被配置，可以通过DDE进行通信。和应用程序 Microsoft Excel之间的数据传输可以自动维持，直到由Microsoft Excel或使用整数点'channelname'的DDETerminate()命令 或DDETerminateAll()命令关闭。

D.3.4 DDEPoke

语法

```
returnstate = DDEPoke(channel, "item", pointname)
```

备注

参数	类型	描述
returnstate	布尔型	若函数成立则返回状态为1，否则为0。

参数	类型	描述
channel	整数点	包含DDEInitiate()命令的返回值的整数点。在DDEInitiate()命令下，应用于channel的服务器和主题参数必须打开或报错。
item	字符串	被服务器应用承认的项。比如，在一个电子表格程序中，单元格就是一个项。同样地，一个页面是word 进行应用中的项。它完全依赖服务器。
pointname	点	属性中必须包括'Read/Only' 或'Read/Write'的DDE访问的点。该点的内容被赋予服务器应用程序。

例如

```
channelname = DDEInitiate("Excel", "Sheet1.xls")
DDEPoke(channelname, "R2C5", data)
```

点'data'的内容被发送至Microsoft应用中的'SHEET1.XLS'的2行5列。Microsoft Excel 应用程序和 'SHEET1.XLS'由整数点 'channelname'指定。

D.3.5 DDERequest

语法

```
pointname = DDERequest(channel, "item")
```

备注

参数	类型	描述
channel	整数点	包含DDEInitiate()命令的返回值的整数点。在DDEInitiate()命令下，应用于channel的服务器和主题参数必须打开或报错。
item	字符串	被服务器应用承认的项。比如，在一个电子表格程序中，单元格就是一个项。同样地，一个页面是word 进行应用中的项。它完全依赖服务器。
pointname	点	属性中必须包括'Read/Write'的DDE访问的点。

例如

```
channelname = DDEInitiate("Excel", "Sheet1.xls")
cellref = DDERequest("channelname", "R2C5")
```

点'cellref'被一个特定项填充，该特定项为Microsoft Excel应用中的'SHEET1.XLS' 2行5列的内容，该项由实数点'channelname'指定。

D.3.6 DDETerminate

语法

```
returnstate = DDETerminate(channel)
```

备注

参数	类型	描述
returnstate	布尔型	若函数成立则返回状态为1，否则为0。
channel	整数点	这是一个包含DDEInitiate（）命令的返回值的整数点。 应用于DDEInitiate（）命令渠道的服务器和主题参数都必须打开或报告错误。

例如

```
DDETerminate(channelname)
```

整数点 'channelname'指定的服务器和主题被关闭。

D.3.7 DDETerminateAll

语法

```
returnstate = DDETerminateAll()
```

备注

参数	类型	描述
returnstate	布尔型	若函数成立则返回状态为1，否则为0。

例如

```
DDETerminateAll()
```

所有先前启动的DDE链接都将被关闭。

D.3.8 EnableDDE

语法

```
returnstate = EnableDDE(pointname)
```

备注

参数	类型	描述
returnstate	布尔型	若函数成立则返回状态为1，否则为0。
Pointname	布尔点	保存所需的启用/禁用状态的布尔点。

例如

```
EnableDDE(result)
```

基于点‘result’的值启用DDE函数。 如果'point'为'TRUE'，则启用DDE，如果'point'为'FALSE'，则禁用DDE。

```
EnableDDE(TRUE)
```

DDE函数也可以在不使用点的情况下直接使用以保持所需的状态。

D.4 图表命令

D.4.1 ClearGraph

语法

```
returnstate = ClearGraph("graphid", "pagename")
```

备注

参数	类型	描述
returnstate	布尔型	若函数成立则返回状态为1，否则为0。
graphid	字符串	要清除的趋势图或散点图的标识符。
pagename	string	可选参数，指示图形所在页面的名称。

例如

```
ClearGraph("Graph_1", "TestPage1")
```

具有标识符'Graph_1'的'TestPage1'上的趋势图或散点图已清除其数据。

```
ClearGraph ("Graph_2")
```

当前页面上的趋势图或散点图，标识符为'Graph_2'，其数据已清除。

D.4.2 StartGraph

语法

returnstate = StartGraph("graphid", "pagename")

备注

参数	类型	描述
returnstate	布尔型	若函数成立则返回状态为1，否则为0。
graphid	字符串	要开始的趋势图或散点图的标识符。
pagename	字符串	可选参数，指示图形所在页面的名称。

例如

```
StartGraph("Graph_1", "TestPage1")
```

具有标识符'Graph_1'的'TestPage1'上的趋势图或散点图使其数据记录开始。

```
StartGraph("Graph_2")
```

当前页面上具有标识符'Graph_2'的趋势图或散点图已启动其数据记录。

Note: 提供此命令以与SCS v2.0应用程序兼容。 对于较新的应用程序，应优先使用数据记录工具。

D.4.3 StopGraph

语法

```
returnstate = StopGraph("graphid", "pagename")
```

备注

参数	类型	描述
returnstate	布尔型	若函数成立则返回状态为1，否则为0。
graphid	字符串	要停止的趋势图或散点图的标识符。
pagename	字符串	可选参数，指示图形所在页面的名称。

例如

```
StopGraph("Graph_1", "TestPage1")
```

具有标识符'Graph_1'的'TestPage1'上的趋势图或散点图的数据记录已停止。

```
StopGraph("Graph_2")
```

当前页面上具有标识符"Graph_2"的趋势图或散点图的数据记录已停止。

D.4.4 EditGraph

语法

```
returnstate = EditGraph("graphid")
```

备注

参数	类型	描述
returnstate	布尔型	若函数成立则返回状态为1，否则为0。
graphid	字符串	要编辑的趋势图或散点图的标识符。

例如

```
EditGraph("Graph_1")
```

将显示"编辑图形"对话框，提供用于查看所选趋势图的历史数据的选项。

- 显示数据将当前选择的数据样本，即当前屏幕数据或数据快照加载到趋势图中。
- 快照存储与趋势图关联的当前数据缓冲区。 快照具有带时间戳的默认描述。
- 描述提供更改与快照关联的描述的功能。
- 导入数据提供在以前保存的趋势图文件中加载的能力。
- "导出数据"提供以内部CX-Supervisor格式将快照存储到文件的能力，或作为可导入到其他应用程序的文本文件。
- 删除将移除当前选定的快照。

注意: 提供此命令是以与SCS v2.0应用程序兼容。
对于较新的应用程序，应优先使用数据记录工具。

注意: 仅当趋向图设置为记录到文件时，才能使用此命令。

D.4.5 SaveGraph

语法

```
returnstate = SaveGraph("graphid")
```

备注

参数	类型	描述
returnstate	布尔型	若函数成立则返回状态为1，否则为0。
graphid	字符串	要保存的趋势图或散点图的标识符。
pagename	字符串	可选参数，指示图形所在页面的名称。

例如

```
SaveGraph("Graph_1", "TestPage1")
```

具有标识符‘Graph_1’的页面‘TestPage’上的趋势图将其数据保存到光盘。

```
SaveGraph("Graph_2")
```

当前页上的趋势图'Graph1'中的当前数据被存储，并且能够通过EditGraph命令查看。

D.4.6 Snapshot

语法

```
returnstate = Snapshot("graphid", "pagename")
```

备注

参数	类型	描述
returnstate	布尔型	若函数成立则返回状态为1，否则为0。
graphid	字符串	具有快照的趋势图或散点图的标识符。
pagename	字符串	可选参数，指示图形所在页面的名称。

例如

```
Snapshot("Graph_1", "TestPage1")
```

'TestPage1'的趋势图'Graph1'中的当前数据被存储，并且能够通过EditGraph命令查看。

```
Snapshot("Graph_2")
```

当前页上的趋势图'Graph1'中的当前数据被存储，并且能够通过EditGraph命令查看。

注意: 提供此命令以与SCS v2.0应用程序兼容。 对于较新的应用程序，应优先使用数据记录工具。

D.4.7 GetPointValue

语法

```
returnpoint = GetPointValue(pointname,offset)
```

备注

参数	类型	描述
pointname	点	要返回其内容的点的名称
offset	整数	指定了到数组点的offset。 如果点不是数组点，则为0。
returnpoint	点	包含返回值的点。 返回的数据类型取决于指定的pointname。

例如

```
pointname = 10;
returnpoint = GetPointValue(pointname,0)
```

点'returnpoint'包含值10。偏移量添加到为pointname指定的任何偏移量。 例如：

```
returnpoint = GetPointValue(a[10],10)
```

导致要检索数组'a'的第21个元素（偏移从零开始）。

注意： 通常直接访问数组元素更简单，例如 returnpoint = a [20]。

D.4.8 GetSpoolCount

语法

```
returnstate = GetSpoolCount()
```

备注

参数	类型	描述
returnstate	整数	等待在警报/消息打印机上打印的排队等待的消息数。

例如

```
NumberMessages = GetSpoolCount()
```

返回排队等待发送到CX　Supervisor报警/消息打印机的消息数（通常为打印报警）的计数。

D.4.9 SetPrinterConfig

语法

```
returnstate StePrintConfig(Driver, Device, Port)
```

备注

参数	类型	描述
returnstate	布尔型	若函数成立则返回状态为1，否则为0。
Driver	字符串	打印机设备名称（例如，“Epson9”用于9针Epson打印机。
Device	字符串	特定设备的名称（例如“Epson FX-870”）。 这是可选的。
Port	字符串	端口或文件的名称（例如“LPT1”）。
Line Terminator	字符串	可选。 设置要添加到每条打印行末尾的终结符（例如cr）。

例如

```
SetPrinterConfig("SCSPRN", "", "LPT1:")
```

This uses standard CX-Supervisor line print driver.

```
SetPrinterConfig("", "", "")
```

This uses default Windows printer driver.

```
SetPrinterConfig("Epson9", "", "LPT2:")
```

This uses Epson printer driver, attached to LPT2.

```
SetPrinterConfig(DriverNamePoint,     DeviceNamePoint,
PrintNamePoint)
```

This uses text points.

```
Terminator = FormatText("%c%c",13,10)
```

Character 10 is 'lf' (newline), character 13 is cr (carriage return).

```
SetPrinterConfig("Epson9","","LPT1:",Terminator)
```

附录E
术语表

ADO	ADO 代表活动数据对象，是使用OLE-DB通过统一的方式来访问资源的数据访问技术。如：MS-Access数据库，MS-Excel电子表格和逗号分隔变量文件。
AND	逻辑运算符，用于询问布尔型点。若所有的命令判断结果都为“是”，则AND执行结果将为“是”。一个AND的例子是，假设a和b都是一个语句，若两者结果都为“是”，则AND的执行结果为“是”，若其中一个或两者都为“否”，则AND的执行结果为“否”。
应用程序	以完成特定的任务的软件程序。应用程序的示例如CX-Supervisor,CX-server和Microsoft Excel。CX-Supervisor及其开发环境允许新的应用程序通过图形用户界面（GUI）创建和测试。
参数	单词，短语或数字可以作为命令或语句输入到CX-Supervisor的同一行脚本语言中，以扩大或修正命令或语句。命令将作用于参数。本质上，命令是个动词，而参数是动词的宾语。示例如CX-Supervisor的参数“DDETerminate(channel)”中，DDETerminate是脚本语言的命令，channel是命令将作用的参数。
ASCII	定义一组字符的旧标准。正式使用只有7位二进制数，以允许定义的只有127个字符，并且不包括任何重音。
位图	图像存储在计算机内存中的表示形式。每张图片元素（像素）都以二进制形式存储在内存中。在CX-Supervisor中，一张位图可以被安装为单个对象。
布尔型	当点的值可以为两种状态之一时该点的类型。本质上两种状态是“0”和“1”，但是这些状态可以被赋值为指定意义。示例如下：

状态	示例	示例	示例	示例
0	OFF	FALSE	OUT	CLOSED
1	ON	TRUE	IN	OPEN

COM	COM是一项Microsoft技术，允许组件交互使用。

通信驱动程序	欧姆龙PLCs连同Microsoft Windows的相关通信管理系统，为其他SYSMAC软件提供维护以维护PLC设备和地址信息，以及与欧姆龙PLCs和其他所支持的网络类型沟通。
常量	在CX-Supervisor中，常量是脚本语言中的一个点，只有一个特定的值。
控件对象	在CX-Supervisor中，控件对象用于开发环境，可作为按钮、切换按钮、滑块、趋势图、旋转量表或线形量表。本质上，控件对象也可以是更复杂的由许多图形对象而组成的图形对象，用于用户互动。
CX-Server	欧姆龙PLCs用于提供软件维护以保护PLCs设备和地址信息，以及与欧姆龙PLCs其他所支持的网络类型通信的先进通信管理系统。
数据库连接	数据库连接（或简称为连接）包含了用于访问数据源的详细信息，也可以通过数据源名称（DSN），文件名或目录。
数据库连接级别	数据库连接级别是一个字符串，它决定数据库数层结构中要运行哪些级别。示例列举如下：

"Northwind"	连接层级
"CSV.Result"	记录层级
"Northwind.Order Details.OrderID"	字段层级
"Invoice.Data Types"	图式层级

数据库记录集	数据库记录集（或简称记录集）是一组记录结果。它可以是数据库中实际的表格，或是已生成的表格运行查询的结果。
数据库图式	数据库图式（或简称图式）从供应商处获取数据库图式信息。
数据库服务器查询	数据库服务器查询（或简称服务器查询）是存储在实际数据库中的查询。它们通过数据库设计器被预定义和添加，意味着它们是“固定”在项目的持续时间中。服务器查询可能有预定义的“参数”，在允许标准在运行时传递给查询项，例如作为筛选，以产生不同的查询结果。每一个预定义的参数都必须有定义的关联参数。因为这些查询存储在已编译和测试过的形式中，更高效，所以运行SQL查询会更优越。

SQL数据库查询	SQL数据库查询（或简称SQL查询）是在运行时进行动态翻译。SQL文本可以在运行时被修改，以满足不同的查询能在多种情况下运行，然而SQL文本每次执行时必须被快速编译，因此它的效率低于服务器查询。
DBCS	DBCS代表双字节字符集，是微软使用2字节（16位）来定义字符代码的ASCII的扩展程序。它的范围更大，可以包含重音字符，扩展了ASCII的的字符、北欧字符和符号。
DCOM	DCOM是COM的分布式版本，允许不同电脑的组建通过网络进行交互。
DDE	动态数据交换。通过准备正常的程序通道动态交换数据，以及控制Microsoft Windows中的其他应用程序。DDE技术是出了名的不稳定，因此被替换为OLE技术。 请参见项目、服务器、服务器应用程序和主题。
开发环境	SCADA应用程序是使用CX-Supervisor的开发环境中创建并测试的。结束后，完成的应用程序将作为最终客户应用程序交付到运行环境中运行。
DLL	动态链接库。虽然不能作为可执行文件独立运行一个程序文件，但可以通过一或多个应用程序或程序作为公共服务使用。DLL文件后缀*.DLL扩展名。DLL文件由许多独立的功能组成。在CX-Supervisor中，DLL包含了可访问的图标代表OLE对象的显示部分。对于“MORICONS.DLL”这样的DLL文件，在标准的Microsoft Windows安装中有所提供。
下载	配方是在运行时下载的。这个过程包括了确定合适的配方以及如果有验证代码时，执行验证代码。当所有成分点完成目标值设置，则下载完成。
可执行	一个包含应用程序的程序和命令的文件可以通过一个用户或另一个应用程序执行。可执行文件有*.EXE文件扩展名。CX-Supervisor提供了两种可执行文件，一种针对开发环境（CXSUPERVISORDEV.EXE），另一种针对运行环境（SCS.EXE）。
表达式	在CX-Supervisor脚本语言中，表达式是计算一个或多个操作数的值的构造。例如，在示例“lift=height+ rate”中，表达式为“height+ rate”，则该表达式生成的结果将用于“lift”的值。

	除脚本语言外，表达式由运算符和操作数组成，可通过动作应用于控制对象。
字段关联	字段关联可以在CX-Supervisor Point和记录集内的特定字段（例如，列）之间建立链接。
图形对象	在CX-Supervisor中，图形对象在开发环境中创建，可以是直线、弧线、多边形（包括正方形和长方形）、圆角矩形、椭圆（包括正圆）或折线。一个复杂的对象可以以两个或多个图形对象的组合形式存在。
GUI	图形用户界面。是用于与用户进行交互，并充分利用计算机的图形显示优势的程序的一部分。GUI采用下拉式菜单和对话框以方便使用。像所有Microsoft Windows的基础应用程序一样，CX-Supervisor也有GUI。
I/O 类型	输入／输出类型。定义了点的原始和目标数据的点的属性。点的数据可以来源于（从哪里输入）和目标到（输出到）内部计算机的内存或PLC。
图标	图像表征的计算机资源和功能。CX-Supervisor开发环境和运行环境都是从图标运行的。
成分	每个配方都包含至少一个成分。每个成分必须与现有的点相关联。
整数型	一种点的类型，该点的值只能是整的正数或负数。
项	在CX-Supervisor脚本语言中，项是一个点、OPC项或温度控制器项目的通用名称。
JScript	Microsoft Windows脚本宿主所支持的Java风格的脚本语言。
JVM	Java虚拟机。
Microsoft Excel	电子表格应用程序。
Microsoft Windows	针对其GUI，例如多种字体、桌面附件（如时钟、计算器、日历和记事本）的特征，以及通过剪贴板从一个应用程序到另一个移动文本和图形能力的窗口环境。CX-Supervisor只会在Microsoft Windows下运行。DDE函数和其他由CX-Supervisor所支持的应用程序的通信功能也同样以Microsoft Windows为基础。
Microsoft Word for Windows	文字处理应用程序。

嵌套	将一个或多个IF THEN ELSE／ELSEIF ENDIF语句包含进相同种类的结构里。
网络	PLC配置基于设备类型的一部分。可用网络的数量取决于设备类型。 许多计算机链接在一个能够访问所有计算机，叫做服务器的中央处理点上。网络影响CX-Supervisor，如果计算机网络已连接，进一步网络关联选项即可用。
非易失性	定为“非易失性”的点是指其值被保存在磁盘上并在CX-Supervisor恢复运行时自动重新加载。
NOT	逻辑运算符，用于审问产生逆所给参数的布尔值的布尔型点。一个NOT的示例为，如果有一个语句且结果为“FLASE”，则NOT返回结果为“TRUE”。如果一个语句且结果为“TRUE”，则NOT返回结果为“FLASE”。
对象	在CX-Supervisor中，一个对象可以是文本、图形、控件、位图或是ActiveX对象，在开发环境中所创建的。一个复杂的对象可以由两个或多个以上类型的对象组成。具体来说，图形对象可以由直线、弧线、多边形（包括正方形和长方形）、圆角矩形、椭圆（包括正圆）或折线组成。一个控件本质上是一个复杂的图形对象，具体可以是一个按键、切换键、滑块、趋势图、旋转量表或线性量表。
OLE-DB	OLE-DB是ADO所依赖的最底层的数据库技术。OLE-DB被设计为ODBC的继任。
操作数	用于常量或变量的术语。
运算符	作为一项函数使用，当有两个命令（如“+”）时使用中置语法，只有一个命令（如NOT）时使用前置语法。CX-Supervisor脚本语言在内置功能中使用运算符，如算数运算符和逻辑运算符。
OR	逻辑运算符，用于审问布尔型点。若任一给出命令为“TRUE”，则OR返回结果为“TRUE”。一个OR语句的示例为，如果a和b都是一个语句，若a或b其中一个为“TRUE”，则 OR返回结果为“TRUE”。若两个语句结果都为“FLASE”，则OR返回结果为“FLASE”。
页面	页面的组合和操作包含了项目中构成CX-Supervisor基础的对象。每个项目中可以存在多个页面。项目中的页面提供了CX-Supervisor中的可视部分，与包含在每个页面中用于提供检测系统的图形表示的对象相对应显示。

参数关联	参数关联可使常量值或存储在点中的值被传输到服务器查询。
像素	在屏幕上构成显示图像的一个单独可显示的点。计算机视觉显示单元（VDU）的屏幕分辨率由像素纵向和横向的数量（例如1024*768）决定。 另请参阅SVGA模式和VGA模式。
PLC	可编程逻辑控制器。
点变量	在CX-Supervisor脚本语言中的点，存储值或分配到该点的字符串。
点	存储一个预定义类型－布尔型、整型、文本等类型的值的点。点的内容可以由对象或I/O机制，例如PLC通信控制。点的内容可以控制对象的动作或外观，或通过I/O机制用于输出。 另请参阅布尔型、整型、点变量、实数类型和文本类型。
项目	CX-Supervisor应用程序由一个或多个页面相互链接组成。页面中可能包含被动或主动的图形、文本或动画，也可能将它们按逻辑分组在一起以形成一个项目。一个项目可能由多个页面组成，或只是单个页面。项目可以在CX-Supervisor开发环境中创建及测试，并且在CX-Supervisor运行环境下独立运作。 在CX-Supervisor开发环境中一次只能打开一个项目以编辑。
实数类型	值可以是任何数字，包括含有小数点的数字的点。
配方	配方是一套预定义的步骤以执行特定任务。CX-Supervisor项目可能没有或包含多个配方。配方在开发环境中被定义及执行，或在运行环境中下载。
运行环境	SCADA应用程序使用CX-Supervisor的运行环境来运作，接在CX-Supervisor开发环境中创建应用程序以后。
SCADA	监督控制和数据采集。
服务器	服务器是网络可访问所有计算机的中央处理点。当网络已连接，网络影响CX-Supervisor使进一步相关选项可用。
服务器应用程序	可用于查看或交互的应用程序，目前存在在CX-Supervisor中。

语句	在CX-Supervisor脚本语言中，语句是运行环境所理解的指令。语句由命令和参数构成，当它们组合在一起时，有助于明确表达一个在运行环境中所使用的完整的应用程序
字符串	只包含了字母数字字符的文本类型点的内容。一个字符串在左引号后开始，在右引号前结束，例如“name=“spot””中，点“name”包含了字符串spot。
SVGA模式	800*600像素分辨率（或更高），16或更多种颜色并在超级视频图形适配器系统上所支持的视频显示。
CX-Supervisor	SCADA软件应用程序，创建并维护图形用户界面，并与PLC和其他I/O机制通信。
目标值	必须为相关点指定目标值的成分。是在菜单下载完成后在运行环境中将要被设置的值。
任务栏	Microsoft Windows允许其基础应用程序被启用的组成部分。CX-Supervisor便是从任务栏运行的。
文本对象	在CX-Supervisor中，文本对象是一个页面上的字符串。例如字体、字号、加粗、斜体、下划线、左对齐、右对齐、居中等属性可用于加强文本对象展示效果。
文本类型	包含字符串的点的一种类型。
Unicode	一种多字节字符集，不仅包含欧洲字符，例如DBCS，也包含了全球支持，例如日文、中文以及斯拉夫字体。然而，Unicode在所有的Windows平台上不被支持。
验证代码	配方验证代码是CX-Supervisor脚本语言，用于在下载配方前检查点的值。
VBScript	由Microsoft Windows脚本宿主所支持的Visual Basic类型的脚本语言。
VGA 模式	640*480像素分辨率，16色，并被视频图形适配器系统所支持的视频显示模式。
Windows 桌面	Microsoft Windows允许其基础应用从图标启动且可用于整理组织所有应用程序的部分。 CX-Supervisor可从Windows 桌面上被运行。
Windows 脚本宿主	由微软提供用于运行VBScript或Jscript的脚本引擎。参见http://msdn.microsoft.com/scripting

向导	向导是CX-Supervisor开发环境用于带领用户在简化的分步过程中使用复杂操作的对话。

修订记录

手册修订代码将在手册的封面上显示为目录编号的后缀。

Cat. No. W09E-EN-02

以下表格列出了每次手册修订中所做的更改。修订的页码指以前版本中的页码。

修订代码	日期	修订内容
01	2010年9月	标准欧姆龙格式中的第一个版本。
02	2011年6月	对于CX-Supervisor3.2版本的更新。

OMRON

授权分销商：

Cat. No. W09E-CN5-02

注意: 规格如有变更，恕不另行通知.

上海印刷