



Die Idee: Optimale Anpassbarkeit durch Scripte

Die Ausdrucksverarbeitung
STScript

20. Juni 2008

Inhalt

ALLGEMEINES	4
IEC 61131-3.....	4
ST SCRIPT.....	5
<i>Abweichungen vom Standard</i>	5
<i>Ausdrücke in Meldetexten</i>	5
SPRACHELEMENTE	6
ANWEISUNGEN	6
KOMMENTARE.....	6
ZUWEISUNGEN.....	6
OPERATOREN.....	7
<i>Rangfolge der Operatoren</i>	7
KONSTANTEN.....	8
<i>numerische Konstanten</i>	8
<i>Wahr/Falsch</i>	8
<i>Textuelle Konstanten</i>	8
<i>Datums- und Zeit-Konstanten</i>	9
<i>Aktuelles Datum und Uhrzeit</i>	9
EXTERNE WERTE	10
<i>OPC</i>	10
<i>SNMP</i>	11
<i>DDE</i>	11
DIGITALE EIN-/AUSGÄNGE	13
SYSTEMVARIABLEN	14
UMGEBUNGSVARIABLEN.....	14
DATUMSARITHMETIK	15
ADDITION.....	15
SUBTRAKTION.....	15
STANDARDFUNKTIONEN	16
ALLGEMEINE FUNKTIONEN	18
<i>VAR</i>	18
<i>CLRVAR</i>	18
<i>EVAL</i>	18
ARITHMETISCHE FUNKTIONEN	19
<i>SHL</i>	19
<i>SHR</i>	19
BEDINGTE AUSFÜHRUNG	20
<i>SEL</i>	20
<i>IF, THEN, ELSIF, END_IF</i>	20
STRINGVERARBEITUNG	21
<i>MID</i>	21
<i>LEN</i>	21
<i>LEFT</i>	21
<i>RIGHT</i>	22
<i>FIND</i>	22
<i>CONCAT</i>	22
<i>REPLACE</i>	23
<i>REPLACEALL</i>	23
<i>UPPERCASE</i>	23
<i>LOWERCASE</i>	23
<i>FORMAT</i>	24
<i>GETCSV</i>	26
<i>REGEXP</i>	27
DATENBANKFUNKTIONEN	28
<i>Datenbankverbindungen</i>	28
<i>DBFIELD</i>	29
<i>DBLOOKUP</i>	29

<i>DBREAD</i>	30
<i>DBWRITE</i>	31
<i>DBUPDATE</i>	32
<i>DBDELETE</i>	32
<i>DBEXEC</i>	32
<i>DBCOUNT</i>	33
<i>DBXML</i>	33
<i>DBTABLE</i>	34
<i>DBCLOSE</i>	34
ZUGRIFF AUF INI-DATEIEN.....	35
<i>INILOOKUP</i>	35
<i>INIWRITE</i>	35
ALLGEMEINE DATEIOPERATIONEN.....	36
<i>FILETEXT</i>	36
NETZWERKFUNKTIONEN.....	37
<i>PING</i>	37
<i>WAKEONLAN</i>	38
DIGITALE EIN-/AUSGABE.....	41
<i>QIMPULSE</i>	41
AUSFÜHRUNG EXTERNER BEFEHLE.....	42
<i>EXECUTE</i>	42
<i>DDEEXEC</i>	42
ZUGRIFF AUF DIE INFORMEL KONFIGURATION.....	43
<i>CFGLOGIN</i>	43
<i>CFG</i>	43
<i>CFGADDITEM</i>	43
<i>CFGREMITEM</i>	44
REGULÄRE AUSDRÜCKE	45
ANWENDUNGSSPEZIFISCHE SYSTEMVARIABLEN	46

Allgemeines

In allen informel Produkten kommt eine Scriptverarbeitung zum Einsatz die auf internationale Standards in der Automatisierungstechnik basiert. Dies garantiert die Akzeptanz und die einfache Einarbeitung in diese Art der Verarbeitung.

IEC 61131-3

Die IEC 61131-3 ist der einzige internationale Standard für Programmiersprachen von speicher-programmierbaren Steuerungen. Sie harmonisiert die Art in der industrielle Steuerungen entworfen und programmiert werden. Ein Standardinterface erlaubt Personen mit unterschiedlichen Interessen und Kenntnissen unterschiedliche Teile während des Lebenszykluses des Produkts zu erstellen: Spezifikation, Design, Implementierung, Test, Installation und Wartung. Alle Teile haben eine gemeinsame Struktur und passen harmonisch zueinander. Der Standard beinhaltet die Definition der Schrittkettenprogrammierung (AS), die zur internen Organisation eines Programms dient, und vier austauschbare Sprachen: Anweisungsliste (AWL), Kontaktplan (KOP), Funktionsplan (FUP) und eine Hochsprache (Strukturierter Text, ST). Durch die Möglichkeit der Aufteilung in logische Einheiten, Modularisierung und moderne Software Technologien ist jedes Programm strukturiert. Teillösungen können jederzeit wieder verwendet werden, Fehler werden reduziert und die Produktivität der Entwicklung wird gesteigert.

IEC 61131-3 is the only global standard for industrial control programming. It harmonizes the way people design and operate industrial controls by standardizing the programming interface. A standard programming interface allows people with different backgrounds and skills to create different elements of a program during different stages of the software lifecycle: specification, design, implementation, testing, installation and maintenance. Yet all pieces adhere to a common structure and work together harmoniously. The standard includes the definition of the Sequential Function Chart (SFC) language, used to structure the internal organization of a program, and four inter-operable languages: Instruction List (IL), Ladder Diagram (LD), Function Block Diagram (FBD) and Structured Text (ST). Via decomposition into logical elements, due to modularization and modern software techniques, each program is structured, increasing its re-usability, reducing errors and increasing programming and user efficiency.

[PLCopen](#)

Der Standard ist in den USA, Deutschland, Japan und Westeuropa als nationale Norm übernommen (z.B. in Deutschland als DIN EN 61131-3).

Er definiert ein Programmiermodell und 5 Programmiersprachen:

Deutsche Bezeichnung	English Description
AWL Anweisungsliste	IL Instruction List
KOP Kontaktplan	LD Ladder Diagram
FUP Funktionsplan	FBD Function Block Diagram
ST Hochsprache	ST Structured Text
AS Schrittkettenprogrammierung	SFC Sequential Function Chart

Die Norm definiert also zum Beispiel die Anweisungen in AWL, so dass jede IEC 61131-programmierbare SPS in ein- und demselben AWL programmiert werden kann. In KOP werden die Kontakte und Spulen definiert, wie sie verknüpft werden können usw.

ST Script

Wir haben die Idee eines internationalen Standards und dessen weite Verbreitung zum Anlass genommen unsere Programme mit entsprechender Programmierbarkeit auszustatten. Die Norm beinhaltet leider keine Script-Sprache die für unsere Einsatzzwecke notwendig ist. Wir entschieden uns für den Einsatz einer angepassten Variante der Hochsprache ST. Diese ist insbesondere für kurze Anweisungen zum Zugriff auf externe Daten passend.

Abweichungen vom Standard

Gegenüber der Standard Hochsprache ST haben wir folgende Änderungen eingeführt:

- Keine Deklaration von Variablen (Scripting)
- Automatische Typkonvertierung
- Erweiterung um einige Funktionen zum Zugriff auf externe Daten
- Einbettbar in Meldungstexte
- Schreibgeschützte Systemvariablen

Alle informel Programme verwenden diese Script Sprache für die Angabe von Ausdrücken gemäß der hier beschriebenen Syntax. Die Syntax der Ausdrücke erfolgt weitestgehend gemäß IEC 61131-3 Strukturierter Text (ST). Im Gegensatz zu diesem wird hier ein Interpreter verwendet um die Ausdrücke zu bearbeiten. Wir sprechen deshalb hier von STScript.

Ausdrücke in Meldetexten

Eine Besonderheit ist, dass die Ausdrücke auch in Meldetexte eingefügt werden können. Um sie vom eigentlichen Text unterscheiden zu können ist es notwendig, dass sie in doppelte geschweifte Klammern '{{' und '}}' eingeschlossen werden. Texte können somit zur Laufzeit berechnete, dynamische Werte enthalten.

Ein Beispiel

Der externe OPC Wert soll innerhalb einer Meldung verwendet werden.

Angabe des Textes:

```
Die Kesseltemperatur beträgt {{OPC('Advantech.ADAM', 'Device1.Group1.Temp1')}} Grad
```

Bei der Ausgabe des Textes wird der eingebettete Ausdruck berechnet und das Ergebnis eingesetzt:

```
Die Kesseltemperatur beträgt 55 Grad
```

Sprachelemente

In STScript stehen folgende Sprachelemente zur Verfügung

- Anweisungen
- Kommentare
- Operatoren
- Konstanten
- externe Werte (OPC, DDE)
- Digitale Ein-/Ausgänge
- Systemvariablen
- Umgebungsvariablen
- Funktionen

Anweisungen

Ein Script besteht aus einem oder mehreren Anweisungen. Mehrere Anweisungen werden durch Strichpunkte voneinander getrennt.

Syntax:

```
Anweisung {; Anweisung}
```

Das Ergebnis eines Scripts ist immer das Ergebnis der letzten Anweisung.

Kommentare

Innerhalb eines Scripts können Kommentare eingefügt werden die durch Klammern und Sterne von den Anweisungen getrennt werden.

Syntax:

```
(* Kommentar *)
```

Zuweisungen

Variablen kann ohne vorige Deklaration ein beliebiger numerischer oder textueller Wert zugewiesen werden.

Syntax:

```
Variable := Wert
```

► Beispiele

```
Meldung := 'Störung Getriebe' (* textueller Wert *)  
Sollwert := 17 (* numerischer Wert *)
```

Operatoren

Die Ausdrücke können folgende Operatoren beinhalten.

Arithmetische Operatoren	+, -, *, /, MOD, ** (Potenzierung)
Boolesche Operatoren	AND, OR, XOR, NOT
Vergleichsoperatoren	=, > (GT), < (LT), >= (GE), <= (LE), <> (NE)

Außer diesen Operatoren kann eine Klammerung von Teilausdrücken erfolgen.

Hinweis

Um die Vergleichsoperatoren einfacher in XML Texten verwenden zu können ist wahlweise die Darstellung als **GT**, **LT**, **GE**, **LE** und **NE** möglich.

Rangfolge der Operatoren

Bei der Auswertung der Ausdrücke werden die Operatoren in folgender Rangfolge bearbeitet:

Rangstufe	Operator	Operation
1	()	Klammerung
2	**	Potenzierung
3	-	Negation
4	NOT	Komplement
5	*	Multiplikation
6	/	Division
7	MOD	Modulo
8	+	Addition
9	-	Subtraktion
10	<, >, >=, <=	Vergleich
11	=	Gleichheit
12	<>	Ungleichheit
13	AND, &	boolesches UND (bitweise Verknüpfung)
14	XOR	exklusives ODER (bitweise Verknüpfung)
15	OR	boolesches ODER (bitweise Verknüpfung)
16	:=	Zuweisung

Konstanten

numerische Konstanten

Die Ausdrücke können numerische Konstanten (Literele) in unterschiedlicher Kodierung enthalten:

► Beispiele

Ganzzahlige Konstanten:

```
5
10
```

Gleitkomma-Konstanten (mit oder ohne Exponent):

```
7.33
10.5E-2
```

Binäre Konstanten:

Basis 2, Binär:

```
2#1010 2#11111111
```

Basis 8, Oktal:

```
8#12 8#377
```

Basis 16, Hexadezimal:

```
16#a 16#FF
```

Wahr/Falsch

Außerdem können die booleschen Werte **TRUE** (=1) und **FALSE** (=0) direkt angegeben werden.

Textuelle Konstanten

Textkonstanten bestehen aus beliebigen in einfachen Anführungsstrichen eingeschlossen Zeichen.

► Beispiele

```
'kommend'
'gehend'
'Dies ist eine Textkonstante'
```

Steuerzeichen

Die Zeichenketten können folgende Steuerzeichen enthalten

Steuerzeichen	ASCII Code	Beschreibung
\$L oder \$I	0AH	Zeilenvorschub (Line Feed)
\$P oder \$p	0CH	Seitenvorschub (Form Feed)
\$R oder \$r	0DH	Wagenrücklauf (Carrige Return)
\$N oder \$n	0DH, 0AH	Neue Zeile (New Line, Wagenrücklauf und Zeilenvorschub)
\$'	27H	Einfaches Anführungszeichen

Datums- und Zeit-Konstanten

Textkonstanten bestehen aus beliebigen in einfachen Anführungsstrichen eingeschlossenen Zeichen.

Syntax:

`DT#Year-Month-Day-Hour:Minute:Second`

▶ Beispiele

`DT#1995-06-01-12:00:00`

`DT#2006-05-29-09:03:15`

Aktuelles Datum und Uhrzeit

Zur einfachen Angabe des momentanen Zeitpunkts steht die spezielle Datums/Zeitkonstante `DT#NOW` zur Verfügung.

Syntax:

`DT#NOW`

▶ Beispiele

Ausdruck:

`DT#NOW`

Ergebnis:

`DT#2006-05-29-09:05:51`

Externe Werte

In den Ausdrücken können Werte von externen Quellen verwendet werden. Diese müssen von der externen Software über die standardisierten Schnittstellen OPC, SNMP und DDE bereitgestellt werden.

Die Angabe der Werte erfolgt in wie ein Funktionsaufruf mit Parametern die den Server (die externe Software) und die Bezeichnung des Wertes (Tag, Item). Im Unterschied zu einem Funktionsaufruf können externen Werten auch neue Inhalte zugewiesen werden.

OPC

OPC steht für Openness, Productivity, Collaboration (vormals für: OLE for Process Control). Es handelt sich dabei um eine standardisierte Software-Schnittstelle, die es Anwendungen unterschiedlichster Hersteller ermöglichen soll, Daten auszutauschen.

[OPC Foundation](#)

Syntax:

```
OPC(Server, ItemPath)
```

Server Name des OPC Servers
ItemPath Voller Pfad des Wertes

★ Tipp

Sie können den von informel kostenlos zur Verfügung gestellten OPC-Browser verwenden um die vom Server zur Verfügung gestellten Werte anzuzeigen.

▶ Beispiele

Auslesen eines Wertes aus einer SIMATIC S7.

Ausdruck:

```
OPC('OPC.SimaticNET', 'S7:[CPU|DB|CP_L2_1:]E1')
```

Hierbei ist 'OPC.SimaticNET' der Name des Servers und 'S7:[CPU|DB|CP_L2_1:]E1' die Bezeichnung des Tags im Siemens OPC Server.

Zuweisung eines neuen Temperaturwertes.

Ausdruck:

```
OPC('Advantech.ADAM', 'Device1.Group1.Temp1') := 60;
```

SNMP

Das Simple Network Management Protocol (englisch für "einfaches Netzwerkverwaltungsprotokoll", kurz SNMP), ist ein Netzwerkprotokoll, das von der IETF definiert wurden. Es wurde entwickelt um Netzwerkelemente (Router, Server, Switches, Drucker, Computer usw.) von einer zentralen Managementstation aus überwachen und steuern zu können.

⚠ Achtung
SNMP Werte können nur gelesen werden.

Syntax:

```
SNMP ( Server, OID )
```

Server Name des Servers

OID Object ID (OID) des zu lesenden Wertes

★ Tipp

Sie können den von informel kostenlos zur Verfügung gestellten SNMP-Browser verwenden um die vom Server zur Verfügung gestellten Werte anzuzeigen.

▶ Beispiel

Auslesen der UpTime (Laufzeit seit Neustart) eines Servers.

Ausdruck:

```
SNMP ('10.1.1.254', '.iso.org.dod.internet.mgmt.mib-2.system.sysUpTime.0')
```

Hierbei ist '10.1.1.254' die IP Adresse des Servers und '.iso.org.dod.internet.mgmt.mib-2.system.sysUpTime.0' die OID der UpTime des Server.

Um den Zugriff auf den Standardzweig mib-2 zu vereinfachen steht die Konstante MIB2 zur Verfügung. Diese enthält den Pfad zu diesem Zweig.

Syntax:

```
MIB2
```

Wert:

```
'.iso.org.dod.internet.mgmt.mib-2.'
```

▶ Beispiel

Ausdruck:

```
SNMP ('10.1.1.254', CONCAT (MIB2, 'system.sysUpTime.0'))
```

DDE

Dynamic Data Exchange (engl., Abk. DDE) bedeutet dynamischer Datenaustausch. Es handelt sich um ein Protokoll für den Datenaustausch zwischen verschiedenen Anwendungsprogrammen. Dieses Protokoll in den Betriebssystemen Windows (ab Version 3.0) verfügbar.

Syntax:

```
DDE (Application, Topic, Item)
```

Application Name der DDE Anwendung (Server)

Topic Bezeichnung des DDE Themas

Item Bezeichnung des DDE Wertes

★ Tipp

Sie können den von informel kostenlos zur Verfügung gestellten DDE-Browser verwenden um die vom Server zur Verfügung gestellten Werte anzuzeigen.

▶ Beispiel

Übernahme einer Excel Zelle (Zeile 1, Spalte 1).

Ausdruck:

```
DDE('Excel','Mappel','Z1S1')
```

Digitale Ein-/Ausgänge

Zur Steuerung externer Geräte oder Übernahme externer Signale können direkt digitale Ein- und Ausgänge angegeben werden.

Achtung

Diese Funktionalität wird nicht in allen informel Produkten unterstützt.

Syntax:

Digitaler Eingang Nummer (1..n)

%INummer

Digitaler Ausgang Nummer (1..n)

%QNummer

Beispiel

```
(* PageControl Zyklusprogramm *)
```

```
%Q1 := $JOB_COUNT > 0; (* Ausgangssignal wenn Rufe anstehen *)
```

```
%Q2 := $DEVICE_STATE / 2 MOD 2 (* Ausgangssignal bei Modemfehler *)
```

Hinweis

Siehe auch Funktion **QIMPULSE** zur Ausgabe von Ausgangsimpulsen.

Systemvariablen

Um zusätzliche Informationen bereitstellen zu können, sind Systemvariablen verfügbar, die in den Ausdrücken verwendet werden können. Alle Systemvariablen sind schreibgeschützt. Die Bezeichner aller Systemvariablen beginnen mit einem Dollarzeichen (\$).

Es sind folgende allgemein verfügbare Systemvariablen definiert:

Name	Beschreibung	Beispiel
\$TIME	Aktuelle Uhrzeit (Stunde und Minute)	09:15
\$DATE	Aktuelles Datum mit vierstelliger Jahreszahl	11.06.1997
\$HOUR	Aktuelle Stunde im 24 Stunden Format (00 .. 23)	09
\$MINUTE	Aktuelle Minute (00 .. 59)	15
\$SECOND	Aktuelle Sekunde (00 .. 59)	30
\$WEEKDAY	Aktueller Wochentag (0-Montag .. 6-Sonntag)	2
\$WEEK	Kalenderwoche (1 .. 53)	02
\$DAY	Aktueller Tag (01 .. 31)	11
\$MONTH	Aktueller Monat (01 .. 12)	06
\$YEAR	Aktuelles Jahr (vierstellig)	1997
\$ERROR	Beschreibung (englisch) des letzten aufgetretenen Laufzeitfehlers	

Neben diesen kann jede Anwendung spezifische Systemvariablen zur Verfügung stellen um auf spezielle interne Informationen zugreifen zu können.

Im Anhang finden Sie eine Liste der [anwendungsspezifischen Systemvariablen](#).

Umgebungsvariablen

Wird bei der Auswertung einer Variablen keine entsprechende im Script definierte Variable gefunden, so wird versucht eine entsprechende Umgebungsvariable zu finden, die den geforderten Wert liefern kann.

► Beispiel

Meldung von Station {{COMPUTERNAME}}

Liest den aktuellen Namen des Computers aus den Umgebungseinstellungen.

```
C:\>SET
ALLUSERSPROFILE=C:\Dokumente und Einstellungen\All Users
APPDATA=C:\Dokumente und Einstellungen\Frank.INFORMEL\Anwendungsdaten
CommonProgramFiles=C:\Programme\Gemeinsame Dateien
COMPUTERNAME=FRANK4
ComSpec=C:\WINDOWS\system32\cmd.exe
...
```

Liefert

Meldung von Station **FRANK4**

Datumsarithmetik

In begrenztem Umfang können mit Datums- und Uhrzeitwerten arithmetische Operationen ausgeführt werden. Unterstützt werden Addition und Subtraktion. Die verwendeten Zeitdauern werden immer in Sekunden angegeben.

Addition

Additionen werden verwendet um Zeitpunkte relativ zu einem gegebenen Zeitpunkt zu berechnen.

Syntax:

Zeitpunkt + *Zeitdauer*

► Beispiele

Zeitpunkt plus eine Stunde (60 Minuten * 60 Sekunden)

Ausdruck:

`DT#2006-05-29-09:05:51 + (60 * 60)`

Ergebnis:

`DT#2006-05-29-10:05:51`

Ausdruck:

`DT#NOW + (60 * 60)`

Ergebnis: Zeitpunkt in einer Stunde ab jetzt

Subtraktion

Mit Hilfe der Subtraktion kann die Dauer zwischen zwei Zeitpunkten bestimmt werden. Ist der Zeitpunkt1 früher als der Zeitpunkt2, so ist das Ergebnis negativ.

Syntax:

Zeitpunkt1 - *Zeitpunkt2*

► Beispiele

Zeitpunkt plus eine Stunde (60 Minuten * 60 Sekunden)

Ausdruck:

`DT#2006-05-29-09:05:51 - DT#2006-05-29-09:05:00`

Ergebnis:

51

Ausdruck:

`DT#2006-05-29-10:00:00 - DT#2006-05-29-09:05:51`

Ergebnis:

3249

Ausdruck:

`DT#2006-05-29-09:05:51 - DT#2006-05-29-10:00:00`

Ergebnis:

-3249

Standardfunktionen

Zur erweiterten Bearbeitung von Werten und Texten stehen einige eingebaute Funktionen zur Verfügung. Auf den folgenden Seiten werden die im informel STScript zur Verfügung stehenden Funktionen beschrieben.

Allgemeine Funktionen

VAR	Erlaubt die Verwendung von Variablen deren Namen Sonderzeichen enthalten
CLRVAR	Löscht den Wert der angegebenen Variablen
EVAL	Liefert das Ergebnis eines in einer Zeichenkette gespeicherten Ausdrucks

Arithmetische Funktionen

SHL	bitweises linksschieben
SHR	bitweises rechtsschieben

Bedingte Ausführung

SEL	wählt zwischen zwei Werten abhängig von einer Bedingung
IF, THEN, ELSIF, END_IF	Bedingte Ausführung von Anweisungen

Stringverarbeitung

MID	liefert einen mittleren Teil einer Zeichenkette
LEN	liefert die Länge einer Zeichenkette
LEFT	liefert den linken Teil einer Zeichenkette
RIGHT	liefert den rechten Teil einer Zeichenkette
FIND	liefert die erste Fundstelle einer Zeichenkette in einer anderen Zeichenkette
CONCAT	verbindet zwei oder mehrere Zeichenketten
REPLACE	ersetzt einen Abschnitt in einer Zeichenkette durch einen anderen
REPLACEALL	ersetzt alle Zeichenfolgen in einer Zeichenkette durch andere
FORMAT	formatiert einen Wert
GETCSV	liefert einen Eintrag aus einer mit Kommas getrennten Liste (CSV)
REGEXP	Analysiert eine Zeichenfolge gemäß dem angegebenen regulären Ausdruck

Datenbankfunktionen

DBFIELD	Zugriff auf Felder mit speziellen Namen
DBLOOKUP	liefert einen Wert aus einer Datenbankabfrage
DBREAD	Setzt Variablen mit dem Ergebnis einer Datenbankabfrage
DBWRITE	Schreibt einen neuen Datensatz mit dem Inhalt von Variablen
DBUPDATE	Ändert einen Datensatz mit den angegebenen Werten
DBDELETE*	Löscht Datensätze die der angegebenen Bedingung entsprechen
DBEXEC	Ruft eine in der Datenbank gespeicherte Prozedur auf
DBCOUNT	liefert die Anzahl der Datensätze einer Datenbankabfrage
DBXML	liefert das einer Datenbankabfrage im XML Format
DBTABLE	liefert das Ergebnis einer Datenbankabfrage in tabellarischem Format
DBCLOSE*	Schließt die angegebene Datenbankverbindung

Zugriff auf INI-Dateien

INILOOKUP	liest einen Wert aus einer INI Datei
INIWRITE	schreibt einen Wert in eine INI Datei

Allgemeine Dateioperationen

FILETEXT	liefert den Text, der in einer Datei abgelegt ist
----------	---

Netzwerkfunktionen

PING	Führt einen PING zum angegebene Host aus und liefert die benötigte Zeit
------	---

Digitale Ein-/Ausgabe


QIMPULSE*	gibt einen Impuls auf einem digitalen Ausgang aus
-----------	---

Ausführung externer Befehle

EXECUTE Ausführen eines externen Programms
DDEEXEC Ausführen eines DDE Befehls

Zugriff auf die informel Konfiguration

CFGLOGIN* Führt eine Anmeldung an der Konfigurationsdatenbank durch
CFG* Zugriff auf ein Konfigurationselement
CFGADDITEM* Legt ein neues Unterelement an
CFGREMITEM* Löscht das angegebene Element

 Achtung

Die mit * (Stern) gekennzeichneten Funktionen sind nicht in allen informel Produkten unterstützt.

Allgemeine Funktionen

VAR

Erlaubt den Zugriff auf Variablen deren Name Sondezeichen enthält.

Syntax:

```
VAR(Name)
```

▶ Beispiele

Ausdruck: Zugriff auf die Variable mit dem Namen „SPS-Wert“

```
X := VAR('SPS-Wert');  
VAR('SPS-Wert') := Y;
```

CLRVAR

Löscht den Inhalt der angegebenen Variablen.

Syntax:

```
CLRVAR(Name)
```

▶ Beispiel

Ausdruck

```
CLRVAR('DB_TIME');
```

★ Tipp

Beim Schreiben einer Datenbank mit DBWRITE werden Variablen deren Werte mit CLRVAR gelöscht wurden nicht gesetzt.

EVAL

Führt einen in einer Zeichenkette gespeicherten Script-Ausdruck aus und liefert dessen Ergebnis.

Syntax:

```
EVAL(Ausdruck)
```

▶ Beispiele

Ausdruck:

```
X := '1 + 1';  
EVAL(X);
```

Ergebnis:

```
2
```

Arithmetische Funktionen

SHL

Liefert den um Anzahl Bits nach links verschobenen Wert.

Syntax:

`SHL(Wert, Anzahl)`

▶ Beispiele

Ausdruck: $1 = 2\#0001$

`SHL(1, 2)`

Ergebnis: $4 = 2\#0100$

4

SHR

Liefert den um Anzahl Bits nach rechts verschobenen Wert.

Syntax:

`SHR(Wert, Anzahl)`

▶ Beispiele

Ausdruck: $8 = 2\#1000$

`SHR(8, 2)`

Ergebnis: $2 = 2\#0010$

2

Bedingte Ausführung

SEL

Liefert, abhängig ob die angegebene Bedingung wahr oder falsch ist, den Wert von AusdruckWahr oder AusdruckFalsch.

Syntax:

```
SEL(Bedingung, AusdruckWahr, AusdruckFalsch)
```

▶ Beispiel

Ausdruck:

```
SEL(LEFT($MESSAGE), 1) = '>', 'kommt', 'geht')
```

Ergebnis: wenn die Meldung mit dem Zeichen '>' beginnt

```
'kommt'
```

sonst

```
'geht'
```

IF, THEN, ELSIF, END_IF

Führt mehrere Script-Anweisungen in abhängigkeit einer Bedingung aus.

Syntax:

```
IF Bedingung THEN  
    AnweisungenWahr  
[ELSIF  
    AnweisungenFalsch]  
END_IF
```

▶ Beispiel

Ausdruck:

```
IF LEFT($MESSAGE), 1) = '>' THEN  
    MSG := 'kommt';  
ELSIF  
    MSG := 'geht';  
END_IF;
```

Stringverarbeitung

MID

Liefert den Teiltext von Text ab dem angegebenen Zeichen.

Syntax: liefert alle Zeichen ab der angegebenen Position

```
MID(Text, AbZeichen[, Länge])
```

Text	Originaltext
AbZeichen	Nummer des ersten Zeichens (1..n)
Länge	Anzahl der Zeichen die zurück gegeben werden sollen

► Beispiele

Ausdruck:

```
MID('abcdefghij', 5)
```

Ergebnis:

```
"ghij"
```

Ausdruck:

```
MID('abcdefghij', 5, 2)
```

Ergebnis:

```
"fg"
```

LEN

Liefert die Länge der angegebenen Zeichenkette.

Syntax:

```
LEN(Text)
```

Text	Text, dessen Länge bestimmt werden soll
-------------	---

► Beispiel

Ausdruck:

```
LEN('abcdef')
```

Ergebnis:

```
6
```

LEFT

Liefert den linken Teil von Text mit der angegebenen Länge.

Syntax:

```
LEFT(Text, Länge)
```

► Beispiel

Ausdruck:

```
LEFT('abcdefghij', 5)
```

Ergebnis:

```
"abcde"
```

RIGHT

Liefert den rechten Teil von Text mit der angegebenen Länge.

Syntax:

```
RIGHT(Text, Länge)
```

► Beispiel

Ausdruck:

```
RIGHT('abcdefghij', 5)
```

Ergebnis:

```
"fghij"
```

FIND

Liefert die erste Fundstelle von Suchtext innerhalb der Zeichenkette Text. Wird der Suchtext nicht gefunden, so ist das Ergebnis Null.

Syntax:

```
FIND(Text, Suchtext)
```

► Beispiel

Ausdruck:

```
FIND('abcdefghij', 'de')
```

Ergebnis:

```
4
```

CONCAT

Verbindet mehrere Zeichenketten, so dass eine Zeichenkette entsteht.

Syntax:

```
CONCAT(Text1{, Text2})
```

► Beispiele

Ausdruck:

```
CONCAT('Abc', 'Def')
```

Ergebnis:

```
"AbcDef"
```

Ausdruck:

```
CONCAT('abc', 'DEF', 'ghi')
```

Ergebnis:

```
"abcDEFghi"
```

Ausdruck:

```
CONCAT($TIME, ':', $SECOND)
```

Ergebnis: (z.B.)

```
"11:12:48"
```

REPLACE

Ersetzt einen Abschnitt in einer Zeichenkette durch einen anderen.

Syntax:

```
REPLACE(Text, Ersetzung, Länge, Position)
```

▶ Beispiel

Ausdruck:

```
REPLACE('ABCDE', 'X', 2, 3)
```

Ergebnis:

```
'ABXE'
```

REPLACEALL

Ersetzt alle Zeichenfolgen in einer Zeichenkette durch andere.

Syntax:

```
REPLACEALL(Text, SuchText, Ersetzung)
```

▶ Beispiele

Ausdruck:

```
REPLACEALL('Diest ist ein Test', 'e', 'X')
```

Ergebnis:

```
'DiXs ist Xin TXst'
```

Ausdruck:

```
REPLACEALL('CSV,123,abc,456', ',', ';')
```

Ergebnis:

```
'CSV;123;abc;456'
```

UPPERCASE

Gibt die Zeichenfolgen in Großbuchstaben zurück.

Syntax:

```
UPPERCASE(Text)
```

▶ Beispiel

Ausdruck:

```
UPPERCASE('Diest ist ein Test')
```

Ergebnis:

```
'DIES IST EIN TEST'
```

LOWERCASE

Gibt die Zeichenfolgen in Kleinbuchstaben zurück.

Syntax:

```
LOWERCASE(Text)
```

▶ Beispiel

Ausdruck:

```
LOWERCASE('Diest ist ein Test')
```

Ergebnis:

```
'dies ist ein test'
```

FORMAT

Liefert einen numerischen Wert gemäß AusgabeFormat formatiert.

Syntax:

```
FORMAT(NumerischerWert, AusgabeFormat)
```

Beispiele

Ausdruck:

```
FORMAT(123.4567, '0000.##')
```

Ergebnis:

```
'0123.45'
```

Ausdruck:

```
FORMAT(123.4, '####.00')
```

Ergebnis:

```
' 123.40'
```

Ausdruck:

```
FORMAT(123.4, '##.00')
```

Ergebnis:

```
'123.40'
```

Formatierungsmöglichkeiten

Ab 21-Mar-2006 erweiterte Formatierungsmöglichkeiten.

Syntax der Formatangabe:

[+|-]{#}{0}[*][.]{0}{#}]

Zeichen	Bedeutung
+	Erzwungenes Vorzeichen. Es wird ein Plus oder Minus ausgegeben.
-	Optionales Vorzeichen. Es wird bei negativen Werten ein Minus, bei positiven ein Leerzeichen ausgegeben.
#	Optionale Stelle. Ist die Zahl kürzer, so wird ein Leerzeichen ausgegeben.
0	Erzwungene Ziffer. Es wird die Ziffer oder eine Null ausgegeben.
*	Formatüberlauf. Mit dieser Angabe werden zu große Zahlen als Überlauf angezeigt.
.	Dezimalpunkt
0	Erzwungene Nachkommastelle. Es wird die Ziffer oder eine Null ausgegeben.
#	Optionale Nachkommastelle. Ist die Zahl kürzer, so erfolgt keine Ausgabe.

► Beispiele

Leerzeichen in der Ausgabe sind als □ dargestellt.

Wert	Formatangabe	Ausgabe	Bemerkung
1.23	0#0.##	1.23	ungültige Formatangabe
1.23	#0	□1	
1.23	##	□1	
1.23	#00	□01	
-1.23	##	-□1	
-1.23	0	-1	
-1.23	-00	-01	
1.23	00	01	
1.23	-00	□01	Vorzeichen nur bei negativen Werten
1.23	+00	+01	erzwungenes Vorzeichen
1.23	##.##	□1.23	
1.2345	##.##	□1.23	Abrundung
1.2356	##.##	□1.24	Aufrundung
1.23	#00.##	□01.23	
1.23	##.0000	□01.2300	
1.23	##.000#	□01.230	optionale vierte Nachkommastelle wird unterdrückt
1.1234	##.000#	□01.1234	optionale vierte Nachkommastelle wird ausgegeben
1.12345	##.000#	□01.1235	Aufrundung
-1.1234	##.000#	-□1.1234	
-1.12345	##.000#	-□1.1235	Aufrundung
92345.23	00000.###	92345.23	
92345.23	00000.000	92345.230	
92345.23	000*.###	#####	Formatüberlauf, Wert kann nicht vierstellig ausgegeben werden
123.0	*	#	Formatüberlauf, Wert kann nicht einstellig ausgegeben werden
123.0	#*	##	Formatüberlauf, Wert kann nicht zweistellig ausgegeben werden
123.0	##*	123	
123.0	###*	□123	
123.0	#000*	□0123	
1234.0	0000*	01234	
12345.0	0000*	12345	
123456.0	0000*	#####	Formatüberlauf, Wert kann nicht fünfstellig ausgegeben werden
123456.0	#####*	#####	Formatüberlauf, Wert kann nicht fünfstellig ausgegeben werden

GETCSV

Liefert einen Eintrag aus einer mit Kommas getrennten Liste (CSV).

Für Listen mit einem anderen Trennzeichen kann dieses wahlweise angegeben werden.

Die Zählung der Elementnummer beginnt bei eins.

Syntax:

```
GETCSV(WerteListe, ElementNummer[, Trennzeichen])
```

► Beispiele

Ausdruck:

```
GETCSV('12.03.2003,Störung,Anlage 1,kommend', 1)
```

Ergebnis:

```
'12.03.2003'
```

Ausdruck:

```
GETCSV('12.03.2003,Störung,Anlage 1,kommend', 3)
```

Ergebnis:

```
'Anlage 1'
```

Angabe alternativer Trennzeichen

Ausdruck: Trennzeichen Strichpunkt

```
GETCSV('12.03.2003;Störung;Anlage 1;kommend', 3, ';')
```

Ausdruck: Trennzeichen Tabulator

```
GETCSV('12.03.2003$Störung$Anlage 1$tkommend', 3, '$t')
```

REGEXP

Analysiert eine Zeichenfolge gemäß dem angegebenen regulären Ausdruck und ermöglicht die Rückgabe eines erkannten Teilausdrucks.

Syntax:

```
REGEXP(QuellText, RegFilter, Rückgabe)
```

QuellText Text der analysiert werden soll
RegFilter Regulärer Ausdruck der auf den Text angewendet werden soll
Rückgabe Ergebnistext der zurückgegeben werden soll.
 Dieser Text darf folgende Platzhalter enthalten:
 & Der komplette dem Muster entsprechende Text
 \1 Teilstück 1
 ...
 \9 Teilstück 9

► Beispiele

Extraktion der zwei dem Text **MotId**: folgenden Zeichen.

Ausdruck:

```
REGEXP('Prst:PS5010;MotId:EX144-3306;Meldung_01', 'MotId:(..)', '\1')
```

Ergebnis:

Zwei beliebige (Punkt) dem Text **MotId**: folgenden Zeichen (Parameter 1)

```
EX
```

Ausdruck:

```
REGEXP('Prst:PS5010;MotId:EX144-3306;Meldung_01', 'MotId:(..)', '&')
```

Ergebnis:

Der gesamte dem Muster entsprechende Text

```
ModId:EX
```

★ Tipp

Zum Test ob eine Zeichenfolge enthalten ist kann die Länge der Rückgabe ausgewertet werden.

Ausdruck:

```
LEN(REGEXP('Prst:PS5010;MotId:EX144-3306;Meldung_01', 'MotId:(..)', '&')) > 0
```

Ergebnis:

```
1
```

Datenbankfunktionen

Für den Zugriff auf externe Datenbanken stehen einige eingebaute Funktionen zur Verfügung.

Für die Beispiele in den folgenden Beschreibungen wird folgende Tabelle zu Grunde gelegt.

Tabelle: Maschine

Nummer (numerisch)	Bezeichnung (Text)	Stoerung (numerisch)	Zeitpunkt (Datum/Uhrzeit)
11	Presse	0	10:00
23	Band 1	7	10:31
34	Heizung	0	10:00
35	Lüftung	1	10:40
49	Druckluft	0	10:00

Datenbankverbindungen

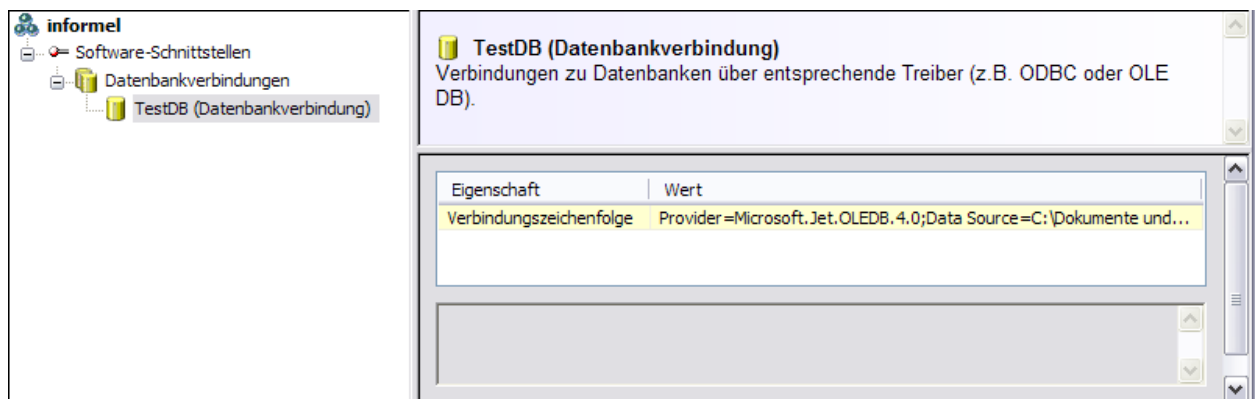
Unter einer Datenbankverbindung versteht man eine Zeichenkette, die die Verbindung zur Datenbank angibt. Diese Zeichenkette ist eine Datenverknüpfungseigenschaft und kann über ein geeignetes Programm generiert werden.

In PageControl dient hierzu die Funktion **Extras - Datenbankverbindungen**. Die Verbindungen werden hier unter dem Namen DB1, DB2, DB3 und DB4 bereitgestellt.

In Programmen die über den informel Konfigurations-Manager konfiguriert werden können die Datenbankverbindungen unter **Software-Schnittstellen – Datenbankverbindungen** eingerichtet werden. Im Script stehen diese Verbindungen unter dem Namen mit vorangestelltem „\$DB_“ zur Verfügung.

Syntax:

`$DB_` *Bezeichnung*



▶ Beispiel

Verbindung zu einer Microsoft Access Datenbank mit dem Namen „D:\Temp\TestData.mdb“.

Bezeichnung:

TestDB

Verbindungszeichenfolge:

Provider=Microsoft.Jet.OLEDB.4.0;Data Source=D:\Temp\TestData.mdb

Zugriff aus dem Script:

`DBREAD($DB_TestDB, 'Stoerung, Zeitpunkt', 'Maschine', 'Nummer=23')`

Hinweis

Die Groß- und Kleinschreibung der Bezeichnung muss beachtet werden.

DBFIELD

Die Funktion DBFIELD dient zum Zugriff auf Felder, deren Namen Sonderzeichen oder Leerzeichen enthalten. In diesen Fällen ist der Zugriff über die Bezeichnung DB_FeldName nicht möglich.

Syntax:

```
DBFIELD(FeldName)
```

▶ Beispiel

Es soll auf ein Feld mit Namen „Stoerungs-Nr“ zugegriffen werden.

Ausdruck:

```
DBFIELD('Stoerungs-Nr') := 5;
```

DBLOOKUP

Die DBLOOKUP Funktion wird verwendet, um einen Wert eines speziellen Feldes eines angegebenen Datensatzes zu lesen.

Syntax:

```
DBLOOKUP(DbConnection, Field, DataSource[, WhereClause[, OrderClause [DESC]])
```

- | | |
|---------------------|---|
| DbConnection | Zeichenkette, die die Verbindung zur Datenbank angibt. Diese Zeichenkette ist eine Datenverknüpfungseigenschaft und kann über ein geeignetes Programm generiert werden. |
| Field | In PageControl dient hierzu die Funktion Extras-Datenbankverbindungen. Gibt den Namen des Feldes an, dessen Wert als Ergebnis geliefert werden soll. |
| DataSource | Name der Tabelle oder Abfrage (View), die den Datensatz liefern soll. |
| WhereClause | Angabe (wahlweise) einer Bedingung, die den zu suchenden Datensatz auswählt. Ohne diese Angabe wird der erste Datensatz gelesen. |
| OrderClause | Angabe (wahlweise) eines Feldes nach dem sortiert werden soll. Ohne Angabe wird aufsteigend sortiert, mit der Angabe DESC absteigend. |

▶ Beispiel

Es soll die Bezeichnung von Maschine Nummer 34 aus der Tabelle Maschine gelesen werden. Die notwendige Verbindungszeichenfolge soll bereits in Variable DB1 gespeichert sein.

Ausdruck:

```
DBLOOKUP(DB1, 'Bezeichnung', 'Maschine', 'Nummer=34', 'Zeitpunkt DESC')
```

Ergebnis: liefert die gesuchte Bezeichnung

```
'Heizung'
```

DBREAD

Die Funktion DBREAD liest einen oder mehrere Felder des angegebenen Datensatzes und speichert die Werte in Variablen mit der Bezeichnung DB_FeldName, wobei jeweils FeldName durch den tatsächlichen Namen des Datenbankfeldes ersetzt wird.

Syntax1:

```
DBREAD(DbConnection, Fields, DataSource, WhereClause[, RecordNumber])
```

Syntax2:

```
DBREAD(DbConnection, Fields, DataSource, WhereClause, Order, RecordNumber)
```

DbConnection	Zeichenkette, die die Verbindung zur Datenbank angibt. Diese Zeichenkette ist eine Datenverknüpfungseigenschaft und kann über ein geeignetes Programm generiert werden. In PageControl dient hierzu die Funktion Extras-Datenbankverbindungen.
Fields	Zeichenkette mit der Liste der Feldnamen, durch Komma getrennt, deren Werte als Ergebnis geliefert werden soll.
DataSource	Name der Tabelle oder Abfrage (View), die den Datensatz liefern soll.
WhereClause	Angabe einer Bedingung, die den zu suchenden Datensatz auswählt.
Order	Angabe der Sortierreihenfolge.
RecordNumber	Angabe (wahlweise) der Datensatznummer, die gelesen werden soll. Ohne diese Angabe wird der erste Datensatz gelesen.

► Beispiele

Es soll die Störung und der Zeitpunkt von Band 1 (Nummer 23) aus der Tabelle Maschine gelesen werden. Die notwendige Verbindungszeichenfolge soll bereits in Variable DB1 gespeichert sein.

Ausdruck:

```
DBREAD(DB1, 'Stoerung, Zeitpunkt', 'Maschine', 'Nummer=23')
```

Ergebnis: liefert die Störung (Wert = 7) in der Variablen DB_Stoerung und den Zeitpunkt (10:31) in der Variablen DB_Zeitpunkt.

Ausdruck: 1. Eintrag lesen

```
DBREAD(DB1, 'Stoerung, Bezeichnung', 'Maschine', 'Stoerung=0', 1)
```

Ergebnis:

```
DB_Stoerung = '11'; DB_Bezeichnung = 'Presse';
```

Ausdruck: 2. Eintrag lesen

```
DBREAD(DB1, 'Stoerung, Bezeichnung', 'Maschine', 'Stoerung=0', 2)
```

Ergebnis:

```
DB_Stoerung = '34'; DB_Bezeichnung = 'Heizung';
```

DBWRITE

Die Funktion DBWRITE schreibt einen neuen Datensatz in die angegebene Tabelle. Die Werte der Felder werden den Variablen mit der Bezeichnung DB_FeldName entnommen, wobei jeweils FeldName durch den tatsächlichen Namen des Datenbankfeldes ersetzt wird.

Syntax:

```
DBWRITE(DbConnection, DataSource)
```

DbConnection Zeichenkette, die die Verbindung zur Datenbank angibt. Diese Zeichenkette ist eine Datenverknüpfungseigenschaft und kann über ein geeignetes Programm generiert werden.

In PageControl dient hierzu die Funktion Extras-Datenbankverbindungen.

DataSource Name der Tabelle in die der neue Datensatz geschrieben werden soll.

▶ Beispiel

Es soll ein neuer Eintrag für Maschine 77 (Rührwerk) in der Tabelle Maschine aufgenommen werden. Die notwendige Verbindungszeichenfolge soll bereits in Variable DB1 gespeichert sein.

```
DB_Nummer := 77;  
DB_Bezeichnung := 'Rührwerk';  
DB_Stoerung := 0;  
DB_Zeitpunkt := '10:00';  
DBWRITE(DB1, 'Maschine');
```

Die Funktion DBWRITE liefert TRUE, wenn der Datensatz geschrieben werden konnte, sonst FALSE.

⚠ Achtung

Diese Funktion kann keine bestehenden Datensätze ändern sondern nur neue Datensätze hinzufügen.

DBUPDATE

Syntax:

```
DBUPDATE (DbConnection, DataSource, Fields, Values, WhereClause)
```

DbConnection	Zeichenkette, die die Verbindung zur Datenbank angibt. Diese Zeichenkette ist eine Datenverknüpfungseigenschaft und kann über ein geeignetes Programm generiert werden. In PageControl dient hierzu die Funktion Extras-Datenbankverbindungen.
DataSource	Name der Tabelle dessen Datensätze geändert werden sollen.
Fields	Zeichenkette mit der Liste der Feldnamen, durch Komma getrennt, deren Werte geändert werden sollen.
Values	Liste der Werte die den Feldern zugewiesen werden sollen
WhereClause	Angabe einer Bedingung, die die zu ändernden Datensätze auswählt.

► Beispiel

Es soll die Störung der Lüftung (Nummer=35) auf Null gesetzt und die aktuelle Zeit eingetragen werden. Die notwendige Verbindungszeichenfolge soll bereits in Variable DB1 gespeichert sein.

```
DBUPDATE (DB1, 'Maschine', 'Stoerung,Zeitpunkt', CONCAT('0,', $TIME), 'Nummer=35');
```

DBDELETE

Die Funktion DBDELETE löscht die Datensätze in der angegebenen Tabelle.

Syntax:

```
DBDELETE (DbConnection, DataSource, WhereClause)
```

DbConnection	Zeichenkette, die die Verbindung zur Datenbank angibt. Diese Zeichenkette ist eine Datenverknüpfungseigenschaft und kann über ein geeignetes Programm generiert werden. In PageControl dient hierzu die Funktion Extras-Datenbankverbindungen.
DataSource	Name der Tabelle dessen Datensätze gelöscht werden sollen.
WhereClause	Angabe einer Bedingung, die die zu löschenden Datensätze auswählt. Ohne diese Angabe (leer) werden alle Datensätze gelöscht.

► Beispiel

Es soll der Eintrag für Maschine 77 (Rührwerk) aus der Tabelle Maschine gelöscht werden. Die notwendige Verbindungszeichenfolge soll bereits in Variable DB1 gespeichert sein.

Ausdruck:

```
DBDELETE (DB1, 'Maschine', 'Nummer=77');
```

DBEXEC

Ruft eine in der Datenbank gespeicherte Prozedur auf.

Syntax:

```
DBEXEC (DbConnection, DbProcedure)
```

► Beispiel

Die in der Datenbank hinterlegte Prozedur „AddMasch“ soll aufgerufen werden. Die notwendige Verbindungszeichenfolge soll bereits in Variable DB1 gespeichert sein.

Ausdruck:

```
DBEXEC (DB1, 'AddMasch ()')
```

DBCOUNT

Die Funktion liefert die Anzahl der Datensätze, die den gegebenen Bedingungen entsprechen.

Syntax:

```
DBCOUNT(DbConnection, DataSource, WhereClause)
```

- DbConnection** Zeichenkette, die die Verbindung zur Datenbank angibt. Diese Zeichenkette ist eine Datenverknüpfungseigenschaft und kann über ein geeignetes Programm generiert werden.
In PageControl dient hierzu die Funktion Extras-Datenbankverbindungen.
- DataSource** Name der Tabelle oder Abfrage (View), deren Datensätze gezählt werden sollen.
- WhereClause** Angabe einer Bedingung, die die zu zählenden Datensätze auswählen.

► Beispiel

Es soll die Anzahl der Maschinen in Störung bestimmt werden.

Die notwendige Verbindungszeichenfolge soll bereits in Variable DB1 gespeichert sein.

Ausdruck:

```
DBCOUNT(DB1, 'Maschine', 'Stoerung<>0')
```

Ergebnis: Band 1 und Lüftung

2

DBXML

Die Funktion DBXML liefert das Ergebnis einer Datenbankabfrage im XML Format zurück.

Syntax:

```
DBXML(DbConnection, SqlStatement)
```

- DbConnection** Zeichenkette, die die Verbindung zur Datenbank angibt. Diese Zeichenkette ist eine Datenverknüpfungseigenschaft und kann über ein geeignetes Programm generiert werden.
In PageControl dient hierzu die Funktion Extras-Datenbankverbindungen.
- SqlStatement** SQL Ausdruck, der die Datenbankabfrage beschreibt.

► Beispiel

Es soll die Liste der Maschinen in Störung gelesen werden.

Die notwendige Verbindungszeichenfolge soll bereits in Variable DB1 gespeichert sein.

Ausdruck:

```
DBXML(DB1, 'SELECT * FROM Maschine WHERE Stoerung<>0')
```

Ergebnis: liefert die folgenden zwei Datensätze im XML Format.

```
<data>
  <rs>
    <Nummer>23</Nummer>
    <Bezeichnung>Band 1</Bezeichnung>
    <Stoerung>7</Stoerung>
    <Zeitpunkt>10:31</Zeitpunkt>
  </rs>
  <rs>
    <Nummer>35</Nummer>
    <Bezeichnung>Lüftung</Bezeichnung>
    <Stoerung>1</Stoerung>
    <Zeitpunkt>10:40</Zeitpunkt>
  </rs>
</data>
```

DBTABLE

Die Funktion DBTABLE liest einen oder mehrere Felder des angegebenen Datensatzes und liefert die Datensätze als Zeilen mit durch Tabulator getrennten Werten.

Syntax:

```
DBTABLE(DbConnection, DataSource, Fields, WhereClause)
```

- DbConnection** Zeichenkette, die die Verbindung zur Datenbank angibt. Diese Zeichenkette ist eine Datenverknüpfungseigenschaft und kann über ein geeignetes Programm generiert werden.
In PageControl dient hierzu die Funktion Extras-Datenbankverbindungen.
- DataSource** Name der Tabelle oder Abfrage (View), die den Datensatz liefern soll.
- Fields** Zeichenkette mit der Liste der Feldnamen, durch Komma getrennt, deren Werte als Ergebnis geliefert werden soll.
- WhereClause** Angabe einer Bedingung, die den zu suchenden Datensatz auswählt.

► Beispiel

Es sollen alle Maschinen in Störung als Tabelle ausgegeben werden.

Die notwendige Verbindungszeichenfolge soll bereits in Variable DB1 gespeichert sein.

Ausdruck:

```
DBTABLE(DB1, 'Maschine', 'Nummer, Bezeichnung', 'Stoerung>0')
```

Ergebnis: liefert die folgende Tabelle.

```
23   Band 1
35   Lüftung
```

DBCLOSE

Schließt die angegebene Datenbankverbindung.

Syntax:

```
DBCLOSE(DbConnection)
```

- DbConnection** Zeichenkette, die die Verbindung zur Datenbank angibt. Diese Zeichenkette ist eine Datenverknüpfungseigenschaft und kann über ein geeignetes Programm generiert werden.
In PageControl dient hierzu die Funktion Extras-Datenbankverbindungen.

► Beispiel

Ausdruck:

```
DBCLOSE(DB1);
```

Zugriff auf INI-Dateien

INILOOKUP

Wandelt den angegebenen Wert gemäß dem Abschnitt in der INI-Datei. Der Wert darf hierbei sowohl numerisch oder textuell sein. Es können mehrere Abschnitte in einer Datei enthalten sein. Es muss der volle Dateipfad angegeben werden. Wird kein passender Eintrag gefunden, so ist das Ergebnis ein leerer Text.

Syntax:

```
INILOOKUP(Abschnitt, IniDatei, Schlüssel)
```

► Beispiel

Wandelung einer Störungsnummer (hier fest Nummer 17) in den zugehörigen Text (hier 'Überstromabschaltung Rührwerk 2')

Inhalt der Datei C:\Programme\informel\PageControl\MTexte.ini:

```
[MsgText]
1=Störung Pumpe 2
17=Überstromabschaltung Rührwerk 2
20=Steuerungsstörung Anlage 2
```

Ausdruck:

```
INILOOKUP('MsgText', 'C:\Programme\informel\PageControl\MTexte.ini', 17)
```

Ergebnis:

```
'Überstromabschaltung Rührwerk 2'
```

INIWRITE

Schreibt den angegebenen Wert gemäß dem Abschnitt und dem Schlüssel in die INI-Datei. Der Wert darf hierbei sowohl numerisch oder textuell sein. Es können mehrere Abschnitte in einer Datei enthalten sein. Es muss der volle Dateipfad angegeben werden. Wird die Datei bzw. der Abschnitt nicht gefunden, so werden sie automatisch angelegt.

Syntax:

```
INIWRITE(IniDatei, Abschnitt, Schlüssel, Wert)
```

► Beispiel

Schreibt den Text für Störungsnummer 18 in die INI-Datei

Ausdruck:

```
INIWRITE('C:\PageControl\MTexte.ini', 'MsgText', 18, 'Pegel unterschritten')
```

Ergebnis: Inhalt der Datei C:\PageControl\MTexte.ini nach Ausführung des Befehls

```
[MsgText]
1=Störung Pumpe 2
17=Überstromabschaltung Rührwerk 2
18=Pegel unterschritten
20=Steuerungsstörung Anlage 2
```

Allgemeine Dateioperationen

FILETEXT

Diese Funktion liefert den Text, der in der Datei mit dem Namen `DateiName` abgelegt ist. Ohne die Angabe des Modusses wird nur die erste Zeile geliefert.

Syntax:

```
FILETEXT(DateiName [, Modus])
```

Es stehen folgende Modi zur Verfügung

FT_FirstLine Liefert nur die erste Zeile (Standard)

FT_NoCrLf Liefert alle Zeilen wobei die Zeilenvorschübe durch Leerzeichen ersetzt werden.

FT_AllLines Liefert alle Zeilen unverändert.

Achtung

Diese Funktion begrenzt die Größe des Ergebnisses auf eine Länge von 4kB.

Die Beispiele lesen aus einer Textdatei (`D:\Test.txt`) mit folgendem Inhalt.

```
Störung 17:15
Pumpwerk Untermatt
Druckschalter
```

Beispiele

Ausdruck:

```
FILETEXT('D:\Test.txt')
```

Ergebnis:

```
'Störung 17:15'
```

Ausdruck:

```
FILETEXT('D:\Test.txt', FT_AllLines)
```

Ergebnis:

```
'Störung 17:15
Pumpwerk Untermatt
Druckschalter'
```

Ausdruck:

```
FILETEXT('D:\Test.txt', FT_NoCrLf)
```

Ergebnis:

```
'Störung 17:15 Pumpwerk Untermatt Druckschalter'
```

Netzwerkfunktionen

PING

Führt einen PING zum angegebenen Host aus und liefert die benötigte Zeit

Syntax:

```
PING (Host)
```

Host Name oder IP-Adresse des Hosts.

Beispiel

Ausdruck:

```
PING ('10.1.1.2')
```

Ergebnis:

```
0.512885
```

WAKEONLAN

Steuerung der WakeOnLan Funktion von Rechnern im lokalen Netzwerk.

Rechner in Standby bringen und Rechner wieder aufwecken.

Als Ergebnis wird 1, wenn die Ausführung erfolgreich war. Im Fehlerfall liefert die Funktion 0 (Null) und Fehler-Informationen in der Systemvariablen \$ERROR.

Syntax:

```
WAKEONLAN(Anmeldung, IpAdresse[:Port] [/ (Präfix|SubnetMask)], MacAdresse, Funktion)
```

Anmeldung Hier wird der Benutzername und als Kennwort des Benutzers angegeben der das Recht hat den Vorgang durchzuführen.

Format:

```
User=Benutzername;Password=Kennwort
```

Um die Anmeldedaten nicht an dieser Stelle im Klartext angeben zu müssen kann eine Identität aus der Konfiguration angegeben werden.

IpAdresse Hier wird die IP-Adresse des zu steuernden Rechners angegeben. Sie ist für das korrekte Routing des Pakets notwendig.
Im Falle des wieder Aufweckens ist eine Broadcastadresse notwendig. Um aus der IP-Adresse die zugehörige Broadcastadresse zu berechnen muss entweder die Netzwerkgröße oder die Subnetzmaske direkt angegeben werden.

Port Wahlweise kann eine Portnummer angegeben werden über die das Paket verschickt wird. Die Standardeinstellung ist Port 80 (http).

Präfix / SubnetMask Netzwerkgröße oder Subnetzmaske

Subnetzmaske	Präfix
255.0.0.0	/8
:	:
255.255.0.0	/16
:	:
255.255.255.0	/24
:	:
255.255.255.252	/30

Ohne Angabe Netzwerkgröße oder Subnetzmaske wird an alle Broadcastadressen der Netzwerke 8 bis 30 gesendet.

MacAdresse Diese Adresse wird für das wieder Aufwecken benötigt, da sie in den Daten des zu sendenden IP-Pakets enthalten sein muss.

Funktion Über diesen Parameter wird die auszuführende Funktionalität gewählt. Für die Funktionen 1 bis 5 kann zusätzlich die Option Ping (101 bis 105) angegeben werden. Die Erreichbarkeit des Zielrechners wird dann vor Ausführung der Funktion per Ping geprüft.

Wert	Funktion
0	WakeOnLan
1	Abmelden
2	Standby
3	Hibernate
4	Shutdown
5	Restart
+100	Ping

▶ Beispiel1

Rechner mit der IP-Adresse 10.1.1.26 in Standby versetzen mit voriger Erreichbarkeitsprüfung mit Ping.

Ausdruck:

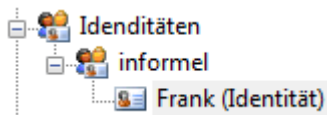
```
WAKEONLAN('User=informel\Frank;Password=test','10.1.1.26:100/8','00-08-74-07-47-CF',102)
```

Ergebnis:

1

▶ Beispiel2

Angabe einer Identität aus der Konfiguration:



Eigenschaft	Wert
Beschreibung	Ein informel Account
Benutzername	informel\Frank
Kennwort	****

Ausdruck:

```
WAKEONLAN('Frank','10.1.1.26:100/8','00-08-74-07-47-CF',2)
```

Ergebnis:

1

⚠ Achtung

Für die Ausführung der Funktionen 1-5 (Abmelden-Restart) benötigt der angegebene Benutzer die Zugriffsrechte auf WMI. Bei den Funktionen 4+5 (Shutdown, Restart) zusätzlich das Recht den Computer herunterzufahren.

Funktionsweise

WakeOnLan

Um Wake-On-LAN nutzen zu können, müssen sowohl die Hauptplatine, als auch die Netzwerkkarte ACPI (Advanced Configuration and Power Interface) unterstützen. Während der Computer ausgeschaltet ist, wird die Netzwerkkarte über den Standby-Stromzweig des Netzteils weiterhin mit Strom versorgt.

Die Netzwerkkarte wartet auf ein so genanntes Magic Packet (Schutzmarke von AMD), bei dessen Empfang der Rechner eingeschaltet wird.

Das Datenpaket ist entweder direkt an die Netzwerkkarte adressiert oder wird als Broadcast verschickt. Es enthält 6 mal in Folge den hexadezimalen Wert FF; unmittelbar danach erscheint die ununterbrochene 16-malige Wiederholung der MAC-Adresse der Netzwerkkarte. Dieser Inhalt wird in einem UDP Paket verschickt.

Standby

Das Versetzen eines Rechners in den Standby-Zustand wird über WMI (Windows Management Instrumentation) und die Klasse Win32_Process erreicht. Hierzu wird der folgende Prozess ausgeführt.

```
RunDll32.exe Kernel32.dll,SetSystemPowerState 1,0
```

Systemvoaussetzungen

Damit die Wake-On-LAN Funktionalität genutzt werden kann sind die folgenden Systemvoraussetzungen zu erfüllen.

Dienste

Die folgenden Windows-Dienste müssen gestartet sein.

- Remoteprozeduraufruf (RPC)
- Windows-Verwaltungsinstrumentation

Firewall freischalten

Die Windows-Firewall muss so konfiguriert sein, dass eingehende WMI Anfragen nicht blockiert werden.

Dies kann z.B. mit dem folgenden Kommando eingestellt werden.

```
netsh firewall set service type=remoteadmin mode=enable scope=all profile=all
```

Digitale Ein-/Ausgabe

QIMPULSE

Es wird ein Impuls auf dem digitalen Ausgang ausgegeben. Die Impulsdauer wird hierbei in Zehntelsekunden angegeben. Eine negative Impulsdauer gibt einen negativen Impuls aus.

Syntax:

```
QIMPULSE (Ausgang, Impulsdauer)
```

Beispiel

Ausdruck:

```
QIMPULSE (%Q3, 30)
```

Ergebnis: setzt den Ausgang 3 für drei Sekunden

0

Das Ergebnis der Funktion ist der Zustand des betroffenen Ausgangs vor Ausführung des Befehls.

Ausführung externer Befehle

EXECUTE

Startet ein externes Programm. Wahlweise kann das Fenster des gestarteten Programms sichtbar (=1, Standardwert) oder unsichtbar (=0) sein.

Syntax:

```
EXECUTE(Kommandozeile[, FensterSichtbar])
```

Die Funktion liefert als Ergebnis Null, wenn der Prozess fehlerfrei gestartet werden konnte, sonst den Fehlercode des Betriebssystems.

Beispiel

Ausdruck:

```
EXECUTE('Notepad')
```

Ergebnis: startet den Windows Editor

0

Hinweis

Soll ein Kommandozeilenbefehl ausgeführt werden so muss ausdrücklich der Kommandointerpreter angegeben werden.

Soll zum Beispiel ein externes VBScript Programm ausgeführt werden so muss der Befehl wie folgt angegeben werden.

Es ist sinnvoll in diesem Fall das Fenster des Kommandointerpreters zu verstecken.

Ausdruck:

```
EXECUTE('CMD /c MyProg.vbs', 0)
```

Ergebnis:

0

DDEEXEC

Führt ein DDE Kommando der angegebenen Anwendung aus.

Syntax:

```
DDEEXEC(Server, Topic, Command)
```

Server	Name der DDE Anwendung
Topic	Bezeichnung des DDE Themas
Command	Das auszuführende Kommando

Beispiel

Ausdruck:

```
DDEEXEC('Excel', 'System', '[open("C:\TEST.XLS")]')
```

Ergebnis: Excel öffnet die Datei C:\TEST.XLS.

Zugriff auf die informel Konfiguration

CFGLOGIN

Führt eine Anmeldung bei der informel Konfigurationsdatenbank aus. Ohne diese Anmeldung ist kein Zugriff auf die Konfiguration möglich.

Syntax:

```
CFGLOGIN(Benutzer, Kennwort)
```

Die Funktion liefert TRUE oder FALSE in Abhängigkeit des Erfolgs der Anmeldung.

► Beispiel

Ausdruck:

```
CFGLOGIN('Müller', 'T6zzu2')
```

Ergebnis: Meldet den Benutzer „Müller“ mit dem Kennwort „T6zzu2“ an

1

⚠ Achtung

Das Kennwort muss Unverschlüsselt angegeben werden. Damit ist es für alle die auf das Script zugreifen können lesbar.

CFG

Erlaubt den Zugriff auf ein Element der informel Konfigurationsdatenbank. Es ist sowohl lesender als auch schreibender Zugriff möglich. Die Zugriffsrechte der Anmeldung werden jedoch berücksichtigt.

Syntax:

```
CFG(KonfigurationsPfad)
```

KonfigurationsPfad Kompletter Pfad zum Konfigurationselement in interner Darstellung

► Beispiel

Ausdruck:

```
CFG('PageServer/Targets/Müller/Comment')
```

Ergebnis: Der Kommentar der zu Empfänger „Müller“ in der Konfiguration abgelegt ist

```
'Bernd'
```

CFGADDITEM

Fügt ein neues Element in den Konfigurationsbaum ein. Die Zugriffsrechte der Anmeldung werden jedoch berücksichtigt.

Syntax:

```
CFGADDITEM(KonfigurationsPfad, TypNummer, Bezeichnung)
```

KonfigurationsPfad Kompletter Pfad zum Konfigurationselement in interner Darstellung

TypNummer Numerische Angabe des Typs des neuen Elements

Bezeichnung Bezeichnung mit der das neue Element angelegt werden soll

Die Funktion liefert TRUE wenn das Element hinzugefügt werden konnte, sonst FALSE.

► Beispiel

Ausdruck:

```
CFGADDITEM('PageServer/Targets', 12560, 'Meier')
```

Ergebnis: Fügt einen neuen Empfänger vom Typ „E-Mail über SMTP“ mit der Bezeichnung „Meier“ ein.

1

CFGREMITEM

Entfernt ein Element aus den Konfigurationsbaum. Die Zugriffsrechte der Anmeldung werden jedoch berücksichtigt.

Syntax:

```
CFGREMITEM(KonfigurationsPfad, Element)
```

KonfigurationsPfad	Kompletter Pfad zum Konfigurationselement in interner Darstellung
TypNummer	Numerische Angabe des Typs des neuen Elements
Bezeichnung	Bezeichnung mit der das neue Element angelegt werden soll

Die Funktion liefert TRUE wenn das Element entfernt werden konnte, sonst FALSE.

Beispiel

Ausdruck:

```
CFGADDITEM('PageServer/Targets', 'Meier')
```

Ergebnis: Löscht den Empfänger „Meier“

1

Reguläre Ausdrücke

Reguläre Ausdrücke werden immer dann verwendet, wenn ein Text auf ein bestimmtes Format geprüft werden soll und/oder Teile eines Textes extrahiert werden sollen. Der reguläre Ausdruck gibt dann das erwartete Format an. Durch Klammerung können teile des Textes extrahiert werden.

Zeichen	Beschreibung
^	Anfang des Textes. Der Ausdruck "^A" passt nur zu einem 'A' am Anfang des Textes.
[^	Das Dach Zeichen (Caret, ^) direkt nach einer eckigen Klammer auf ([]) hat eine andere Bedeutung. Es wird dazu verwendet, die restlichen Zeichen innerhalb der Klammer auszuschließen. Der Ausdruck "[^0-9]" gibt an, dass der Text an dieser Stelle keine Ziffer enthalten darf.
\$	Das Dollar Zeichen (\$) gibt die Position des Textendes an. Der Ausdruck "abc\$" passt nur dann zu "abc" wenn es am Ende des Textes steht.
	Der senkrechte Strich () erlaubt die Oder Verknüpfung zweier Ausdrücke. Der Ausdruck "a b" passt sowohl zu 'a' als auch zu 'b'.
.	Der Punkt (.) passt zu jedem Zeichen.
*	Der Stern (*) zeigt an, dass das Zeichen links davon im Text nicht oder mehrmals hintereinander steht.
+	Das Plus (+) ist ähnlich wie der Stern, es muss jedoch mindestens einmal das Zeichen links davon im Text stehen.
?	Das Fragezeichen (?) gibt an, dass das Zeichen links davon nicht oder nur einmal auftreten darf.
()	Die Klammern geben den Teil bzw. die Teile des Textes an, der für den Meldungsfilter relevant ist/sind.
[]	Die eckigen Klammern ([und]) geben eine Menge von Zeichen an, die an dieser Stelle des Textes stehen dürfen.
\	Das folgende Zeichen wird als Literal verwendet. Dies ist notwendig, wenn z.B. auf einen Punkt geprüft werden soll.

Anwendungsspezifische Systemvariablen

Name	Beschreibung	Beispiel	Anwendung(en)
\$INPORT	Schnittstelle der Meldungsquelle	Seriell1	MessageControl
\$MSGSRC	Meldungsquelle	MsgJuli	RemoteControl
\$MESSAGE	Originaltext einer Meldung	Alarm Anlage 1	PageControl, MessageControl
\$MSGFIELD[M]	Feld einer Meldung die, durch Strichpunkt getrennt, mehrere Felder hat entspricht GETCSV(\$MESSAGE,N)		MessageControl (4.30.001)
\$SIGNCODE	Quittungscode der aktuellen Meldung	123	PageControl
\$TRIGGER	Auslöser der Meldung ("DDE" oder Signalname)	DDE	PageControl
\$GROUP	Bezeichnung der Meldungsgruppe	Service	PageControl (6.40.017)
\$RECEIVER	Bezeichnung des Meldungsempfängers	Schmidt	PageControl
\$PRIORITY	Priorität der Meldung	5	PageControl
\$ESCALATIONLEVEL	Eskalationsstufe der Meldung	2	PageControl (7.30.001)
\$DEVICESTATE	Aktueller Status der Schnittstelle und des Modems (Bitcodiert)	2	PageControl
\$BARRINGACTIVE	Aktueller Status der extern gesteuerten Rufsperr	1	PageControl
\$JOBSTATE	Aktueller Sendestatus	10	PageControl
\$JOBCOUNT	Anzahl der Rufe in der Sendeliste	2	PageControl
\$LOGCOUNT	Anzahl der Einträge im Logbuch	50	PageControl
\$CALLID	Eindeutige ID eines Rufs	3976DEC93BD631	PageControl
\$STATIONNAME	Bezeichnung der PageControl Station	Pumpwerk 2	PageControl, RemoteControl
\$STATIONNUMBER	Nummer der PageControl Station	0721 9414208	PageControl, RemoteControl
\$MSGLIST	Liste der aktiven Meldungen (begrenzt auf 20 Meldungen)	Störung Pumpe 1 Übertemperatur Motor	PageControl
\$CURRENTSHIFT	Nummer der momentan eingestellten Schicht	2	PageControl
\$PARAMETER	Parameter eines Fernwirkbefehls	55	PageControl (6.40.027)
\$EXTAPPINSTALLED	Installierte externe Anwendungen (Bitcodiert)	3	PageControl (7.10.001)
\$EXTAPPSTATE	Status der externen Anwendungen (Bitcodiert)	3	PageControl (7.10.001)
\$LASTCHOICE	Letzte Auswahl	5	PhoneControl
\$RECORDEDFILE	Dateiname der letzten Aufzeichnung	C:\RECVAB_R10001.WAV	PhoneControl
\$CALLINGNUMBER	Nummer des Anrufers, wenn bekannt (CLI)	07214098910	PhoneControl
\$CALLEDNUMBER	Nummer die angerufen wurde (DNIS)	07219414208	PhoneControl
\$LINE	Nummer der Leitung auf der der Anruf geführt wird	1	PhoneControl
\$PATH	Verzeichnis des momentanen Ablaufplans	C:\Phone\AB	PhoneControl
\$TRANSFERRESULT	Ergebnis einer Weitervermittlung	0	PhoneControl
\$EXECUTERESULT	Ergebnis eines extern ausgeführten Programms	0	PhoneControl
\$ITEMNAME	Name des aktuellen Elements (Meldung oder Wert)		RemoteControl