PWR_CMD

Användarhandledning

Revision: 99 03 30 Claes Sjöfors

SSAB Oxelösund

Allmänt

Pwr_cmd innehåller ett antal funktion för att hantera utvecklingsdatabasen, ta ut listor, få information om plcprogram, skapa laddatafiler, kompilera mm.

Pwr_cmd anropas från dcl med symbolen 'pwr_cmd', eller från Utilities i pwr_dev.

I pwr_cmd finns kommandon för distribution av laddatafiler. Dessa beskrivs i användarhandledning för distributören.

Kommandon

Nedan följer beskrivning på kommandon i pwr_cmd.

compile

Kompilerar logik och grafcet fönster skapade i PldEdit.
Kompilering kan ske för ett fönster, för alla fönster i ett plcpgm eller för alla fönster och plcpgm i en node.

Syntax

pwr>compile /window=
pwr>compile /plcpgm=
pwr>compile /volumes=
pwr>compile /allvolumes
pwr>compile /from_plcpgm

/window

Hierarkinamnet på det window-objekt som ska komipleras.

/plcpgm

Hierarkinamnet på det plcpgm som ska komipleras.

/volume

Namn på en eller flera volymer. Samtiga plcpgm i angivna volymer kommer att kompileras. Kan vara en lista på volymer

/allvolumes

Samtliga plcpgm i samtliga volymer kommer att kompileras.

/debug

Kompilering sker med debug.

/modified

Endast plc-moduler som är modifierade sedan senaste kompilering kommer att kompileras.

/from_plcpgm

Om en kompilering av alla program i volymen har avbrutits av någon anledning kan men forsätta kompileringen på ett visst plcpgm.

configure

Hjälpmedel för att konfigurera systemet.

configure card konfigurerar ett kort.
Syntax:

pwr> configure card /rack= /cardname= /cardclass=
[/chanidentity=]
[/chandescription=][/table=]

configure card

Skapar ett kortobjekt av specificerad klass med tillhörande kanalobjekt.

/rack

Namnet på rackobjektet under vilket kortet ska ligga.

/cardname

Namn på kortobjektet som ska skapas.
Endast sista segmentet av namnet anges.

/channelname

Namn på kanalobjekt som ska skapas.
Endast sista segmentet av namnet anges.
Ett # tecken kommer att ersättas av kannalnummer
Ex /channelname=di33# ger kanalnamn di3301, di3302, di3303...
För kort som innehåller flera kanaler måste namnet innehålla ett # tecken.

/chanidentity

Kanalens identitet.
Läggs in i attributet Identity i kanalobjektet.
Ett # tecken kommer att ersättas av kannalnummer

/chandescription

Beskrivning av kanalen.
Läggs in i attributet Description i kanalobjektet.
Ett # tecken kommer att ersättas av kannalnummer

/table

De kanaler som ska vara av typen tabell (tex ChanAit)
anges med en lista eller ett intervall av kanalnummer.

Ex
/table=3
/table=(3,5,6)
/table=(3,5,6,10-20)

connect

Koppla ihop ett signalobjekt med ett kanalobjekt.

/source

Ett kanal eller signalobjekt.

/destination

Ett kanal eller signalobjekt

/reconnect

Om något av objekten redan är kopplat görs disconnect före kopplingen.

copy

Kopierar objekt i databasen.

copy objects

Kopierar ett objekt eller en hierararki av objekt.

Syntax:

pwr> copy objects /source= /destination= /name=[/hierarchy] [/before] [/after] [/first] [/last]

/source

Namn på objekt som ska kopieras.

/destination

Plats dit aktuellt objekt ska kopieras. Namn på förälder eller syskon. Om /last eller /first anges kommer objektet att läggas som barn till destinations objeket, om /before eller /after som syskon.

/hierarchy

Samtliga objekt som ligger under source objektet kopieras
också.

/namn

Namn på objekt som ska skapas, sista namn ledet.

/first

Objektet läggs som första barn till destinations objektet.

/last

Objektet läggs som sista barn till destinations objektet (default).

/after

Objektet läggs som syskon efter destinations objeketet

/before

Objektet läggs som syskon före destinations objeketet.

Exempel
pwr> copy objects /destination=ugn-motor-start /source=ugn-fläktmotor /name=start /hierarchy

create

Skapar objekt, laddatafiler, bootfiler eller structfiler.

create bootfile

Skapar en nya bootfiler.

/nodeconfig

NodeConfig-objekt för vilket boot-fil ska skapas.
Kan vara en lista på NodeConfig-objekt.

/debug

Plc-programmet länkas med debug.

/allnodes

Skapar bootfiler för samtliga noder i systemet.

create loadfiles

Skapar laddatafiler.

/volume

Volym för vilken laddatafile ska skapas. Kan vara en lista på volymer.

/allvolumes

Laddatafiler skapas för alla volymer i databasen (ej klassvolymer).

/classvolumes

Laddatafiler skapas för samtliga klassvolymer i databasen.

/commonobjects

Skapar laddatafil för gemensamma objekt.

/allnodes

Skapar laddatafiler för alla noder i systemet.

/new

Skapar laddatafiler för hela systemet, dvs för systemmetaobjects, projectmetaobjects, commonobjects och för samtliga noder.

/debug

Plcprogrammet länkas med debug.

create object

Skapa ett objekt.

/destination

Plats där objektet ska skapas. Namn på förälder eller syskon. Om /last eller /first anges kommer objektet att läggas som barn till destinations objeket, om /before eller /after som syskon.

/namn

Namn på objekt som ska skapas, sista namn ledet.

/class

Class på objekt som ska skapas.

/first

Objektet läggs som första barn till destinations objektet.

/last

Objektet läggs som sista barn till destinations objektet.

/after

Objektet läggs som syskon efter destinations objektet

/before

Objektet läggs som syskon före destinations objektet.

Exempel
pwr> create object /destination=ugn-motor /name=start /class=Dv /first

create structfiles

Skapar en includefil för en klassvolym som innehåller volymens klasser i beskrivna i form av c-structar.

/volume

Klassvolym för vilken structfil ska skapas. Kan vara en lista på klassvolymer.

Exempel

pwr> create structfiles /volume=CVolVKVEND

crossreference

Listar referenser i plcprogrammet för
- ett objekt
- samtliga objekt av en klass.
- samtliga objekt av en klass under ett objekt i hierarkin.

Syntax:

pwr>crossreferense /object=
pwr>crossreferense /class=
pwr>crossreferense /hierarchy= /class= /name=

/object

Hierarkinamnet på det objekt för vilket referenserna ska listas.

/class

Namnet på den klass för vilken referenser för samtliga objekt som tillhör klassen ska listas.

/hierarchy

Namnet på det objekt för vilket referenser för samtliga objekt som ligger under objektet i hierarkien och som tillhör en viss klass ska listas.

/name

Namnet på objekt för vilka korsreferenser ska listas.
Wildcard accepteras.

/output

Resultatet av operationen skrivs ut på den fil som anges.

/noterminal

Resultatet av operationen skrivs ej ut på terminalen.

delete

delete objects

Tar bort objekt ur databasen.
Objekten tas bort ett och ett. Fråga för bekräftelse av borttagningen och loggning sker får varje enskilt objekt. Objekt som har barn tas ej bort.

Syntax:

pwr>show objects /class= /hierarchy= /name=

/class

Namnet på den klassen eller en lista på klasser för vilken objekt ska tas bort.
Ex pwr> delete object/class=(ChanDi,ChanDo)

/hierarchy

Namnet på det objekt i hierarkin under under vilket alla objekt ska tas bort.

/name

Namn på objekt.
Wildcard är tillåtet.

/noconfirm

Bortagningen sker utan föregående fråga om objektet ska tas bort eller ej.

/nolog

Bortagning av objekt loggas ej på skärmen.

delete tree

Tar bort ett träd av objekt i databasen.

Objekten tas bort i klump. Fråga på bekräftelse av borttagningen fås endast för hela trädet, ej för varje enskilt objekt

/name

Namn på anfadern till objektsträdet.

/noconfirm

Bortagningen sker utan föregående fråga om objektetträdet ska tas bort eller ej.

/nolog

Bortagning av objekt loggas ej på skärmen.

disconnect

Bryter kopplingen mellan kanal och signalobjekt.

/source

Namn på ett av objekten.

document

Hanterar dokument.
Sidnumrerar och listar dokumenten i ett system.

· sidnumrering av dokumenten.
· utskrift av en lista på samtliga dokument i systemet.

Syntax:

pwr>document /repage
pwr>document /list /repage

Sidnumrering av dokument. Varje dokument ges en sidnumering bestående av dels en numrering av plcprogrammen och dels av dokumenten inom ett plcprogram (t ex 3.22). Samtliga dokument i systemet numreras. Sidnummer för ett dokumentobjekt ges av den ordning objekten ligger i databasen. Sidnumreringen lagras i Page attributet och kan även ändras från attribut editorn eller med 'set attribute' funktionen i pwr_cmd. Sidnumreringen sparas i databasen vid ett exit eller save kommando.

/list

Skriver ut en förteckning över dokumenten i ett system.

exit

Sparar gjorda ändringar i databasen och avslutar exekveringen.

export object

Exporterar ett objektsnamn till terminalfönster eller till en fil.

pwr> export object /name= [/prefix=] [/debugparameter] [/prooper] [/output=] [/append]

/name

Namn på objekt som ska exporteras.

/prefix

Text som läggs ut före objektsnamnet.

/debugparameter

Attributet som visas vid trace i PlcEdit adderas till objektsnamnet. Om det inte finns något trace attribut adderas ActualValue.

/prooper

Objekts eller attributenamnet konverteras till prooper syntax, dvs '-' substitueras med '$' och '.' med '$_'.

/output

Resultatet skrivs på fil.

/append

Om det redan finns en fil med aktuellt filnamn adderas resultatet till denna fil.

Exempel
pwr> export object /name=C1-Motor /prefix=M1$$ /debug /prooper

ger resultatet M1$$C1$Motor$_ActualValue.

list

Skriver ut listor på objekt och attribut. Listornas utseende beskrivs av listdescriptor objekt. För signaler, kanaler, hierarkier och plcprogram finns fördefinierade listobjekt.

list signals	Lista på signaler med korsreferenser.
list channels	Lista på kort och kanaler.
list hierarchy	Lista på hierarkier och plcpgm
list plcpgm	Lista på ett fönster i ett plcprogram.
list descriptor	Lista specificera av användaren med ett ListDescriptor objekt.

Listorna skickas till en printerkö som specificeras med symbolen PWR_FOE_PRINT tex

```
PWR_FOE_PRINT == "print/queue=laser_vu1"
```

Syntax:

```
pwr>list signals [/hierarchy=]
pwr>list channels [/node=]
pwr>list hierarchy [/hierarchy=]
pwr>list plcpgm [/hierarchy=] [/plcpgm=]
pwr>list descriptor /descriptor=
```

list signals

Lista på signaler samt korsreferenser till varje signal.

/hierarchy

Namn på ett hierakiobjekt under vilket signaler ska listas.

/output

Filbeskrivning om resultatet ska skrivas på fil istället för att skickas till printer.

list channels

Lista på kort och kanaler.

/node

Namn på den node för vilken kort och kanaler ska listas
Default listas samtliga noder.

/output

Filbeskrivning om resultatet ska skrivas på fil istället för att skickas till printer.

/volume

Namn på en eller flera volymer i vilka objeket ska listas.

/allvolumes

Objekt is samtliga volymer i databasen ska listas.

list hierarchy

Lista på hierarki och plcpgm objekt.

/hierarchy

Namn på ett hierakiobjekt under vilket signaler ska listas.

/output

Filbeskrivning om resultatet ska skrivas på fil istället för att skickas till printer.

/volume

Namn på en eller flera volymer i vilka objeket ska listas.

/allvolumes

Objekt is samtliga volymer i databasen ska listas.

list plcpgm

Lista på plc program.
För varje fönster listas:

- signaler som refereras i fönstret.

- objekt i andra plcprogram som refereras i fönstret.
- objekt i fönstret som refereras i andra plcprogram.

/hierarchy

Namn på ett hierakiobjekt under vilket signaler ska listas.

/plcpgm

Namn på ett plcpgm som ska listas.

/output

Filbeskrivning om resultatet ska skrivas på fil istället för att skickas till printer.

/volume

Namn på en eller flera volymer i vilka objeket ska listas.

/allvolumes

Objekt is samtliga volymer i databasen ska listas.

list descriptor

Lista på objekt och attribut specificerade med ett ListDescriptor objekt.

/descriptor

Namn på det ListDescriptor objekt som specificerar listan.

/hierarchy

Namn på ett hierakiobjekt under vilket objekt ska listas.

/object

Namn på ett objekt som ska listas.

/output

Filbeskrivning om resultatet ska skrivas på fil istället för att skickas till printer.

/volume

Namn på en eller flera volymer i vilka objeket ska listas.

/allvolumes

Objekt is samtliga volymer i databasen ska listas.

move

move object

Flytta ett objekt eller ändra namn på ett objekt.

/destination

Plats dit aktuellt objekt ska flyttas. Namn på förälder eller syskon. Om /last eller /first anges kommer objektet att läggas som barn till destinations objektet, om /before eller /after som syskon.

/source

Objekt som ska flyttas.

/rename

Om objekts namnet, dvs sista ledet i namnet, ska ändras kan ett nytt namn anges här. Om /destination utelämnas ändras enbart namnet på objektet.

/first

Objektet läggs som första barn till destinations objektet.

/last

Objektet läggs som sista barn till destinations objektet.

/after

Objektet läggs som syskon efter destinations objektet

/before

Objektet läggs som syskon före destinations objektet.

Exempel

pwr> move object /destination=ugn-motor /source=ung-kran-start /first
pwr> move object /source = ugn-motor-start /rename=stopp

print

Hanterar utskrift av dokument.
· utskrift av samtliga dokument i ett plcprogram.

· utskrift av samtiliga dokument i alla plcprogram under ett objekt i anläggningshierarkien.

Syntax:

pwr>print /plcpgm=
pwr>print /hierarchy=

/plcpgm

Namnet på det plcpgm objekt för vilket dokumenten ska skrivas ut.

/hierarchy

Namnet på ett objekt i anläggningshierarkien.
Samtliga dokument i alla plcprogram under ett objekt i anläggningshierarkien skrivs ut.

save

Sparar gjorda ändringar i databasen.

set

set attribute

Gör det möjligt att sätta värden på attribut i ett eller ett antal objekt.

De objekt som ska datasättas specificeras med kvalifierarna /name /class och /hierarchy. Om samma värde ska tilldelas samtliga objekt ges detta med kvalifieraren /value. Om denna utelämnas frågas efter ett nytt värde för varje enskilt objekt.

Datasättningen sparas i databasen vid ett exit eller save kommando.

Syntax:

pwr>set attribute /attribute= /name= /hierarchy= /class=
pwr>set attribute /attribute= /value= /name= /hierarchy= /class=

/attribute

Namn på det attribut som ska datasättas.

/value

Värdet som ska tilldelas attributet. Om /value utelämnas frågas efter ett värde för varje enskilt objekt.

/class

Väljer ut objekt av en viss klass.

/hierarchy

Namnet på ett objekt för vilket samtliga objekt som ligger under objektet i hierarkien ska påverkas av setkommandot.

/name

Namnet på objekt för vilka ett attribut ska sättas.
Wildcard accepteras.

Om wildcard används måste kvaliferaren /class ges.

/log

Loggar alla gjorda datasättningar på terminalen eller på fil.

/noconfirm

Datasättningen sker utan föregående fråga om datasättningen ska genomföras eller ej.

/output

Resultatet av operationen skrivs ut på den fil som anges.

/noterminal

Resultatet av operationen skrivs ej ut på terminalen.

set db

Knyter pwr_cmd databasen med angivet id.
En databas får inte redan ha öppnats för att man ska kunna knyta sig till en databas.

/dbid

Id för databasen.

set message

Sätter av och på utskrift av felmeddelanden.

/on

Sätter på utskrift av felmeddelanden.

/off

Stänger av utskrift av felmeddelanden.

set volume

Knyter pwr_cmd till en volym.

/volumename

Namn på volymen.

set template

Sätt defaultvärden på antal segment av namnet för kanaler och signaler som ska visas i plc-dokument.

/signalobjectseg

Antal segment i signalnamnet.

/shosigchancon

Anger om kanalen ska visas eller ej.

/shodetecttext

Anger om larmtexten i ASup och DSupobjekt ska visas eller ej.

set verify

Sätter verify på. Varje rad i kommandofiler som exekveras kommer att visas.

set noverify

Stänger av verify.

show

Show visar diverse information om systemet.

show node

Listar samtliga noder i systemet.

Syntax:
pwr>show node

show plcpgm

Listar samtliga noder i systemet och de plcpgm som hör till respektive node.
Syntax:

pwr>show plcpgm

show window

Listar samtliga windows i ett plcpgm.

Syntax:
pwr>show window/plcpgm=

/plcpgm

Hierarkinamnet på det plcpgm-objekt för vilket window-objekten ska listas.

show objects

Visar
· vilka objekt som finns i ett fönster.
· vilka objekt som finns i systemet av en viss klass, med ett viss namn och/eller under en viss hierarki.
· Innehållet i ett objekt.

· objektsnamn för ett objdid.

Syntax:
pwr>show objects/window=
pwr>show objects/objid=
pwr>show objects/class=
pwr>show objects/class= /hierarchy= /name= /full /window

Hierarkinamnet på det window-objekt för vilket objekten ska listas.

/objid

Objid på det objekt för vilket objektsnamnet ska visas.

/class

Namnet på den klassen eller en lista på klasser för vilken alla instanser ska listas.
Ex pwr> show object/class=(And,Or)

/hierarchy

Namnet på det objekt i hierarkin under för vilken alla objekt av en viss klass ska listas.

/name

Namn på objekt.
Wildcard är tillåtet.

/allvolumes

Anger att sökning efter objekt ska ske i samtliga volymer i aktuell databas.

/full

Innehållet i attribut som är datasatta från attributeditorn skrivs ut.

/parameter

Skriver ut innehållet i angivet attribut. Lista på attribut är tillåtet.
Kan endast användas tillsammans med kvalifierarna /name, /class eller /hierarchy. Om wildcard används i namnet krävs att klassen anges.

Ex
pwr>show object/class=ChanDi/parameter=(Identity, Description)

/output

Skriver ut resultatet av operationen på angiven fil.

/noterminal

Skriver ej ut resultatet av operationen på terminalen.

show connections

Listar samtliga connection-object i ett window.
Syntax:

pwr>show connections/window=

/window

Hierarkinamnet på det window-objekt för vilket connections-objekten ska listas.

show modules

Visar information om objectsmodulerna som skapas vid kompilering av plcpgm och windows.

Syntax:
pwr>show modules
pwr>show modules/name=
pwr>show modules/objdid=

/name

Hierarkinamnet på det window-objekt eller plcpgm för vilket info ska visas.

/objdid

Objdid på det window-objekt eller plcpgm för vilket info ska visas.

/output

Resultatet av operationen skrivs ut på den fil som anges.

show class

Visar alla klasser i en klassvolym. Default klassvolym är pwrb:

Syntax:
pwr>show class
pwr>show class/classhier=uuu

/classhier

Namnet på den klasshierarki (objekt av klassen $ClassHier) för vilkan all klasser ska visas.

/name

Namn på klass.
Wildcard accepteras.

/output

Resultatet av operationen skrivs ut på den fil som anges.

/noterminal

Resultatet av operationen skrivs ej ut på terminalen.

/full

Skriver ut namnet på alla attribut i klassen, samt typ
och storlek på attributet.

/all

Listar alla klassobjekt, dvs även body-objekt och
attributobjekt.

/contents

Skriver ut innehållet i ett klassobjekt.

show hierarchy

Visar samtliga hierarkiobjekt (av klassen $Planthier) i systemet.

Syntax:
pwr>show hierarchy

/noterminal

Resultatet av operationen skrivs ej ut på terminalen.

/output

Resultatet av operationen skrivs ut på den fil som anges.

sort

Sorterar objekt i en hierarkinivå i bokstavsordning.
Objekt av klassen $Planthier läggs först, sedan PlcPgm objek och sedan övriga objekt. Med qualifieran /class ordnas dessutom signalerna efter klasstillhörighet i ordningen Ai, Ao, Av, Co, Di, Do, Dv.

/parent

Förälder till objekt som ska sorteras.

/class

Sorterar objekt efter klasstillhörighet.

/signals

Sorterar signaler efter klass och övriga objekt i bokstavsordning.

quit

Avslutar utan att spara.

wb

Laddar in en textfil i databasen, eller skriver ut innehållet
i databasen på en textfil.

Syntax:

```
pwr>wb load/loadfile= /output= [/ignore][/full]
pwr>wb dump/output= [/hierarchy=][/volume=][/ignore][/full]
pwr>wb import/loadfile= /root= /output= [/ignore][/full]
pwr>wb export/ouput= /hierarchy=
```

wb dump

Dumpar databasen eller en del av databasen på en textfil.

Dumpning av databasen kan göras när man har ändrat i någon klass-volym, när vill flytta objekt från ett system till ett annat, eller när man vill slå ihop två system.

Vid versionsbyte av Proveiw/R krävs i vissa fall också dumpning av databasen.

Textfilen kan återladdas med 'wb load' kommandot.
Med /volume eller /hierarchy kan vissa delar av databasen väljas ut.
Default kommer samtliga volymer utom klassvolymer att dumpas.

Syntax:

pwr>wb dump /output= [/hierarchy=] [/volume=][/indent=][/header]

/output

Filnamn för dumpfilen,

/volume

Namn på en eller flera volymer som ska dumpas. Hela volymerna (inklusive själva volymsobjektet) kommer att dumpas.

/header

En header på tre rader som innehåller objektnamnet skrivs ut före varje objekt..

/indent

Antal blanktecken i marginalen i textfilen för varje ny objektsnivå (default 3).

wb load

Laddning av objekt beskrivna i en textfil till databasen.

/loadfile

Namn på textfilen som ska laddas.

/output

Namn på fil som innehåller resultatet av laddningen och eventuella felmeddelanden.

/full

Outputfilen innehåller listning av alla rader i laddfilen.

/ignore

Även om fel upptäcks vid laddningen sparas framgångsrika operationer i databasen.

/noindex

Anger att objid som finns i textfilen ej ska laddas in utan att objekten ska ges nya objid. Denna kvaliferare ska användas om det finns risk att en objid i textfilen kolliderar med en existerande objid, t ex när man slår ihop två volymer eller kopierar objekt från mellan volymer.

wb export

Skriver objekt i databasen under en hierarki på en textfil.
Hierarkin kan återladdas under ett annat namn, eller i en annan databas.

Syntax:

```
pwr> wb export /output= [/hierarchy=] [/indent=][/header]
```

/output

Filnamn för dumpfilen,

/hierarchy

Namn på rot-objektet i den hieraki som ska exporteras.

/header

En header på tre rader som innehåller objektnamnet skrivs ut före varje objekt..

/indent

Antal blanktecken i marginalen i textfilen för varje ny objektsnivå (default 3).

wb import

Laddar en fil skapad med 'wb export'.

/root ska innehålla namnet på rot-objektet.
Föräldern till rotobjektet måste existera, man själva rotobjektet skapas vid laddningen.

Om syntax fel ges laddas ej innehållet i databasen.

/loadfile

Namn på textfilen som ska laddas.

/output

Namn på fil som innehåller resultatet av laddningen och eventuella felmeddelanden.

/full

Outputfilen innehåller listning av alla rader i laddfilen.

/ignore

Även om fel upptäcks vid laddningen sparas framgångsrika operationer i databasen.

Script

Pwr_cmd-script är ett sätt att programmera pwr_cmd-kommandon. Scripthanteraren ger dessutom möjlighet till att göra beräkningar, utföra vilkors-satser, loop-satser, deklarerara varibler och funktioner.

Ett script startas med '@' följt av scriptfils namnet och eventuella argument.

Exempel

```
pwr> @my_script
```

Programsatser

Programsatserna han innehålla nyckelord, operatorer, variabler och uttryck. Varje programstats tillhör någon av följande kategorier:

- Deklarationssatser, som namnger variabler och funktioner.
- Uttryck, som utför beräkningar, funktionsanrop och tilldelningar.
- Villkors- loop- och hopp-satser, som utför villkorlig exekvering eller hopp i programkoden.
- Pwr_cmd-kommandon, skickas vidare till pwr_cmd´s kommandotolk efter eventuell substitution av variabler.

Kommentarsrader skrivs med ett utrops-tecken i första positionen.
Om en sats behöver delas upp på flera rader markeras radbrytningen med backslash som den brutna raden sista tecken. Det får inte finnas några tecken till höger om backslash-tecknet.
Maximalt antal tecken i en sats är 159.
Deklarationssatser för variabler, samt uttryckssatser ska alltid avslutas med semikolon.

Include

En script-fil innehållande funktioner kan inkluderas med #include satsen.
Default filtyp är '.pwr_com'.

Exempel

```
#include <my_functions>
```

Datatyper

Datatyper för variabler, konstanter och funktioner är int, float och string:

- int heltalsvärde.
- float 32-bitars flyttalsvärde.
- string sträng med 80 tecken (null-terminerad).

En variabel kan deklareras i tre olika namn-tabeller:

- local variabeln är känd inom en funktion.
- global variabeln är känd av alla funktioner inom ett script (med includefiler).
- extern variabeln är känd inom alla script-filer som exekveras i en pwr_cmd-session.

Deklaration av variabler

Alla variabler som används i ett script måste vara deklarerade. En variabel deklaration måste ligga inom en funktion eller inom main. Övriga deklarationer kommer att ignoreras.

Den variabeldeklaration består av

- namn-tabell (global eller extern, om namntabellen är local anges inte detta).
- datatyp (int, float eller string).
- variabelnamn.
- Om variabeln är en vektor anges antal element omgivet av hakparenteser.
- Likamed-tecken följt av initialvärde, om detta ej anges sätts värdet till 0 eller null-string. Vektorer kan inte ges ett initalvärde.
- semikolon.

Variabelnamnet ska inledas av ett alfabetiskt tecken eller och kan följas av alfabetiska eller alfanumeriska tecken eller understreck '_'. Max 32 tecken.

Exempel

```
int             i;
float           flow = 33.4;
string          str = "Hello";
string          v[10];
extern  int     jakob;
global  float   ferdinand = 1234;
```

Variabler i uttryck

Variabler som används i uttryck måste vara deklarerade. Om variablen är av typen vektor kan endast element hanteras i ett uttryck eller ett funktionsanrop, dvs elementindex måste anges. Element-index indexeras från 0 och uppåt. Som element-index kan en integer-variabel anges, men ett uttryck som element-index är ej tillåtet.

Exempel

```
int   i;
float          a;
float          b[3];
float c[3];

c[0] = 6 * a;
for ( i = 0; i < 3; i++)
  b[i] = c[0] + i;
endfor
```

Datatypkonvertering

Om ett uttryck består av variabler, konstanter och funktioner av olika datatyper kommer variablerna att konverteras med företrädesordningen string, float, int, dvs om två operander är av typen float och string, eller int och string, kommer resultatet att bli string. Vid en tilldelning kommer värdet av uttrycket som variabeln ska tilldelas att konverteras till variabelns datatyp. Detta gäller även konvertering från string till int och från string till float om detta är möjligt.

Exemel

```
string  str;
int     i = 35;
str = "Luthor" + i;
```

Värdet i str kommer att bli "Luthor35".

```
float   f;
string  str = "3.14";
int     i = 159;
f = str + i;
```

Värdet i f kommer att bli 3.14159

Konstanter

Konstanter är värden som förekommer i uttryck och variabeldeklarationer. Konstanter är av typen int, float eller string.

int

En int-konstant består av alfanumeriska tecken samt ev inledande minustecken.
Exempel
0, 123, -32.

float

En float-konstant består av alfanumeriska tecken, en punkt, samt ev inledande minustecken. Exponent är ej ännu implementerat.

Exempel
1.2, -12.98.

string

En string-konstant består av en teckensträng omgiven av dubbelapostrof.
Max längd är 80 tecken.
Exempel
”Motorn har startat”

Externa system-variabler

Det finns att antal externa variabler som är tillgängliga:

cmd _os
Innehåller aktuell plattform (”os_vms”). Datatyp string.

cmd_hw
Innehåller aktuell plattform (”hw_vax” eller ”hw_axp”). Datatyp string.

cmd_status
Innehåller retur-status från senaste pwr_cmd-kommando. Om resultatet är lyckat är cmd_status = 1, annars en jämn felkod. Datatyp int.

Uttryck

Ett uttryck består av operander och operatorer. När det står som en enskild sats avslutas det med semikolon. Uttryck förekommer även i if-, for- och while-statser.

Operander

Operander är variabler, konstanter eller funktionsanrop. Variabler och funktioner måste vara deklarerade innan de används i ett uttryck.

För vektorer gäller att endast ett element kan hanteras i uttrycket, inte hela vektorer.

Operatorer

Operatorerna har samma funktion som i c, med vissa begränsningar. Alla c-operatorer är inte implementerade. Vissa operatorer kan till skillnad från c även operera på strängar (+,=,==,!=). Prioriteten är samma som i c.

Operator	Beskrivning	Datatyper
+	addition	int, float, string
-	subtraktion	int, float
*	multiplikation	int, float

/	division	int, float
++	inkrement, endast postfix	int, float
--	dekrement, endast postfix	int, float
>>	högerskift	int
<<	vänsterskift	int
<	mindre än	int, float
>	större än	int, float
<=	mindre eller lika med	int, float
>=	större eller lika med	int, float
==	lika med	int, float, string
!=	ej lika med	int, float, string
&	bitvis och	int
|	bitvis eller	int
&&	logisk och	int
||	logisk eller	int
!	logisk ej	int
=	tilldelning	int, float, string
+=	addition och tilldelning	int, float, string
-=	subtraktion och tilldelning	int, float
&=	logisk och och tilldelning	int
|=	logisk eller och tilldelning	int

Funktioner

Deklaration

Funktioner kan vara av datatyperna int, float eller string.
Funktioner består av en deklaration, följt av diverse satser avslutningsvis nyckelorder ‘endfunction’. En speciell sats för funktioner är return, som tilldelar funktionen ett värde och överlämnar exekveringen till anroparen

En funktionsdeklaration består av följande:

- Nyckelorder ‘function’.
- Funktionens datatyp.
- Namn på funktionen.
- En argumentlista, omgiven av parenteser, med argumenten separerade med kommatecken. Varje argument i argumentlistan innehåller datatyp och variabelnamn. En deklaration i argumentlistan för ej en vektor.

Exempel

```
function string create_message( string name1, string
name2)
  string msg;
```

```
  msg = ”Some message to ” + name1 ” and ” + name2;
  return msg;
endfunction
```

Variablerna i argumentlistan läggs i den lokala variabel-tabellen. Argumenten kan fungera både som in- och ut-data till funktionen. Vid returnering från funktionen kommer aktuella värden på argument-variablerna att överföras till motsvarande variabler hos anroparen.

Anrop

Anrop av funktionen sker i ett uttryck.
Argumenten i anropet kan vara variabler eller konstanter, däremot ej uttryck.

Exempel

```
string name;
string         msg;

name = ”Erik”;
msg = create_message( name, ”Kalle”);
```

main

Exekveringen av ett script startar alltid i huvud-funktionen ‘main’.
Huvudfunktionen inleds med satsen ‘main()’ och avslutas med satsen ‘endmain’. Om main-endmain satserna inte finns i sciptet startas exekveringen vid början på filen och avslutas vid slutet på filen. I detta fallet får inte filen innehålla några funktionsdeklarationer.
main och endmain satserna ska *inte* avslutas med semikolon.

Exempel

```
main()
  int i;
  string name;

  for ( i = 0; i < 10; i++)
    name = ”motor-obj” + i;
    create object/name=’name’
  endfor
endmain
```

Argument

Argument som skickas till med vid exekvering av scriptfilen läggs i de globala variablerna p1-p8.

Villkors-satser

if-else-endif

if-satsen innehåller ett uttryck som är sant eller falskt. Om uttrycket är sant kommer satserna mellan if och endif att exekveras annars sker ett hopp till endif-satsen. Om det finns en else-sats mellan if och endif sker hoppet till else-satsen istället.

Uttrycket i if-satsen ska omgärdas av parenteser.
if, else och endif satserna ska *inte* avslutas med semikolon.
Flera nivåer av if-else-endif är tillåtet.

Exempel

```
if ( i < 10 && i > 5)
  a = b + c;
endif

if ( i < 10)
  a = b + c;
else
  a = b - c;
endif
```

Loop-satser

Loopsatserna är av typen for eller while.

for-endfor

for-loopen inleds med en for-sats och avslutas med en endfor-sats.
for-satsen innehåller tre uttryck. Det första exekveras före första loopen, det tredje exekveras före de efterföljande looparna, och mitten-uttrycket evalueras före varje loop. Om uttrycket är sant sker ytterligare en loop annars fortsätter exekveringen efter endfor-satsen.
Flera nivåer av end-enfor är tillåtna. Med continue- eller break-satserna kan man hoppa till nästa loop eller hoppa ur loopen.

Exempel

```
for ( i = 0; i < 10; i++)
  a += b;
endfor
```

while-endwhile

while-loopen inleds med en while-sats och avslutas med en endwhile-sats.
while-satsen innehåller ett uttryck som evalueras före varje loop, så länge uttrycket är sant pågår loopningen.
Flera nivåer av while-endwhile är tillåtna. Med continue aller break-satserna kan men hoppa till nästa loop eller hoppa ur loopen.

Exempel

```
while ( i < 10)
  i++;
endwhile
```

break

En break-sats kommer att orsaka att exekveringen forsätter efter närmaste efterföljand endfor eller endwhile-stats. break-satsen ska avslutas med semikolon.

Exempel

```
for ( i = 0; i < 10; i++)
  a += b;
  if ( a > 100)
    break;
endfor
```

continue

En continue-sats kommer att orsaka att exekveringen av en loop forsätter vid närmast föregående for- eller while-sats.

Exempel

```
for ( i = 0; i < 10; i++)
  b = my_function(i);
  if ( b > 100)
    continue;
  a += b;
endfor
```

Hoppsatser

goto

goto-satsen kommer att orsaka ett hopp till en rad definierad av en label.
En label-sats är ett namn avslutat med kolon.

Exempel

```
  b = attribute("MOTOR-ON.ActualValue", sts);
  if (!sts)
    goto some_error;
  ...
some_error:
  say("Something went wrong!");
```

pwr_cmd-kommandon

Samtliga pwr_cmd-kommandon är tillgängliga i script-koden.
En pwr_cmd-kommandorad ska *inte* avslutas med semikolon.
Variabler kommer att substitueras i kommandoraden om de omgärdas av enkel-apostrofer.
Det externa vaiableln cmd_status innehåller retur-status från senaste kommando.

Exempel

```
  string name;
  string par_name;
  string root_name;
  int j;
  int i;

  root_name = ”vkv_test”;
  for ( i = 0; i < 3; i++)
    name = ”Obj” + (i+1);
    par_name = root_name + ”-” + name;
    crea obj/sour='root_name'/nam=name/class=table/first
    if ( cmd_status != 1)
      printf( ”Objekt %s kunde ej skapas\n”,par_name);
      exit();
    endif
    for ( j = 0; j < 3; j++)
      name = "-Obj" + (j+1);
      crea
obj/sour=’par_name’/nam='name'/class=table/first
    endfor
  endfor
```

Sammanfattning

Syntaxen är ganska lik c.Här följer en lista på några saker som skiljer sig från c.

- Måsvinge-parenteser används ej. Villkors-satser, loopar och funktioner avgränsas av nyckelord, tex if-endif, for-endfor, while-endwhile, function-endfunction.
- Om en sats delas upp på flera rader ska radbrytningen med backslash.
- Det finns bara tre datatyper: int, float och string.
- Arrayer kan endast ha en dimension.
- Argument som skickas med en function, kan inte vara ett uttryck, endast en variabel eller in konstans är tillåtet.
- Om ett argument till en funktion, ändras i funktionen, förs ändringen över till anroparen.
- Index i en array kan inte vara ett uttryck, endast en variabel eller en konstant är tillåtet.
- Satser utan nyckelord och utan semikolon tolkas som pwr_cmd-kommandon.
- Eventuella argument som skickas med vid exekvering av filen, lagras i p1-p9.
- Sammanslagning och jämförelse av strängar sker med ‘+’ resp ‘==’ operatorerna.

- Ytterligra några saker som inte finns är pekare, struct, enum, switch, typedef, prekompilatorsatser (förutom #include).

Inbyggda funktioner

Scriptfilen kan anropa ett antal inbyggda funktioner för att hantera utskrifter, inmatning, filer, strängar, hämta attribut och söka efter objekt i databasen mm.

In och utmatning

Funktion	Beskrivning
ask	Skriver ut en fråga och läser in ett svar
say	Skriver ut en sträng
printf	Formaterad utskrift
scanf	Formaterad inläsning

Filhantering

Funktion	Beskrivning
fclose	Stäng en fil
felement	Hämta ett element ur den med fgets senaste lästa raden.
fgets	Läsning av en rad från fil
fopen	Öppna en fil
fprintf	Formaterad skrivning på fil
fscanf	Formaterad läsning från fil

Hantering av strängar

Funktion	Beskrivning
edit	Rensa bort space och tabbar i början och i slutet av en sträng, samt ta bort multipla space och tabbar i strängen
element	Hämta ett element i en sträng
extract	Hämta ett antal tecken i en sträng
sprintf	Formaterat skrivning i en sträng-variabel.
strchr	Leta efter första förekomsten av ett tecken i en sträng
strlen	Längden av en sträng
strrchr	Leta efter sista förekomsten av ett tecken i en sträng
strstr	Leta efter första förekomsten av en teckensekvens i en sträng
toupper	Konvertera till versaler

Databas funktioner

Funktion	Beskrivning
CutObjectName	Hämta de sista segmenten i ett objektsnamn
GetAttribute	Hämta ett attribut
GetChild	Hämta första barnet till ett objekt

GetNextSibling	Hämta nästa syskon till ett objekt
GetNextVolume	Hämta nästa volym
GetParent	Hämta förälder till ett objekt
GetObjectClass	Hämta klassen till ett objekt
GetRootList	Hämta första objekt i rot-listan
GetVolumeClass	Hämta klassen för en volym

System funktioner

Funktion	**Beskrivning**
exit	Avsluta exekveringen av ett skript
time	Hämta systemtiden
system	Exekvera ett DCL-kommando
verify	Sätt verify på eller av

ask()

int ask(string question, (arbitrary type) answer)

Beskrivning

Skriver ut en fråga och läser in ett svar.
Returnerar antal lästa tecken.

Argument

string	question	Fråga som skrivs ut.
arbitrary type	answer	Inläst svar. Returnerad. Kan vara int, float eller string.

Exempel

```
 string  answer;

 ask( ”Do you really want to continue ? (Y/N) ”,
answer);
 if ( !(answer == ”y” || answer == ”Y”))
   exit();
 endif
```

CutObjectName()

string CutObjectName(string name, int segments)

Beskrivning

Ta bort de första segmenten i ett objektsnamn.
Returnerar de sista segmenten. Antalet segment anges av argumentet ‘segments’.

Argument

string	name	Objektsnamn (path-name).
int	segments	Antal segment som ska returneras.

Exempel

```
string path_name;
string object_name;

path_name = GetChild("Rt-Motor");
object_name = CutObjectName( path_name, 1);
```

GetAttribute()

(variable type) GetAttribute(string name [, int status])

Description

Returnerar värde i det angivna attributet. Typen på det returnerade värde är beroende på attributets datatyp. Värdet kommer att konverteras till int, float eller string.

Argument

string name namn på attribut som ska hämtas.
int status status på operation. Återlämnat. Om 0 kunde inte attributet hämtas. Optional.

Exempel

```
  int alarm;
  int sts;

  alarm = GetAttribute("Roller-Motor
Alarm.ActualValue");
  on = GetAttribute("Roller-Motor-On.ActualValue", sts);
  if ( !sts)
    say("Could not find motor on attribute!");
```

GetChild()

string GetChild(string name)

Beskrivning

Hämtar första barnet till ett objekt. De övriga barnen kan hämtas med GetNextSibling(). Returnerar namnet på barnet. Om det inte finns något barn returneras en null-sträng.

Argument

string name name på objektet.

Exempel

```
string child;
child = GetChild("Roller-Motor");
```

GetNextSibling()

string GetNextSibling(string name)

Beskrivning

Hämtar nästa syskon till ett objekt.
Returnerar namnet på syskonet. Om inget nästa syskon finns, returneras en null-sträng.

Argument

string name namn på objekt.

Exempel

```
string name;
int not_first;

name = GetChild("Rt");

while ( name != "")
  printf( ”Found object: %s\n”, name);
  name = GetNextSibling(name);
endwhile
```

GetNextVolume()

string GetNextVolume(string name)

Beskrivning

Hämtar nästa volym i volymslistan. Första volymen hämtas med GetVolumeList()
Returnerar namnet på volymen. Om ingen nästa volym, returneras en null-sträng.

Argument

string	name	namn på volym.

GetParent()

string GetParent(string name)

Beskrivning

Hämta föräldern till ett objekt.
Returnerar namnet på föräldern. Om det inte finns någon förälder returneras en null-sträng.

Argument

string name name på objektet.

Exempel

```
string parent;

parent = GetChild("Roller-Motor");
```

GetObjectClass()

string GetObjectClass(string name)

Beskrivning

Hämtar klassen för ett objekt.
Returnerar namnet på klassen.

Argument

string name namn på objektet.

Exempel

```
string class;

class = GetObjectClass("Roller-Motor");
```

GetRootList()

string GetRootList()

Beskrivning

Hämta första objektet i rot-listan.
Returnerar namnet på rot-objektet. Nästföljande objekt i rot-listan kan hämtas med GetNextSibling().

Exempel

```
string name;

name = GetRootList();
while( name != "")
  printf( "Root object found: %s", name);
  name = GetNextSibling(name);
endwhile
```

GetVolumeClass()

string GetVolumeClass(string name)

Beskrivning

Hämtar klassen för ett volym.
Returnerar namnet på klassen.

Argument

string name namn på volym.

Exempel

```
string class;

class = GetVolumeClass("CVolVKVDKR");
```

GetVolumeList()

string GetVolumeList()

Beskrivning

Hämta första objektet i volyms-listan.
Returnerar namnet på volymen. Nästföljande volym i volym-listan kan hämtas med GetNextVolume().

Exempel

```
string name;

name = GetVolumeList();
while( name != "")
  printf( "Volume found: %s", name);
  name = GetNextVolume(name);
endwhile
```

edit()

string edit(string str)

Beskrivning

Tar bort inledande och avslutande tabbar och space, och ersätter multipla tabbar och space med enstaka space.
Returnerar den editerade strängen.

Argument

string	str	sträng som ska editeras.

Exempel

```
collapsed_str = edit(str);
```

element()

string element(int number, string delimiter, string str)

Beskrivning

Extraherar ett element från en sträng av element.
Returnerar det extraherade elementet.

Argument

int	number	nummer på element som ska extraheras.
string	delimiter	avgränsnings-tecken.
string	str	sträng med element.

Exempel

```
string str = "mary, lisa, anna, john";
string elem1;
elem1 = elment( 1, ",", str);
```

exit()

int exit()

Beskrivning

Avslutar exekveringen av ett script.

Exempel

```
exit();
```

extract()

string extract(int start, int length, string str)

Beskrivning

Extraherar ett antal tecken ur en sträng.
Returnerar de extraherade tecknen som en sträng.

Argument

int	start	start position för det första tecknet.
int	length	antal tecken som ska extraheras.
string	str	sträng från vilken tecken ska extraheras.

Exempel

```
extracted_str = extract( 5, 7, str);
```

fclose()

int fclose(int file)

Beskrivning

Stänger en öppnad fil.

Argument

int file fil-id returnerad av fopen.

Exempel

```
int infile;
infile = fopen("some_file.txt","r");
...
fclose( infile);
```

felement()

string felement(int number, string delimiter)

Beskrivning

Extraherar ett element från den med fgets senast lästa raden. felement gör det möjligt att hangera fil-rader längre än 80 tecken.
Returnerar det extraherade elementet.

Argument

int	number	nummer på element som ska extraheras.
string	delimiter	avgränsnings-tecken.

Exempel

```
while ( fgets( str, file))
  str1 = felement(2, ”/”);
  str2 = felement(3, ”/”);
  printf( ”str1: %s\nstr2: %s\n”, str1, str2);
endwhile
```

fgets()

int fgets(string str, int file)

Beskrivning

Läser en rad från en specificera fil.
Returnerar 0 om filslut.

Argument

string	str	Läst rad. Returnerad.
int	file	fil-id returnerad av fopen().

Exempel

```
file = fopen("some_file.txt","r");
while( fgets( str, file))
  say( str);
endwhile
fclose( file);
```

fopen()

int fopen(string filespec, string mode)

Beskrivning

Öppnar en fil för läsning eller skrivning.
Returnerar en fil-identitet. Om filen inte kan öppnas, returneras 0.

Argument

string	filespec	Namn på fil.
string	mode	Access mod.

Exempel

```
int infile;
int outfile;

infile = fopen("some_file.txt","r");
outfile = fopen("another_file.txt","w");
if ( !infile || !outfile)
  say(”Kan ej öppna fil”);
  exit();
endif
…
fclose( infile);
fclose( outfile);
```

fprintf()

int fprintf(int file, string format [, (arbitrary type) arg1…)

Beskrivning

Formaterad utskrift på fil. C-syntax. Fil samt format argument samt ytterligare argument av godtyckligt antal och av godtycklig typ.
Returnerar antal utskrivna tecken.

Argument

int	file	Fil-id returnerat av fopen.
string	format	Uttskifts format.
arbitrary type	arg	Värde argument. Optional. Godtyckling antal. Kan vara int, float eller string.

Exempel

```
int outfile;
 outfile = fopen( "my_file.txt", "w");
 if (!outfile)
   exit();
 fprintf( outfile, "Some text\n");
 fprintf( outfile, "a = %d\n", a);
 fclose( outfile);
```

printf()

int printf(string format [, (arbitrary type) arg…)

Beskrivning

Formaterad utskrift. C-syntax. Format argument samt ytterligare argument av godtyckligt antal och av godtycklig typ.

Returnerar antal utskrivna tecken.

Argument

string	format	Uttskifts format.
arbitrary type	arg1	Värde argument. Optional. Godtyckligt antal. Kan vara int, float eller string.

Exempel

```
printf( "Watch out!\n");
printf( "a = %d", a);
printf( "a = %d och str = %s\n", a, str);
```

scanf()

int scanf(string format , (arbitrary type) arg1)

Beskrivning

Formaterad läsning. C-syntax
Returnerar antal lästa tecken.

Argument

string	format	Format.
arbitrary type	arg1	Värde argument. Returnerad. Kan vara int, float eller string.

Exempel

```
scanf( "%d", i);
```

sprintf()

int sprintf(string str, string format [, (arbitrary type) arg1, (arbitrary type) arg2])

Beskrivning

Formaterad utskrift till en sträng-variabel. C-syntax. Format argument samt upp till två ytterligare argument.
Returnerar antal utskrivna tecken.

Argument

string	str	Sträng som ska skrivas i.
string	format	Uttskifts format.
arbitrary type	arg1	Värde argument. Optional. Kan vara int, float eller string.
arbitrary type	arg2	Värde argument. Optional. Kan vara int, float eller string.

Exempel

```
sprintf( str, "Object%02.2d", nr);
```

strchr()

int strchr(string str, str character)

Beskrivning

Letar efter första förekomsten av ett tecken i en sträng.
Returnerar positionen för tecknet. Om tecknet ej hittas returneras 0.

Argument

string	str	Sträng i vilken man söker efter ett tecken.
string	character	Tecken som ska sökas efter.

Exempel

```
int    pos;
string str;
string name;

pos = strchr( str, ”-”);
pos--;
name = extract( 1, pos, str);
```

strlen()

int strlen(string str)

Beskrivning

Returnerar längden på angiven sträng.

Argument

string	str	Sträng för vilken längden returneras.

Exempel

```
int    len;
string str;

len = strlen( str);
```

strrchr()

int strchr(string str, str character)

Beskrivning

Letar efter sista förekomsten av ett tecken i en sträng.
Returnerar positionen för tecknet. Om tecknet ej hittas returneras 0.

Argument

string	str	Sträng i vilken man söker efter ett tecken.
string	character	Tecken som ska sökas efter.

Exempel

```
int    pos;
string str;
string name;

pos = strrchr( str, ”-”);
pos++;
name = extract( pos, 30, str);
```

strstr()

int strstr(string s1, str s2)

Beskrivning

Letar efter första förekomsten av en sekvens av tecken (s2) i en sträng (s1). Returnerar positionen för tecknet. Om tecknet ej hittas returneras 0.

Argument

string	s1	Sträng i vilken man söker efter teckensekvensen.
string	s2	Teckensekkvens som ska sökas efter.

Exempel

```
int    pos;
string str;
string name;

pos = strstr( str, ”Allan”);
name = extract( pos, 10, str);
```

time()

string time()

Beskrivning

Returnerar aktuell tid i strängformat.

Exempel

```
string t;
 t = time();
```

say()

int say(string text)

Beskrivning

Utmating av en sträng.
Returnerar antalet utskrivna tecken.

Exempel

```
say(”—- Evaluating the data…” );
```

system()

int system(string command)

Beskrivning

Skickar en given sträng till operativsystemets kommandotolk.
Returnerar returstatus.

Exempel

```
system(”purge sys$login:*.dat”);
```

toupper()

string toupper(string str)

Beskrivning

Konverterar en sträng till versaler.
Returnerar den konverterade strängen.

Argument

string	str	Sträng som ska konverteras till versaler.

Exempel

```
upstr = toupper( str);
```

verify()

int verify([int mode])

Beskrivning

Sätter eller visar verify-mod. Om verifikationsmoden är 1 visas alla exekverade rader på skärmen.
Returnerar den nuvarande verify-moden.

Argument

int	mode	verification till (1) eller från (0). Optional.

Exempel

```
verify(1);
```

Appendix A

Exempel på konfigurering mha script

Ett exempel på ett script som läser en signallista genererad från EXCEL, och skapar kort, kanaler och signaler. OBS! Långa rader har brutits av för att kunna visas.

```
#include <ssab_exe:ssab_config_card>
!
! Read a file and create cards, channels and signals
! The file is generated from EXCEL with semicolon
delimiter ! and should have the following format
!
!       Chan    Description      Signal
Ident
!       ----    -----------      ------              ---
--
!       Di3101; Drift fläktmotor;VKV-Motor-Drift;
Blad 2
!       Di3102; Ventil öppen;    vkv-Ventil-ÖppenGL;
Blad 2
!       Di3103; Ventil stängd;   vkv-Ventil-
StängdGL;Blad 2
!       Di3104; Pump till;       vkv-Pump-DriftHK;
Blad 2
!       Di3105; Tk starta pump;  vkv-Pump-StartTK;
Blad 6
!       Di3106; Tk Stoppa pump;  vkv-Pump-StoppTK;
Blad 6
!       Di3107; Tk öppna ventil; vkv-Ventil-ÖppnaTK;
Blad 6
!       Di3108; Tk stäng ventil; vkv-Ventil-StängTK;
Blad 6
!       ...
!
main()
  int           file;
  string        str;
  string        chan;
  string        signal;
  string        desc;
  string        ident;

  file = fopen( "bok1.skv", "r");
  if ( !file)
    say( "Unable to open bok1.skv");
    exit();
```

```
  endif

  set vol/vol=volvkvend
  while ( fgets( str, file))
    chan = felement( 1, ";");
    chan = edit( chan);
    if ( strlen( chan) == 0)
      continue;
    endif
    desc = felement( 2, ";");
    desc = edit( desc);
    signal = felement( 3, ";");
    signal = edit( signal);
    ident = felement( 3, ";");
    ident = edit( ident);
    printf("—Processing %s '%s'\n", chan, signal);
    ssab_create_signal( "noder-vkvend-R1", "VKV-Reserv",
           chan, signal, desc, ident, desc);
  endwhile
  fclose( file);
  save
endmain


!
! ssab_exe:ssab_config_card.pwr_com
!
! Create a signal.
! If the card of the signal does not exist, the card
with
! channels, and signals positioned in the
reservhierarchy,
! will be created, and signals and channels will be
! connected.
! If the card already exist the signal will be move from
! the reservhierarchy to the given destination
!
function int ssab_create_signal( string rname,
    string resname, string chname, string signame,
    string chan_descr, string chan_ident, string
sig_descr)
  string        cname;
  string        cardclass;
  string        type;
  int           channels;
  int           i;
  int           j;
  string        newname;
  string        oldname;
  string        rnam;
  string        chnam;
```

```
  string        cnam;
  string        hnam;
  string        snam;
  string        chnr_str;
  string        tnam;
  int           sts;

! Check if card exist
  cname = extract( 1, 4, chname);
  chname = toupper( chname);
  type = extract( 1, 2, chname);
  chnr_str = extract( 5, 2, chname);

  set message/off
  show object/name='rname'-'cname'/out=nl:/noterm
  sts = cmd_status;
  set message/on
  if ( sts != 1)
!   Create the rack and reservhierarchy if not created
    ssab_create_object( rname, "Rack_SSAB");
    ssab_create_object( resname, "$PlantHier");

!   The card didn't exist, create it
    if ( type == "DI")
      cardclass = "Di_DIX2";
      channels = 32;
    endif
    if ( type == "DO")
      cardclass = "Do_HVDO32";
      channels = 32;
    endif
    if ( type == "AI")
      cardclass = "Ai_HVAI32";
      channels = 32;
    endif
    if ( type == "AO")
      cardclass = "Ao_HVAO4";
      channels = 4;
    endif
    if ( type == "CO")
      cardclass = "Co_PI24BO";
      channels = 1;
    endif
!   Create the card
    conf card/rack='rname'/cardname='cname'
         /cardclass='cardclass'/channelname=#
         /chanidentity='cname'#
    set attr/noco/name='rname'-
'cname'/attr=ErrorSoftLimit
         /value=15
```

```
    set attr/noco/name='rname'-
'cname'/attr=ErrorHardLimit
         /value=50
    set attr/noco/name='rname'-'cname'/attr=DevName
         /value='cname'

!   Change the name of the channels to offset 01
    for ( i = channels - 1; i >= 0; i--)
      cnam = rname + "-" + cname;
      sprintf( oldname, "%s-%02.2d", cnam, i);
      j = i + 1;
      sprintf( newname, "%02.2d", j);
      move object/source='oldname'/rename='newname'
    endfor

!   Create signalobjects in reserv hierarchy
    for ( i = 1; i < channels + 1; i++)
      sprintf( rnam, "Reserv_%s%02.2d", cname, i);
      create object/dest='resname'/class='type'
           /name="'rnam'"
    endfor

!   Connect channels and signals
    for ( i = 1; i < channels + 1; i++)
      sprintf( rnam, "Reserv_%s%02.2d", cname, i);
      sprintf( chnam, "%02.2d", i);
      set attr/noco/name='resname'-
'rnam'/attr=SigChanCon
          /value='rname'-'cname'-'chnam'
      set attr/noco/name='rname'-'cname'-'chnam'
          /attr=SigChanCon/value='resname'-'rnam'
    endfor
  endif

! Set description and identity in channel
  set attr/noco/name='rname'-'cname'-'chnr_str'
      /attr=Description/value="'chan_descr'"
  set attr/noco/name='rname'-'cname'-'chnr_str'
      /attr=Identity/value="'chan_ident'"

! Set description in signal
  sprintf( rnam, "Reserv_%s%s", cname, chnr_str);
  set attr/noco/name='resname'-'rnam'/attr=Description
      /value="'sig_descr'"

! Move signal to the final destination
  i = strrchr( signame, "-");
  if ( i)
    i++;
    j = i - 2;
    hnam = extract( 1, j, signame);
```

```
    snam = extract( i, 80, signame);
  endif
  ssab_create_object( hnam, "$PlantHier");
  move obj/source='resname'-'rnam'/dest='hnam'
       /rename="'snam'"
endfunction

!
! Create an object and it's ancestors if they doesn't
exist
!
function int  ssab_create_object( string name,
     string class)
  int    i;
  int    j;
  string parent;
  string object;
  int    sts;

  i = strrchr( name, "-");
  if ( i)
    i++;
    j = i - 2;
    parent = extract( 1, j, name);
    object = extract( i, 80, name);
  else
    object = name;
    parent = "";
  endif

  set message/off
  show object/name='name'/out=nl:/noterm
  sts = cmd_status;
  set message/on
  if ( sts != 1)
    if ( parent != "")
      ssab_create_object( parent, class);
    endif
    create object/dest="'parent'"/class='class'
        /name="'object'"
  endif
endfunction
```