rt_rtt

Användarhandledning

Revision: 99 02 18

Inledning

pwr_rtt är ett hjälpmedel för att felsöka i ett Proview/R system eller för enklare operatörskommunikation.

pwr_rtt innehåller

- ett verktyg för att navigera i databasens anläggnings och nod hierarki och visa innehållet i objekt.
- ett antal kommandon för att visa objekt, korsreferenser, hantera tid, logga, skapa/ta bort objekt mm.
- en kommandotolk för att kunna exekvera kommandon i ett script.
- ett antal systembilder för att visa status för job/processer, io-kort, näthanterare, watchdog, grafcet-sekvenser, remotetransar och NMps, samt för att logga attribut i databasen på fil.
- en editor för att bygga menyer och bilder för operatörer och underhåll.
- ett funktionstangentbord för operatörskommunikation.

pwr_rtt finns för OpenVMS på Alpha och VAX, och för VAXELN.

Det finns ett generellt rtt-program som tillhör proview's bassystem. Mha rtt-editorn kan ett projekt bygga en egen variant av detta, välja ut vissa delar av standard-programmet och bygga nya menyer och bilder.

Start av rtt

Standardversionen av rtt ligger i proview’s bassystem på pwr_exe för aktuell platform.

pwr_rtt kräver normalt att proview’s näthanterare är startad i aktuell dator .

ELN versionen (pwrb_root:[os_eln.hw_vax.exp.exe]rt_rtt.exe_eln) läggs med i EBUILD eller laddas ned för hand, och startas från ECL med

```
ECL> ex/w rt_rtt
```

VMS versionen (pwr_exe:rt_rtt.exe) startas med symbolen pwr_rtt.

```
$ pwr_rtt
```

Eventuellt kan det finnas projektspecifika varianter av rtt som startas på annat sätt.

Vid starten visas en inloggnings bild där användaren ska ange ett 'username'. Ett antal olika username finns definierade med olika behörighet att påverka databasen.

Argument

Argument 1

Username.

Inloggningsbilden kan elimineras om pwr_rtt startas med username som första argument.

Om man inte vill knyta rtt till näthanteraren, kan man lägga på prefixet ‘NONETH_’ till användarnamnet, t ex ‘NONETH_OP’

Argument 2

Rtt kommandofil.

Som andra argument kan anges en rtt kommandofil för att direkt navigera till en speciell bild eller för att skapa en dynamisk bild med intressanta databas-parameterar. Rtt behöver inte köras interaktivt vid start av med kommandofil, utan kan styras helt från sekvensen i kommandofilen.

Argument 3

Rtt konfigurations objekt (av kasset RttConfig). Om rtt ska användas som operatörsstation är de lämplig att ange ett speciellt konfigrurationsobjekt här, och specifiera automatisk laddning av larm mm.

Argument 4

Titeln i huvudmenyn.

Objekts hierarki

Objekts hierarkin visar anläggnings och nod-hierarkin. Samtliga objekt på en nivå i trädet, dvs en syskon skara, visas på en bild. Om alla syskon ej får plats på en bild delas bilden up i flera sidor. Ett objekt kan öppnas och innehållet i objektets parametrar visas.

Om ett objekt har barn markeras detta med '*'.

Man navigerar i bilden mha piltangenterna och förflyttar sig ned i hierarkin med RETURN tangenten och upp med PF4. Utvalt objekt öppnas med PF1.

Om ett objekt innehåller ett ‘Description’-attribut kan man välja att skrivas ut i bilden, tillsammans med objektsnamnet. Detta görs genom att sätta rtt’s DescriptionOn parameter från (i RttConfigObjektet, från setup-bilden eller med CTRL/L).

Följade funktioner finns i objektshierarki bilden:

Tangent	**Funktion**
PILTANGENTER	Välj ut ett objekt.
RETURN	Visa nästa nivå i hierarkin (barnen till utvalt objekt).
PF1	Visa innehållet i utvalt objekt.
PF2	Debugga barnen till utvalt objekt.
PF4 el CTRL/R	Återvänd till föregående bild.
NEXT PAGE el CTRL/F	Visa nästa sida.
PREV PAGE el CTRL/D	Visa föregående sida.
CTRL/Z	Återvänd till rotmenyn
HELP	Hjälp
DELETE	Ta bort ett meny entry.
DO el CTRL/B	Kommando mode.
CTRL/V	Lägg in utvalt objekt i 'collection picture'.
CTRL/N	Visa 'collection picture'.
CTRL/K	Kvittera senaste okvitterade larm.
CTRL/L	Toggla mellan att visa Description-attributet för ett objekt eller ej.

Visa objekt

Innehållet i ett objekt visas genom att välja ut objektet och trycka på PF1. De attribut som är relevanta visas (vissa pekare sorteras bort).

Ett attribut kan datasättas genom 'Change value' (PF3).

Attribut som är av typen vektorer markeras med en ‘*’. Dessa måste man öppna med PF1 för att se innehållet i.

Följande funktioner finns i objekts bilden.

Tangent	**Funktion**
PILTANGENTER	Välj ut en parameter.
PF1	Om innehållet i parametern är en referens till ett annat objekt visas innehållet i detta objekt.
PF3	Ändra värdet på en parameter.
PF4 el CTRL/R	Återvänd till föregående bild.
NEXT SIDE el CTRL/F	Visa nästa sida.
PREV SIDE el CTRL/D	Visa föregående sida.

CTRL/Z	Återvänd till rotmenyn.
HELP	Hjälp
DELETE	Ta bort ett meny entry.
DO el CTRL/B	Kommando mode.
CTRL/V	Lägg in utvald parameter i 'collection picture'.
CTRL/N	Visa 'collection picture'.
CTRL/K	Kvittera senaste okvitterade larm.

Debug bild

I debug bilden visas värdet på debugattributet för alla barn till utvalt objekt. Om utvalt objekt endast har ett fönsterobjekt som barn visas barnen till fönsterobjektet istället, dvs plcpgm, csub, substep, trans objekt kan debuggas direkt. Orderobjekt kan däremot ha en del andra objekt som barn och i detta fall måste man välja ut det fönster man vill debugga.

Följande funktioner finns i debug bilden.

Tangent	**Funktion**
PILTANGENTER	Välj ut ett objekt.
RETURN	Visa nästa nivå i hierarkin (barnen till utvalt objekt).
PF1	Visa innehållet i utvalt objekt.
PF2	Debugga barnen till utvalt objekt.
PF3	Ändra värdet på debug attributet.
PF4 el CTRL/R	Återvänd till föregående bild.
NEXT PAGE el CTRL/F	Visa nästa sida.
PREV PAGE el CTRL/D	Visa föregående sida.
CTRL / Z	Återvänd till rotmenyn.
HELP	Hjälp
DELETE	Ta bort ett meny entry.
DO el CTRL/B	Kommando mode.
CTRL/V	Lägg in utvalt objekt i 'collection picture'.
CTRL/N	Visa 'collection picture'.
CTRL/K	Kvittera senaste okvitterade larm.

Collection picture

Collection picture gör det möjlig att samla ihop ett antal intressanta objekt på en bild. Objekten letas upp i objektshierarkin och läggs in i 'collection picture' med CTRL/V. Om ett objekt utan specificerat attribut är utvald läggs debugattributet in i bilden. Om en parameter i en objekts bild är utvald läggs detta attribut in i bilden.

Bilden visas upp med CTRL/N (eller kommandot 'show collection').

Bilden rensas med kommandot 'collect clear'. Enstaka objekt kan tas bort ur bilden med delete tangenten.

(Objekt kan även adderas till bilden med kommandot 'add debug' eller 'add parameter'.)

Följande funktioner finns i collection bilden.

Tangent	**Funktion**
PILTANGENTER	Välj ut ett objekt.
RETURN	Visa nästa nivå i hierarkin (barnen till utvalt objekt).
PF1	Visa innehållet i utvalt objekt.
PF2	Debugga barnen till utvalt objekt.
PF3	Ändra värdet på utvalt objekt.
PF4 el CTRL/R	Återvänd till föregående bild.
NEXT PAGE el CTRL/F	Visa nästa sida.
PREV PAGE el CTRL/D	Visa föregående sida.
CTRL/Z	Återvänd till rotmenyn.
HELP	Hjälp
DELETE	Ta bort ett objekt ur bilden.
DO el CTRL/B	Kommando mode.

System bilder

Rtt innehåller ett antal bilder som visar information om systemet. Bilderna ligger under SYSTEM i menyn.

SHOW SYSTEM

I ett ELN system visas samtliga job i systemet, samt div uppgifter t ex cpu tid för varje jobb. Ytterligare info om ett job erhålls genom att öppna jobbet med PF1. Då visas processer mm.
I ett VMS system visas processer som är relaterade till proview: PAMS, näthanterare, meddelandehanterare, plcprogram mm.

PLCPGM

Bilden visar samtliga plcpgm som finns på noden och den beskrivning som ligger i plcpgm objektet. Med PF tangenterna kan man visa signaler som refereras i plcprogrammet (PF1) eller köra debug på plcprogrammet (PF2).

GRAFCET

Grafcet bilden visar de plcpgm som innehåller grafcet sekvenser. Om grafcet sekvensen i plcprogrammet är aktiv, dvs om den inte är i sitt InitStep markeras detta genom att 'Active' skrivs ut före namnet på plcprogrammet i bilden.

PF1 visar signaler som refereras i plcprogrammet. Med PF2 startar man grafcet-monitorn som visar vilka steg och vilka order som är aktiva i ett plcpgm. Vill man studera flera grafcetsekvenser samtidigt kan man markera/avmarkera plcprogram med RETURN tangenten. De markerade programmen visas av grafcet-monitorn. Maximalt fyra plcprogram kan visas samtidigt. Är inget plcprogram markerat visas det utvalda programmet.

DEVICE

Device visar samtliga kort på noden, samt felräknare och address på respektive kort.

Genom att välja ut ett kort och trycka på RETURN, visas kanalerna på kortet, samt aktuellt värde på kanalen, eventuella test, invertering och conversion flaggor. Vidare visas den signal som är knuten till kanalen. I bilden finns en ‘CngMode’ knapp med vilken man kan se ‘channel description’, ‘signal description’, eller ‘identity’ istället för kopplad signal.

Kanalbilden för Do och Ao kan sättas om i en speciell signaltest-mod genom att SignalTestModeOn-attributet sätts i setup-menyn. Denna mod gör det möjligt tvångsstyra värdet på analoga och digitala utgångar. Test-flaggan för en Do eller Ao-kanal sätts med PF2 och värdet på utgången ändras med PF1 (Do) eller PF3 (Ao). SignalTestModeOn måste sättas innan man tar upp bilden.

WATCHDOG

Watchdog visar innehållet i watchdog objekt på noden.

Watchdog objekten övervakar bla plcprogrammet och kontrollerar att det går med rätt frekvens. Se PROVEW/R Object Reference Manual fär mer information.

PID

PID visar samtliga PID-regulatorer på noden. För varje regulator visas objektsnamn (hierarkinivåer över plcprogrammet skalas bort), tvångsstyrning (ON/OFF), mode (Casc, Auto, Man), ärvärde, börvärde och utsignal.

LOGGNING

Med loggningsbilden kan man logga attribut i databasen på en fil. Loggning på maximalt tio filer kan göras samtidigt.

Här följer en lathund:

- Samla ihop de attribut som ska loggas i collection bilden.
- Gå in i loggnings bilden. Om andra loggningar redan startats, välj ett ledigt entry med tangenterna 'Nästa' eller 'Föreg'.
- Mata in tid genom att välja ut 'Time' med piltangenteran och ändra värde med PF3. Tiden är i ms.
- Mata in filnamn. Om nodnr ej anges i filnamnet kommer filen att hamna på den node som är angiven i RttConfig objektet.
- Mata in typ av loggning under 'Type'. Typen kan vara 'Cont' elller 'Event'. Cont innebär att värden loggas kontinuerligt med angiven periodtid, Event innebär att värdena loggas när de ändrar värde.
- Välj ut 'INSERT' och tryck på PF1 för att lägga in de attribut som valts ut i collection bilden.
- Välj ut 'START' och tryck på PF1 för att starta loggningen. Att loggningen är startad markeras genom att texten '*** ACTIVE ***' blinkar i bilden.
- Stoppa loggningen genom att välja ut 'STOP' och trycka på PF1.
- Titta på loggfilen genom att välja ut ‘SHOW FILE’ och trycka på PF1.

För övriga parameterar se kapitlet Loggning och kommandot logging.

NETHANDLER

Nethandler är en undermeny som innehåller bilder som visar info om näthanteraren.

SHOW NODES

Bilden 'SHOW NODES' visar de noder som akutell nod har kontakt med.

Följande parameterar visas i bilden:

Parameter	**Beskrivning**
Name	Nodnamn
Group	Pams index
Nodeid	Proview nodindex
Link	Kan vara Down, Up, Istrt eller Local Down Pamskontakt ej upprättad. Up Kontakt mellan näthanterare etablerad. Istrt Pamskontakt men ej kontakt med näthanterare. Local Nod på vilken rtt körs

Time up	Tid då näthanteraren startades
Sent	Antal sända transaktioner
Rcvd	Antal mottagna transaktioner

SHOW SUBCLIENT

Visar subscription client, dvs prenumerationer som aktuell node har begärt från andra noder. Observera här antalet okända prenumerationer, om denna är större än noll när samtliga noder i systemet är uppe betyder det att man lägger ut prenumeration på objekt som inte existerar.

SHOW SUBSERVER

Visar subscription server, dvs prenumerationer som andra noder här begärt från akuell nod.

Store

Under STORE i menyn visas lagrade collectionbilder och kommandofiler.

Gör så här:

- Samla ihop ett antal parameterar i databasen i collection picture.
- Gå in i collection picture och skriv kommandot

```
pwr_rtt> store sto1
```

eller

```
pwr_rtt> store/collect sto1
```

- Collection bilden finns nu sparad under STO1.

  Pss kan man spara bilder under valfritt namn. På VMS lagras och hämtas bilderna på den filkatalog man står på när man startar rtt. På ELN hämtas och lagras bilderna på rtt's default-filkatalog.

Loggning

Loggnings funktionen i rtt gör det möjligt att logga parameterar i rtdb på en fil.
Upp till tio loggfiler kan vara öppna samtidig och i varje loggfil kan hundra parametrar loggas.
Initiering av loggningen görs genom att ge data till en loggtabell. Varje loggfil hanteras av ett entry i loggtabellen.
Ett entry hanteras från LOGGING-bilden. Med ‘nästa sida’ och ‘föregående sida’ skiftar man entry.

Man kan även hantera ett entry mha kommandon med 'logging create' kommandot och datasätts eller ändras med 'logging set' eller 'logging delete'.

Loggningen startas med 'logging start' som skapar en process som cykliskt övervakar parameterarna.

Om en conditionparameter har angivits sker enbart loggningen om condition parametern är true.

Loggningen kan vara av två typer, antingen loggas värdet av parametrarna kontinuerligt med en viss frekvens, eller loggas en parameter varje gång dess värdet ändras.

Om loggningen sker under en längre tid är det lämligt att starta och stoppa loggningen med en rtt kommandofil,
eftersom man då inte behöver köra rtt i interaktivt (se kommandofiler samt exempel 4 i appendix C).
Rtt startas då upp med en kommandofil som parameter, varefter man kan logga ur ECL och låta rtt sköta sig själv.

Kontinuerlig loggning

Värdet på parametrarna i entryt loggas vid varje cykel. På filen skrivs dessutom tiden sedan loggningen startades. Filen kan läsas in i t ex DECchart eller Excel för grafisk presentation.
Appendix A innehåller en lathund för att använda DECchart.

Händelse loggning

En parameter i entryt loggas om värdet på parametern har ändrats.
På loggfilen skrivs tiden för ändringen och det nya värdet.

Prioritet

Prioriteten för loggningen kan påverkas i ELN, ej I VMS.

Loggningen sker default med samma prioritet som rtt, dvs 16. Om man vill undvika att tappa viktiga loggpunkter genom att loggningen inte kommer in vid rätt tidpunkt kan man öka prioriteten (lägre värde på priority). Detta kommer att medföra att loggningen sker med hög prioritet men när utskriften av data på fil startas återgår rtt till den ursprungliga prioriteten för att inte störa övriga högprioriterade program. Hur mycket som kan loggas innan utskrift på fil görs, bestämms av buffersize. En buffersize på 100 pages innebär att 100 * 512 = 51 200 tecken lagras internt. Vid utskrifen kommer rtt med stor sannoliket att missa en del loggpunkter. För att undvika detta kan man öka bufferstorleken så att hela loggningen för plats i den interna bufferten. Kontrollera först att det finns tillräckligt med dynamisk minne. Man kan även ange att loggningen automatiskt ska stoppa när bufferten är full.

Kommandofiler i rtt

Rtt innehåller ett kommandofils-språk som används bl a för att lagra collection-picture och loggnings-entryn.

Man kan modifiera dessa filer en text-editor, eller skriva egna kommandoprocedurer t ex för att bygga upp olika test-fall.

Skapa en kommandofil

En kommandofil består av kommentarsrader och kommandorader. Man kan även definiera symboler och utföra enklare arithmetik. Villkorlig exekvering och hopp är också möjlig. Se kapitlet ‘Script’ för en ingående presentation. Här visas hur man kan göra egna eller editera debug och collection-bilder.

Kommentarsraderna inleds med ett utropstecken i radens i fösta position på raden.

Kommandon för att hantera collection picture:

- collect clear
- collect /name='objektsnamn'
- show collect
- add debug /name='objektsnamn1' [/class= /hierarchy=]
- add parameter/parameter='parameter'/name='objektsnamn3' [/class= /hierarchy=]

Kommandon för att bygga en debugbild:

- debug /name='objektsnamn1' [/class= /hierarchy=]
- show parameter/name=...[/class= /hierarchy=]
- add debug /name='objektsnamn1' [/class= /hierarchy=]
- add parameter/parameter='parameter'/name='objektsnamn3' [/class= /hierarchy=]

En debugbild skapas med kommandona 'debug object' eller 'show parameter', och byggs på med kommandona 'add debug' och 'add parameter'.

Appendix C innehåller några exempel på kommandofiler.

Exekvera en kommandofil

En kommandofil exekveras genom med ett '@' tecket följt av filnamnet. I en eln-nod krävs fullständigt filnamn med nod osv.

Start av rtt med kommandofil

Rtt kan startas med en kommandofil som andra argument (första argumentet ska innehålla username). Man kan t ex skapa en symbol som direkt startar rtt i en viss bild eller med fylld collection bild.

Exempel `pwr_rtt_test == "$ssab_exe:rs_rtt sys templogg"`

Kommando filen templog.rtt_com exekveras direkt vid uppstart av rtt.

För att starta från ECL skickar man med argument på följande sätt

```
ECL> ex/w rs_rtt /arg=(sys,3.98::test)
```

Om man har en kommando fil som inte kräver något ingrepp från operatören, dvs avslutas med exit eller quit (se exempel 4 i appendix C), kan man starta rtt från ECL med execute utan /wait och sedan logga ur ECL

```
ECL> ex rs_rtt /arg=(sys,3.9::test)
```

Konfigurering av rtt

Konfigureringsobjekt

Konfigureringsobjektet kan användas för att förenkla vissa kommandon. Objektet ska vara av klassen RttConfig (under klasshierarkin SSAB) och läggs som barn till aktuell nod.

DefaultVMSNode

Nodnummer för den VMS nod som default öppnar filer på. En ELN nod innehåller normalt inte några diskar och för att öppna en fil krävs det att man anger nodnummer. Här kan man ange ett nodnummer som kommer att användas om inte något annat anges. Används av alla funktioner som innehåller en /file kvalifierare: show signals, crossreference, store, logging, alarm print, samt vid exekvering av kommandofiler.

Syntax 'areanummer'.'nodnummer'::
Ex 3.9::

UserObject

Objekt av klassen User.
Här kan ett User objekt kopplas till en rtt operatörsstation.
Se "Konfigurering av larmlista" nedan.

AlarmAutoLoad

Anger om händelselistan ska laddas vid start av rtt.
Om AlarmAutoLoad är 0 kommer händelselistan att laddas när larmlista eller händelselistan visas första gången.

AlarmMessage

Anger om senaste okvitterade larm ska visas på meddelanderaden (rad 24 i terminalfönstret).
Larmlistan
måste vara laddad för att larmen ska kunna visas.

AlarmBeep

Anger om förekomsten av okvitterade larm ska ge upphov till ljudsignal eller ej.

AlarmReturn

Anger att händelselistan ska innehålla händelsen när ett larmtillstånd återställs (endast vid AlarmAutoLoad).

AlarmAck

Anger att händelselistan ska innehålla larmkvittenser (endast vid AlarmAutoLoad).

DescriptionOff

Anger att description-attributet inte ska visas i objekts-hierarkin.

DefaultDirectory

Default filkatalog för rtt. Om ingen annan filkatalog anges vid specifikation av filer vid t ex store, show file, symbolfil, exekvering av kommandofiler mm, kommer filen att skapas resp hämtas från den defaulta filkatalogen.

SymbolFileName

Symboler för t ex kortkommandon kan laddas in vid start av rtt mha en symbolfil. Här anges namnet på den symbolfil som ska laddas.

ScanTime

Scantiden för rtt. Scantiden bestämmer bl a uppdateringsfrekvensen i bilder. Minimumvärde 1 s.

Konfigurering av larmlista

För att man ska kunna ta upp larm och händelselista i rtt krävs att det finns ett objekt med namnet RttUser av klassen User som barn till nodobjektet till den nod på vilken rtt körs.
Det är lämpligt att skapa dessa objekt på samtliga noder i systemet. I objektet ska attributen OpNumber, MaxNoOfAlarms, MaxNoOfEvents och SelectList fyllas i.

Alternativt kan man i RttConfig objektet ange ett userobjekt med annat namn.

I RttConfig objektet sätts lämpliga data på attributen AlarmAutoLoad, AlarmMessage, AlarmBeep, AlarmReturn och AlarmAck.

Tolkning av transbuffrar

I remtrans-bilden kan man se transbuffrarna för respektive remtrans i form av en c-struct. Detta kräver att den struct-namnet och den fil där structen finns definierad läggs in i remtrans-objektet (attributen StructName resp StructFile). Se kommandot ‘show object /type’ för de begränsningar som gäller vid tolkning av c-structar.

Distribution av filer till processtationer

Distribution av filer från utvecklingsmiljön till process- och operatörsstationer konfigureras i projektvolymen. Följande filer har anknytning till rtt.

- Lagrade samlingsbilder, kommandofiler och eventuell symbolfil distribueras mha ett ApplDistr-objekt. Kopiera samtliga filer genom att använda wildcard i source-fil specifikationen (*.rtt_com).
- exe-filer som inte byggs med i systemfilen distribueras med ett ApplDistribute. Om ett projektspecifikt rtt-program ska kunna startas från servicemenyn i en operatörs- eller processtation ska den döpas om till pwrp_exe:rs_rtt.exe i operatörs/process-stationen. Detta åstadkomms genom att ange rs_rtt.exe som target i ApplDistribute objektet.
- Korsreferensfiler för rtt (som skapas från Utilities med ‘create rttfiles’) distribueras tillsammans med laddatafiler till den boot-nod som angivits i NodeConfig-objektet.
- Include-filer som används för tolkning av transbufferar i ‘Show remtrans’ bilden ska distribueras med ett $ApplDistribute-objekt. Till ELN-noder ska de kopieras till disk$sys:<sys0.sysexe>.

Symbolfil

Man kan definiera symboler i rtt t ex för att använda som kortkommandon till bilder eller objekts-hierarkier. En symbol som skapas i rtt lever endast under den aktuella rtt-sessionen. Om man vill bevara symbolerna måste man lagra dem i en symbolfil. Namnet på symbolfilen anges i RttConfig-objektet. Symbolfilen läses in vid uppstart av rtt och är av rtt-script format, dvs bör ha filtyp .rtt_com.
Man kan skapa en symbolfil på två sätt:

- Definiera lämpliga symboler från rtt's kommando-prompt, kontrollera dem med kommandot 'show symbols', och lagra dem med kommandot 'store /symbols'.
- Editera symbolfilen och lägg in lämpliga kommandon. Se exempel nedan.

Symbolfilen kan även innehålla andra setup-kommandon, t ex navigera till en bild eller objekts-hierarki vid uppstart. Följande kommando i symbolfilen kommer att visa upp bilden 'underhåll-motorer'. Observera att den här typen av kommandon förloras om man använder kommandot 'store /symbols'.

```
show menu underhåll-motorer
```

Exempel på symbolfil

```
define FREKVENSOMRIKTARE "SHOW MENU UNDERHÅLL-FREKVENSOMRIKTARE"
define PLÅTFÖLJNING "SHOW MENU UNDERHÅLL-PLÅTFÖLJNING"
define HYDRAULIK "SHOW MENU \"UNDERHÅLL-HYDRAULIK PV15\""
define VKVP15 "SHOW MENU \"UNDERHÅLL-VKVP15 KOMMUNIKATION\""
define GIVARE "SHOW MENU UNDERHÅLL-GIVARE"
define SEKTIONSOBJEKT "SHOW MENU \"UNDERHÅLL-SEKTIONS OBJEKT\""
define GIVAROBJEKT "SHOW MENU \"UNDERHÅLL-GIVARE OBJEKT\""
define END "SHOW CHILD/NAME=VKV-END"
define SB15 "SHOW CHILD/NAME=VKV-END-SB15"
define S0 "SHOW CHILD/NAME=VKV-END-SB15-S0"
define S1 "SHOW CHILD/NAME=VKV-END-SB15-S1"
define S2 "SHOW CHILD/NAME=VKV-END-SB15-S2"
define S3 "SHOW CHILD/NAME=VKV-END-SB15-S3"
```

Följande exempel innehåller ett script som automatisk definierar kortkommandon till samtliga $Planthier-objekt.

```
main()
  string name;
  string segment_name;

  name = GetClassList("$PlantHier");
  while ( nname != "")
    segment_name = CutObjectName(name, 1);
    define 'segment_name' "show child/name='name'"
    nname = GetNextObject(name);
  endwhile
endmain
```

Skapa menyer

Menyer av permanent karaktär kan skapas med rtt-editorn pwr_rtt_edit. Men man kan även skapa menyer av mer dynamisk karaktär med kommandona ‘create menu’ och ‘add menu’. Med dessa kommandon kan man t ex bygga menyer som väljer ut vissa objekt ur databasen. Här visas ett exempel på ett script som väljer ut plåt-objekt i längdintervallet 10 - 15 m.

```
string name;
string attr;
float langd;
int  first

name = GetChild("VKV-Plåtar");
first = 1;
while ( name != "")
  if ( GetObjectClass(name) == "END_Plat")
    attr = nname + "Langd";
    langd = GetAttribute( attr);
    if ( langd >= 10 && langd < 15)
      if ( first)
        create menu/title="Plåtar 10-15 m"/text="'name'"/obj="'name'"
      else
        add menu/text="'name'"/object="'name'"
      endif
    endif
    first = 0;
  endif
  name = GetNextSibling(name);
endwhile
```

Rtt som operatörsterminal

Om man ska använda en fast terminal för rtt-bilder kan denna kopplas till datorn antingen via en terminal server eller till ett seriellt kort (DHV) som sätts i datorn.
Skapa dedicerade RttConfig-objekt dedicerade för terminalerna och skicka med detta som argument.

Terminal på seriell port

Programmet rs_term startar och övervakar rtt-program på VAX_ELN. Programmet läggs in i EBUILD-filen. Som argument till rs_term skickas namn på program som ska startas, port, port för felutskrifter, argument till det startade programmet.

Exempel

```
program SSABB_ROOT:<VAX_ELN.EXE>RS_TERM.EXE_ELN /argument=("""""", -
"""""", """""","""RS_RTT_HFLDS1""", """TTB0:""", -
"""CONSOLE:""","""SYS""","""""","""Noder-HFLDS1-RttConfig_Oper1""")
```

Terminal på terminal server

Här använder man programmet ssab_exe:rs_lat för att koppla upp en latport till en terminalserver samt starta och övervaka rtt-programmet. rs_lat startas av ett $Appl-objekt eller läggs in i EBUILD-filen. Som argument till rs_lat skickas med bla LAT-port, service-namn och port-namn på terminal servern, rtt-program som ska startas samt argument till detta. Porten på terminalservern ska sättas i remote läge.

Exempel

EBUILD fil för uppstart av rtt mot en terminal kopplad till en terminalserver.

```
program SSABB_ROOT:<VAX_ELN.EXE>RS_LAT.EXE_ELN /warm_debug -
/argument=("""""", """""", """""", """REVERSE""", """LTA1""", -
"""VJRT04""", """PORT_7""", """RS_RTT_VKVSU1""", """OP""", """""", -
"""Noder-VKVSU1-RttConfig_maskinskötare""")
```

Learn

Learn funktionen gör det möjlig att lagra en sekvens av tangenttryckningar och kommandon. Lagringen sker på en kommandofil med format beskrivet ovan.

Starta learn funktionen med kommandot

```
pwr_rtt> learn start/file='filnamn'
```

Filnamn måste vara en fullständig filspecifikation (måste innehålla node i eln, tex /file=3.9::test).

Fortsätt med att mata in önskad sekvens av tangentryckningar och rtt kommandon. Det är lämpligt att börja med ctrl/z för att definiera startpunkten för sekvensen. Man kan sätta värden på parameterar med 'set value' funktionen (pf3).

Avsluta learn sessionen med

```
pwr_rtt> learn stop
```

Upprepa den lagrade sekvensen med

```
pwr_rtt> @'filnamn'
```

Korsreferenser

Rtt kan visa korsreferenser, dvs de ställen i plc-koden där referenser till signaler eller andra objekt förekommer. Man kan även söka efter referenser i de objekt av typen arithm som innehåller kod. Korsreferenser visas med komandot ‘crossreference’.

Information om korsreferenser finns inte i runtime utan måste genereras från utvecklingsmiljön. Detta görs med pwr_cmd-kommandot ‘create rttfiles’. Referens-informationen läggs i fyra filer för varje volym: pwrp_load:rtt_crr_’volymsid’.dat, pwrp_load:rtt_crro_’volymsid’.dat, pwrp_load:rtt_crrc_’volymsid’.dat och pwrp_load:rtt_plc_’volymsid’.dat. Dessa distribueras tillsammans med laddatafiler till bootnoden av distributören vid ‘copy load’.

Signaler

Korsreferenser för signaler innebär att alla referenser i plc-program med GetDi, GetDo, GetDo, GetAi, GetAo, GetAv, GetPi, StoDi, StoPi, StoDo, StoDv, StoAi, StoAo, StoAv, CStoAi, CStoAv och CStoAo, listas. Om referensen är skrivande (dvs StoXx eller CStoXx) markeras detta med ‘#’ i korsreferenslistan.

Övriga objekt

Korsreferenser på objekt som inte är signaler innebär referenser i plc-program med GetData, GetDp, GetAp, StoDp, StoAp och CStoAp. Interna kopplingar mellan objekt i ett plc-program som görs med ‘connections’ tas ej med i korsreferenslistan. Om referensen är skrivande (dvs StoXx eller CStoXx) markeras detta med ‘#’ i korsreferenslistan. Om referensen är en pekar-referens (GetData) markeras detta med ‘&’.

Arithm-kod

Objekten CArithm, DataArithm, AArithm och DArithm innehåller c-kod i vilken man bl a kan göra funktionsanrop till funktioner som länkats med plc-programmet. Med crossreferens-kommandot kan man leta efter de arithm-objekt där en viss c-funktion anropas. Det sker med kommandot

```
pwr_rtt> crossref /function=
```

Resultatet blir en lista på de arithm-objekt där funktionen förekommer, samt en utskrift av de rader i koden där anropet görs. Kommandot gör en sökning i koden efter funktionsnamnet.

Sökning på en valfri sträng i arithm-koden kan också göras med /string. Skillnaden mellan /function och /string är att strängen i /funktion måste vara avgränsad med andra tecken än alfabetiska, numiska eller ‘_’ (underscore), medan /string inte ställer några krav på avgränsning.

```
pwr_rtt> crossref /string=
```

Kommandon

Allmänt

pwr_rtt går över från menymode till kommandomode med 'Utför'-tangenten eller CTRL/B och går tillbaka till menymode efter utfört kommando eller med PF4.

add

Adderar objekt eller parameterar till aktuell bild.

add parameter

Adderar objekt till aktuell bild och visar innehållet i specificerad parameter. Funktionen är densamma som 'show
parameter' med det undantaget att funna objekt adderas till nuvarande bild.

Objekt som passar in på klass, namn och hierarki beskrivningen visas på skärmen.

```
pwr_rtt> add parameter /parameter= /name= /class= /hierarchy=
```

/parameter

Namn på parametern som ska visas.

/name

Namn på ett objekt.
Wildcard är tillåtet. Namnet kan skrivas utan '/name='.

/class

Visar objekt som tillhör given klass.

/hierarchy

Visar objekt som ligger under givet hierarkiobjekt i hierarkin. Om inte något objekt anges tas hierarkinivån på aktuell bild som default ('show object /hierarchy').

add debug

Adderar objekt till aktuell bild och visar innehållet i debug parametern. Funktionen är densamma som 'debug object' med det undantaget att funna objekt adderas till nuvarande bild.

Objekt som passar in på klass, namn och hierarki beskrivningen visas på skärmen.

```
pwr_rtt> add debug /parameter= /name= /class= /hierarchy=
```

/local

Endast objekt på den egna noden adderas.

/parameter

Namn på parametern som ska visas.

/name

Namn på ett objekt. Wildcard är tillåtet.
Namnet kan skrivas utan '/name='.

/class

Visar objekt som tillhör given klass.

/hierarchy

Visar objekt som ligger under givet hierarkiobjekt i hierarkin. Om inte något objekt anges tas hierarkinivån
på aktuell bild som default ('show object /hierarchy').
"

add menu

Adderar ett meny-entry till en meny. Ett antal olika typer av menyentryn kan skapas. Default skapas ett meny-entry av typen menu, dvs som kan öppna en undermeny.

/text

Text på menyentry.

/object

Skapar ett meny-entry av typen object. Som värde till kvalifieraren ges namn på objekt som ska knytas till meny-entryt. Objektet kommer att kunna öppnas med PF1, objektets barn öppnas med RETURN och debug på barnen startas med PF2.

/command

Skapar ett meny-entry av typen command. Som värde på kvalifierarn ges ettt rtt-kommando som kommer att exekveras när RETURN aktiveras.

Exempel

Skapa ett menyentry av typen menu
```
pwr_rtt> add menu/text=”SubMenu”
```

Skapa ett menyentry av typen object
```
pwr_rtt> add menu/text=”SomeObj”/obj=Motor-Enable
```

Skapa ett menyentry av typen command
```
pwr_rtt> add menu/text=”SomeCmd”/command=@mycmd
```

alarm

Alarm visar larm och händelser, ger utskrift av händelselistan samt ger möjlighet att sända larm.

alarm list

Visar händelselistan. Händelserna visas fn i omvänd ordning (sista händelsen sist). Aktuella larm och händelser laddas första gången man gör 'alarm show' eller 'alarm list'. Kvalifierarna påverkar endast laddningen och har inte någon funktion när larmen väl är laddade.

/user

Namn på objekt av klassen User på aktuell node.
Default tas ett User objekt med namnet RttUser som ligger under aktuell nod.

/maxalarm

Maximalt antal larm som visas i larmlistan. Default 200.

/maxevent

Maximalt antal larm som visas i händelselistan. Default
500.

/acknowledge

Kvittens av larm visas i händelselistan.

/return

Retur av larmstatus visas i händelselistan.

/bell

Om det finns okvitterade larm i larmlistan ges ljudsignal från terminalen.

Exempel

```
pwr_rtt> alarm list /maxalarm=40 /maxevent=100 /acknowledge /return
```

I rtt program som är skapade med pwr_rtt_edit skapas lämpligen ett menyalternativ som visar händelselistan med kommandot

```
crea/com="alarm list/maxal=40/maxev=100/bell/ret"
/text="händelselista"
```

alarm show

Visar aktuella larm. Larmen visas fn i omvänd ordning (sista larmet sist).
Aktiva larm markeras med '**' och okvitterade med '!!'.
Okvitterade larm kvitteras med PF3.
PF1 visar larm-namnet. Om larmnamnet är ett objekt öppnas objektet.
Aktuella larm och händelser laddas första gången man gör 'alarm show' eller 'alarm list'. Kvalifierarna påverkar endast laddningen och har inte någon funktion när larmen väl är laddade.
För att kunna ladda larmlistan krävs att det finns ett objekt av klassen User på den node där rtt körs.

/user

Namn på object av klassen User på aktuell node.
Default tas ett User objekt med namnet RttUser som ligger under aktuell nod..

/maxalarm

Maximalt antal larm som visas i larmlistan. Default 200.

/maxevent

Maximalt antal larm som visas i händelselistan. Default 500.

/acknowledge

Kvittens av larm visas i händelselistan.

/return

Retur av larmstatus visas i händelselistan.

/bell

Om det finns okvitterade larm i larmlistan ges ljudsignal från terminalen.

Exempel

```
pwr_rtt> alarm show /maxalarm=40 /maxevent=100 /acknowledge /return
```

I rtt program som är skapade med pwr_rtt_edit skapas lämpligen ett menyalternativ som visar händelselistan med kommandot

```
crea/command="alarm show/maxal=40/maxev=100/bell/ret"
/text="larmlista"
```

alarm print

Skriver ut innehållet i händelselistan på fil.
Larmlistan måste först laddas med 'alarm show' eller 'alarm list'.
Observera att det kan upp till 30 sekunder att ladda larmlistan. För att vara säker på att få med hela listan bör man vänta ett tag innan man skriver ut den. Om kvittens och return av larmen ska tas med i

händelselistan anges vid laddningen av larmen. Man kan ange om man vill skriva ut enbart larmtexten eller objektsnamnet.

/file

Filnamn.

/notext

Larmtexten skrivs ej ut.

/noname

Objektsnamnet skrivs ej ut.

Exempel

```
pwr_rtt> alarm print /file=alarmlist.txt/noname
pwr_rtt> alarm print /file=3.9::alarmlist.txt
```

alarm send

Skickar ett larm till larmlistan med texten 'LARM FRÅN RTT 'larmtext''.

```
pwr_rtt> alarm send /text='larmtext' /priority=
```

/text

Larm text som ska skrivas ut.

/prioritet

Larmets prioritet. Kan vara A, B, C eller D (A är default).

Exempel

```
pwr_rtt> alarm send /text="VKVAU1 bootar om 10 min!!" /priority=B
```

classhier

Visar klass hierarkien.

collect

collect

Adderar utvalt objekt till collection bilden, eller objekt med specificerat namn.

```
pwr_rtt> collect
pwr_rtt> collect /name=
```

/name

Namn på parameter som ska adderas till collection picture.

Exempel

```
pwr_rtt> collect /name=sh-fsdn-varvtal.actualvalue
```

collect clear

Rensar collection bilden.
Kan endast utföras i collection bilden.

collect show

Visar objekt som samlats ihop med collect funktionen (identiskt med CTRL/N).

create

create object

Skapar ett objekt i rtdb.
Rtt måste köras i den nod som objektet ska tillhöra.
Om man i rtt tittar på den objektshierarki där objektet ska skapas, kommer denna ej att uppdateras förrän man har gått ur den (med PF4).

/class

Klass på det objekt som ska skapas.

/name

Fullständigt namn på det objekt som ska skapas.
Om gemener ska ingå i objektsnamnet måste detta omges av 'dubbelfnuttar'.

Exempel

```
pwr_rtt> create object /class=Material /name="N2-UGN-PLÅTAR-Mtrl1
```

crossreference

Crossreference visar korsreferenser till en signal (do, di, do, ai, ao, av eller pi) eller andra objekt som refereras i plc-programmet med GetDp, StoDp, GetAp, StoAp, GetData o dyl. Även funktioner, classer och attribut som används i arithm-objekt kan man få korreferenser på.
Informationen om korsreferenser finns ej i runtime databasen, därför hämtas korsreferenserna från den listfil som genereras på kommandot 'create rttfiles' i pwr_cmd och Utilities. För att kunna fungera krävs att det finns en aktuella listfiler.
Om en annan listfil ska användas, eller en listfil på en annan node, kan detta anges med kvalifieraren /file.

Om /name utelämnas visas korsreferenser för utvald signal.

```
pwr_rtt> crossreference
```

/file

Filnamn på listfilen.

/name

Signal objekt eller annat objekt för vilket korsreferenser ska visas.
Wildcard är tillåtet.
Default det objekt som är utvalt.

/function

Namn på den funktion, class, attribut eller dylikt som används i koden för objekt av typen AArithm, DArithm, CArithm och DataArithm. Namnet på arithm-objektet och den eller de kod-rader där funktionen förekommer visas (med /brief visas endast den första kodraden för varje objekt). Skillnaden mellan /funktion och /string är att /string tillåter alla tecken som avgränsningstecken, medan /funktion inte tillåter siffror eller bokstäver.

```
pwr_rtt> crossreference /function= [/brief] [/case_sensitive]
```

/brief

Används vid sökning i koden för arithm-objekt. Med /brief visas endast kod-raden för den första förekomsten av funktionen.

/case_sensitive

Används vid sökning i koden för arithm-objekt. Anger att sökningen ska vara känslig gemener och versaler. Om /case_sensitive används på en sträng med små bokstäver måste funktions-strängen omgärdas av ‘dubbelfnuttar’.

Om /case_sensitive utelämnas är sökningen känslig för gemener och versaler om söksträngen innehåller gemener och är omgärdad av ‘dubbelfnuttar’, annars är sökningen ej känslig. /case_sensitive behöver med andra ord endast användas vid sökning på en sträng med enbart versaler.

Observera att rtt’s kommandotolk översätter alla strängar till versaler som inte är omgärdade av ‘dubbelfnuttar’.

Exempel

```
pwr_rtt> cross /function=MyFunc
```
Sökningen ej känslig för gemener och versaler.

```
pwr_rtt> cross /function=”MyFunc”
```
Sökningen känslig för gemener och versaler.

```
pwr_rtt> cross /function=MyFunc /case_sensitive
```
Sökningen känslig för gemener och versaler.
Söksträngen kommer att bli ‘MYFUNC’.

/string

Sökning efter en sträng i koden för arithm-objekt.

```
pwr_rtt> crossreference /string= [/brief] [/case_sensitive]
```

debug

debug object

Visar värdet på debug parametern för ett eller flera objekt.
Objekt som passar in på klass, namn och hierarki beskrivningen visas på skärmen.

```
pwr_rtt> debug object /parameter= /name= /class= /hierarchy=
```

/global

Även objekt i monterade volymer på andra noder visas.

/name

Namn på ett objekt. Wildcard är tillåtet.
Namnet kan skrivas utan '/name='.

/class

Visar objekt som tillhör given klass.

/hierarchy

Visar objekt som ligger under givet hierarkiobjekt i hierarkin. Om inte något objekt anges tas hierarkinivån på aktuell bild som default ('show object /hierarchy').

debug children

Visar värdet på debug parametern för barnen till ett objekt (även PF2). Om objektet endast har ett fönsterobjekt som barn visas barnen till fönsterobjektet.

```
pwr_rtt> debug children /name=
```

/name

Namn på ett objekt
Namnet kan skrivas utan '/name='.

define

Define kommandot använd för att tilldela värden till symboler.

Första argumentet är namnet på symbolen som ska tilldelas värdet. Namnet ska här skrivas utan enkelapostrofer.
Andra argumentet kan vara ett värde (flyttal, heltal eller sträng).

System symboler

Det finns att antal symboler som är definierade av systemet

rtt_os	innehåller aktuellt operativsystem (VMS eller ELN)
rtt_platform	innehåller aktuell plattform (VAX_VMS, VAX_ELN ellerAXP_VMS)
rtt_time	innehåller aktuell tid (i ascii format).

Tilldelning av ett värde

Tilldelat värde kan vara en string, float, integer eller boolean.

Exempel
Tilldelning av flyttal

```
define num 3.22
define num 3.22e-12
```

Tilldelning av heltal

```
define i 0
```

Tilldelning av sträng

```
define some_str "This is a string"
```

Tilldelning av innehållet i en annan symbol

```
define a1 'a2'
```

Addera två strängsymboler

```
define a1 "MOTOR"
define a2 " START"
define a3 ‘a’’a2’

sho sym a3
%RTT-I-MSG, Symbol a3 = MOTOR START
```

Hämta värden på attribute från rtdb

Man kan hämat upp aktuella värden på attribut från databasen genom att framför attributnamnet skriva ett '#' tecken och omge # och namn med enkelapostrofer.

Exempel

```
define a '#rt-rtt-vax_vms-dv1.actualvalue'
sho sym a
%RTT-I-MSG, Symbol a = 0
```

a tilldelas värdet av rt-rtt-vax_vms-dv1

```
if '#rt-rtt-vax_vms-dv1.actualvalue' eq 0
  say/text="Dv1 är 0"
endif
```

delete

delete object

Tar bort ett objekt ur i rtdb. Endast dynamiska objekt kan tas bort.

Rtt måste köras i den nod som objektet ska tillhöra. Om man i rtt tittar på den objektshierarki där objektet
ligger, kommer denna ej att uppdateras förrän man har gått ur den (med PF4).

/name

Fullständigt namn på det objekt som ska tas bort.

Exempel

```
pwr_rtt> delete object /name=N2-UGN-PLÅTAR-Mtrl1
```

exit

Avslutar exekveringen.

get clock

Sätt systemtiden till tiden på hårdvaruklockan (endast på VAXELN-system med installerad hårdvaruklocka).

```
pwr_rtt> get clock
```

help

Visar hjälptexter.

Om kommandot ges utan parameter visas en lista på de hjälptexter som finns. Ett ämne kan väljas ut med piltangenterna, och med RETURN visas hjälptexten för utvalt ämne.

```
pwr_rtt> help
```

Om ämne eller kommando skickas med som parameter visas hjälptexten för ämnet.

```
pwr_rtt> help ‘ämne’
```

learn

Learn gör det möjlig att lagra en sekvens av knapptryckningar och rtt-kommandon som ofta upprepas. Lagringen sker på en kommandofil av med namnet *.rtt_com. Sekvensen återupprepas genom att starta den skapade kommandofilen.

```
pwr_rtt> learn start /file=
pwr_rtt> learn stop
```

Exempel

```
pwr_rtt> learn start /file=3.9::motorstart
...
diverse kommandon och knapptryckningar
...
pwr_rtt> learn stop


pwr_rtt> @3.9::motorstart
```

learn start

Startar en learn session.
Efterföljande kommandon kommer att lagras på angiven fil.

/file

Namn på kommandofil på vilken kommandona lagras

Exempel

```
pwr_rtt> learn start /file=dosering
```

learn stop

Avslutar en learn session.

logging

Loggings funktionen i rtt gör det möjligt att logga parameterar i rtdb på en fil. Loggningen kan vara av två typer, antingen loggas värdet av parametern kontinuerligt med en viss frekvens, eller loggas parametern varje gång värdet ändras.

```
pwr_rtt> logging create /insert/time= /parameter= /file=/condition=
       /type=/buffer_size=
pwr_rtt> logging set /insert/entry= /time= /parameter= /file=
/condition=       /type=/buffer_size=
pwr_rtt> logging delete /entry= /parameter=
pwr_rtt> logging start /entry=
pwr_rtt> logging stop /entry=
pwr_rtt> logging show /entry=
```

Exempel
Loggning av parametrarna uppsamlade i collection picture.

```
pwr_rtt>logging create /insert/file=3.9::test.dat
pwr_rtt>logging start /entry=1
pwr_rtt>logging stop /entry=1
```

logging create

Skapar ett entry i loggtabellen. Entry innehåller information om loggning på en loggfil. Maximalt 10 entryn kan finnas samtidigt. Vid 'logging create' kommandot erhålls ett entry nr som anges vid senare kommandon som påverkar entryt.

/insert

Lägger in parametrarna i collection picture i entryt.

/time

Cykeltid för loggningen i ms.

/file

Fullständig filspecifikation (inklusive VMS-node).

/parameter

Namn på parameter som ska loggas ('objektsnamn.parameternamn').

/condition

Namn på parameter av typen boolean om villkorlig loggning ska ske. Loggning sker endast då condition parametern är TRUE.

/type

Typ av loggning

- EVENT: varje gång värdet på en parameter i entryt ändras skrivs tid och värde på loggfilen.
- CONTINOUS: (default) värdet av samtliga parametrar i entryt skrivs på loggfilen varje cykel om condition parametern är TRUE.

/buffer_size

Storlek på den buffer som loggingen lagras på i primärminnet i pages. När bufferten full töms den på fil.
Öka buffer_size vid logging av snabba, tidskritiska förlopp.

/priority

Prioritet som loggningen sker på. När bufferten töms återgår rtt till ursprunglig prioritet.

/intern

När bufferten är full stoppas loggningen automatiskt.

logging set

Sätter eller ändrar värden i ett entry i loggtabellen.

/insert

Lägger in parametrarna i collection picture i entryt.

/entry

Entry nr.

/time

Cykeltid för loggningen i ms.

/file

Fullständig filspecifikation (inklusive VMS-node).

/parameter

Namn på parameter som ska loggas ('objekts namn.parameternamn'). Maximalt 10 parameterar per entry kan loggas.

/condition

Namn på parameter av typen boolean om villkorlig loggning ska ske. Loggning sker endast då condition parametern är TRUE.

/type

Typ av loggning.

- EVENT: varje gång värdet på en parameter i entryt ändras skrivs tid och värde på loggfilen.
- CONTINOUS: (default) värdet av samtliga parametrar i entryt skrivs på loggfilen varje cykel om condition parametern är TRUE.

/buffer_size

Storlek på den buffer som loggingen lagras på i primärminnet i pages. När bufferten full töms den på fil.
Öka buffer_size vid logging av snabba, tidskritiska förlopp.

/priority

Prioritet som loggningen sker på. När bufferten töms återgår rtt till ursprunglig prioritet.

/intern

När bufferten är full stoppas loggningen automatiskt.

logging delete

Tar bort en entry i loggtabellen eller en parameter i ett entry. Om /parameter utelämnas tas hela entry bort.

/entry

Entry nr.

/parameter

Parameter som ska tas bort ur entryt.

logging start

Startar loggningen för ett entry. Öppnar en fil och skapar en process som scannar över parametrarna i entryt med vald tidbas.

/entry

Entry nr.

logging stop

Stoppar loggningen för ett entry. Stänger filen och stoppar processen.

/entry

Entry nr.

logging store

Lagra ett logg-entry eller samtliga logg-entryn på rtt-kommandofilsformat.
Lagrade entryn återskapas genom att exekvera kommandofilen.

```
pwr_rtt> logging store /entry=1/file=temperatur_logg
```

/entry

Entry som ska lagras. Default det entry som visas i loggnings-bilden.

/file

Namn på rtt kommandofilen som kommer att innehålla beskrivningen av logg-entryt.

/all

Samtliga logg-entryn kommer att lagras i filen.

logging show

Visar innehållet i ett entry loggtabellen. Om /entry utelämnas visas samtliga entryn.

/entry

Entry nr.

page

Visar specificerad sida.

```
pwr_rtt> page 'sidnummer'
```

plcscan

Plcscan gör det möjligt att stänga av exekveringen av ett eller flera plcfönster.
OBS!! Plcscan bör endast användas vid test och simuleringstillfällen.

Genom att stänga av exekveringen kan man göra modultester på enskilda plcpgm eller plc fönster på följande sätt.

Ladda systemet i rtvaxen utan io-hantering genom att sätta parametrarna IOSimulFlag och IOReadWriteFlag i $Node objektet. Stäng av exekveringen på samtliga plcprogram med plcscan/off/all. Sätt på exekveringen på enstaka plcprogram eller plc fönster med plcscan och använd simulate funktionen i Trace för att sätta och återställa signaler.

/off

Stänger av exekveringen på utvalt plc fönster.

/on

Sätter på exekveringen på utvalt plc fönster.

/hierarchy

Sätter exekvering av eller på på samtliga plcfönster som ligger under utvalt objekt i hierarkin.

/all

Sätter exekvering av eller på på samtliga plcfönster.

/global

Vid /hierarchy och /all sker en sökning på plcfönster. Med /global sker sökningen i hela systemet. Default sker
sökningen enbart på aktuell nod.

print

Skriver ut innehållet i en bild på fil.
Bilden kan vara en collection picture, objektsbild eller debug bild.

/file

Filnamn. Default filtyp är .lis.

Exempel

```
pwr_rtt> print /file=rtt_picture
```

say

Skriver ut en text på sys$output

/text

Text som skrivs ut.

Exempel

```
say /text="Now I'm writing a text on sys$output"
```

search

Letar efter en sträng i den nuvarande bilden.
Sökriktningen är framåt och startar från utvalt objekt. Om inte någon söksträng ges som parameter tas senast inmatade söksträng.

```
pwr_rtt> search 'söksträng'
```

set

set conversion

Sätter conversion av insignaler på eller av.
Kräver systemprivilegier.

```
pwr_rtt> set conversion /name= /on
pwr_rtt> set conversion /name= /off
```

Kanalen som ska modifieras kan anges med /name kvalifieraren eller genom att välja ut kanalobjektet eller tillhörande signalobjekt.

/name

Namn på kanalobjektet eller på tillhörande signalobjekt.

/on

Sätter conversion på.

/off

Sätter conversion av.

set clock

Sätter tiden på hårdvaruklockan till aktuell systemtid (endast på VAXELN system med installerad hårdvaruklocka).

set draw

Uppdatera skärmen vid exekvering av ett script.
Normalt uppdateras inte skärmen vid exekvering av ett script utan bilden ritas om när skriptet avslutas.
‘set draw’ gör att bilden succesivt uppdateras och att den avslutande omritningen ej utförs.
‘set nodraw’ gör att uppdateringen upphör.

```
pwr_rtt> set draw
pwr_rtt> set nodraw
```

set file

Samtliga print kommandon, där fil ej specificeras, kommer att skrivas ut filen som anges i set file/on kommandot. Även meddelanden och kommandon kan skrivas i filen.

/on

Start av set file session.

/off

Stopp av set file session.

/name

Namn på fil.

/message

Meddelanden från rtt kommer att skrivas i filen.

/command

Kommandon som utförs kommer att skrivas i filen.

Exempel

```
set file/on/name=pwrp_lis:set_file_example.txt/message/command
print/text="Start of set file example..."
...
set file/off
quit
```

set invert

Sätter invertering av digitala signaler på eller av.
Kräver systemprivilegier.

```
pwr_rtt> set invert /name= /on
pwr_rtt> set invert /name= /off
```

Kanalen som ska modifieras kan anges med /name kvalifieraren eller genom att välja ut kanalobjektet eller tillhörande signalobjekt.

/name

Namn på kanalobjektet eller på tillhörande signalobjekt.

/on

Sätter invertering på.

/off

Sätter invertering av.

set parameter

Sätter värde på en parameter i databasen.
Kräver system privilegier.

```
pwr_rtt> set parameter /name= /value=
```

/name

Fullständigt namn på parameterm ('objektsnamn'.'parameternamn').

/value

Värde som parametern ska tilldelas.

/bypass

Gör det möjligt att använda 'set parameter' trots att man inte har systemprivilegier.

Exempel

```
pwr_rtt> set par /name=sh-fsdn-motorstart.actualvalue /value=1
pwr_rtt> set par/name=sh-fsdn-varvtal.actualvalue /value=3.33
```

set priority

Ändrar prioriteten på rtt jobbet. Endast ELN.

```
pwr_rtt> set priority /priority=
```

/priority

ELN prioritet i området 1 till 31 (1 är högsta prioritet).

Exempel

```
pwr_rtt> set priority /priority=4
```

set mode

Sätt rtt i olika mode.

set mode address

Kommando för att debugga plcprogrammet.

Man kan även patcha (peka om vissa kopplingar) i plcprogrammet. Funktionen är ganska krånglig fn och kräver insikter i hur plcprogrammet fungerar. Kommandot medför att addresser för pekare till ingångar på beräkningsobjekt kommer att visas i 'show object'. Adresserna i rtdb för parameterarna kommer också att visas.

Adresserna är specifika för varje job, och beroende på var i jobbet som rtdb mappas. Addresserna som ligger i ingångarna har lagts in av plcprogrammet och kan ej direkt jämföras med de addresser som skrivs ut för
parameterarna i objektet, eftersom detta är addresserna i rtt jobbet. För att få dessa att överensstämma kan man ange en rtdb offset som förskjuter rtt addresserna. Titta på addressen på en ingång på ett beräkningsobjekt som ej är kopplad. InxP ska här peka på Inx. Ta skillnaden mellan innehållet i InxP och addressen förInx, mata
in den med kommandot

```
pwr_rtt> set rtdb_offset/offset='skillnaden'
```

Gå ur och in i 'show object' bilden och kontrollera att addressen i InxP och addressen för Inx är samma. Man kan nu jämföra de addreser som ligger i pekarna med de addresser som skrivs ut för parameterarna.

set mode noaddress

Lämnar address mode.

set rtdb_offset

Detta fungarar ej i version X1-0T och senare...

Ger möjlighet att mata in skillnad i rtdb's basaddress mellan plcjobbet och rttjobbet, för att kunna debugga pekare i plcobjekt. Se 'set mode address' hur man kan lista ut skillnaden.

/rtdb_offset

Skillnad i rtdb's basaddress mellan plcjobbet och rttjobbet.

set time

Sätter systemtiden. Tiden anges på formatet 'dd-mmm-yyyy:hh:mm:ss'.

```
pwr_rtt> set time dd-mmm-yyyy:hh:mm:ss
```

Exempel

```
pwr_rtt> set time 30-APR-1997:14:07:00
```

set verify

Alla kommandon i en kommandofil och alla meddelanden skrivs ut på skärmen.

set noverify

Avslutar 'set verify'

setup

Parameterar för uppsättning av rtt.

Parameter	**Beskrivning**
Platform	Visar aktuell platform, ej ändringsbar.
ConfigureObject	Visar aktuellt RttConfig-objekt, ej ändringsbar.
DefaultVMSNode	VMS-nod som ett ELN-system utan lokal disk ska hämta filer ifrån.
DescriptionOn	Description-attributet visas i objektshierarkin.
DefaulDirectory	Default filkatalog för lagrade bilder, kommandofiler mm
Scantime	Skantid, bestämmer bl a uppdateringsfrekvensen i bilder. Minvärde 1000 ms. Tiden anges i ms.
AlarmMessage	Senaste okvitterade larm visas på rtt's meddelanderad (näst nedersta raden).
AlarmBeep	Ljudsignal om det finns okvitterade larm.
AlarmReturn	Retur av larm visas i händelselistan (måste anges innan händelselistan laddas).
AlarmAck	Kvittens av larm visas i händelselistan (måste anges innan händelselistan laddas).
SymbolFilename	Namn på symbolfil som exekveras vis start av rtt.
Verify	Kommandofiler körs med verify, dvs utskrift av exekverade kommandon.
SignalTestModeOn	Sätter 'SHOW DEVICE' för Do och Ao-kort i signal-test mod. Det innebär att utsignaler kan tvångsstyras från bilden.

show

Visar information om objekt i databasen.

show conversion

Visar conversion av en kanal.

```
pwr_rtt> show conversion /name=
```

Kanalen som ska visas kan anges med /name kvalifieraren eller genom att välja ut kanalobjektet eller tillhörande signalobjekt.

/name

Namn på kanalobjektet eller på tillhörande signalobjekt.

show file

Visar filer.
Som parameter kan en filspecifikation anges. Om filspecifikationen utelämnas visas rtt-kommandofiler på default-directoryt.
De filer som matchar filspecifikationen visas upp i en bild där man kan välja ut filerna med piltangenterna. Om filen är av typen .rtt_com (rtt-kommandofil) kan denna exekveras genom att man väljer ut den och trycker på RETURN. Om filen är av typen .txt visas innehållet i filen mha view.

```
pwr_rtt> show file
pwr_rtt> show file pwrp_exe:*.rtt_com
```

parameter

Filspecifikation. Kan innehålla wildcard.
Om ett utropstecken läggs omedelbart före filspecifikationen visas enbart filnamnet, utan disk och filtyp.
Exempel `pwr_rtt> show file !*.rtt_com`

show invert

Visar invertering av en kanal.

```
pwr_rtt> show invert /name=
```

Kanalen som ska visas kan anges med /name kvalifieraren eller genom att välja ut kanalobjektet eller tillhörande signalobjekt.

/name

Namn på kanalobjektet eller på tillhörande signalobjekt.

show object

- Visar objektet för ett givet objdid.

```
pwr_rtt> show object /objdid=
```

- Visar ett objekt med givet namn, eller alla objekt som passar in på en objektsbeskrivning. Objektsbeskrivningen görs med kvalifierarna /class, /hierarchy och /name, vilket gör att objekt som tillhör en viss klass, ligger under ett viss hierarkinivå eller har ett namn som matchar en wildcard beskrivning.

```
pwr_rtt> show object /name= /class= /hierarchy=
```

- Visar en transbuffer tolkad som en c-struct.

```
pwr_rtt> show object /type= /file= /name=
```

/global

Även objekt i monterade volymer på andra noder visas.

/name

Namn på ett objekt. Wildcard är tillåtet.
Namnet kan skrivas utan '/name='.

/class

Visar objekt som tillhör given klass.

/hierarchy

Visar objekt som ligger under givet hierarkiobjekt i hierarkin. Om inte något objekt anges tas hierarkinivån på aktuell bild som default ('show object /hierarchy').

/objdid

Visar objektet med specificerat objdid.

Exempel

```
pwr_rtt> sho obj /class=di
```

Visar alla objekt i systemet av klassen di.

```
pwr_rtt> sho obj /class=ai  *temp*
```

Visar alla objekt av klassen ai där strängen "temp" ingår i namnet.

```
pwr_rtt> sho obj/hier /class=step
```

Visar alla step-objekt på aktuell hierarkinivå.

```
pwr_rtt> sho obj temp_pid
```

Visar objektsbild på objektet 'aktuell hierarinivå'-temp_pid

```
pwr_rtt> sho obj/class=select/ hier=ugn-zon1 *gas
```

Visar alla objekt av klassen select under hierarkinivån ugn-zon1 vars namn slutar på "gas".

/type

Visar ett transbuffer-objekt i form av en c-struct. Namn på den c-struct som ska användas.
Rtt öppnar den include-fil (som anges med /file) och letar efter structen. För att rtt ska kunna tolka c-structen krävs följande begränsningar i syntaxen:

1) structen ska vara definierad med en typedef i den includefil som anges.
2) elementen i structen ska vara av följande typer:
   - vanliga c-typer: (unsigned/signed) int, long, short, char, float, double eller struct.
   - proview-typer (pwr_tInt32, osv).
   - typedef'ade structar. Structarna kan finnas definierade i andra includefiler. Dessa filer måste vara includerade i den include-fil som anges i /file=.
3) arrayer får ha högst två dimensioner.
4) en array av struct får ha en dimension.
5) tomma rader får ej finnas mellan elementen i structen.
6) kommentarsrader måste inledas med /* på varje rad.
7) storlek på arrayer måste anges explicit (ej mha #define)
8) logiska namn på filkataloger för filer includerade med #include måste vara kända i realtidsmiljön om dessa innehåller typdefinitioner som refereras i den ursprungliga structen. För eln-noder förutsätts att include-filerna ligger på disk$sys:<sys0.sysexe>

/file

Filnamn på den includefile som innehåller definitionen c-structen som anges i /type.

show parameter

Visar innehållet i en parameter för ett eller flera objekt. Objekt som passar in på klass, namn och hierarki beskrivningen visas på skärmen.

```
pwr_rtt> show parameter /parameter= /name= /class= /hierarchy=
```

/local

Endast objekt på den egna noden visas.

/parameter

Namn på parametern som ska visas.

/name

Namn på ett objekt. Wildcard är tillåtet.
Namnet kan skrivas utan '/name='.
Om /parameter utelämnas kan namnet även innehålla parameter namnet.

/class

Visar objekt som tillhör given klass.

/hierarchy

Visar objekt som ligger under givet hierarkiobjekt i hierarkin. Om inte något objekt anges tas hierarkinivån på aktuell bild som default ('show object /hierarchy').

Exempel

```
pwr_rtt> sho par /par=valueindex /class=ao
```

Visar parametern valueindex för alla ao-objekt i systemet.

```
pwr_rtt> sho par /name=sh-fsdn-auto.actualvalue
```

show class

Visar klasser. Klasser som passar in på namn och hierarkibeskrivningen visas.

```
pwr_rtt> show class /name= /hierarchy=
```

/name

Namn på classen. Wildcard är tillåtet.
Namnet kan skrivas utan '/name='.

/hierarchy

Visar klasser i en klasshierarki (t ex MpsClasses).

Exempel

```
pwr_rtt> sho class *di*
```

Visar alla klasser under BaseClasses där "di" ingår i namnet.

```
pwr_rtt> sho class /hier=MpsClasses
```

Visar alla klasser under MpsClasses.

show clock

Visar tiden på hårdvaruklockan om sådan finns installerad (endast på VAXELN).

show hierarchy

Visar rot-objekten i objekts hierarkin.

show signals

Show signals visar signaler (do, di, do, ai, ao, av och pi) som refereras i ett plcpgm eller ett plc fönster. Informationen om vilka signaler som refereras finns ej i runtime databasen, därför hämtas signalerna från den
listfil som genereras på kommandot 'list plcpgm' i pwr_plc. För att kunna fungera krävs att det finns en aktuell listfil med namnet pwrp_lis:pwrp_plcmodulelist.plis. Om en annan listfil ska användas, eller en listfil på en annan node, kan detta anges med kvalifieraren /file. I ELN-versionen måste denna kvalifieraren användaseftersom nodenummer ska ingå i filnamnet.

Om man i rtt befinner sig i ett plc fönster visas refererade signaler i aktuellt fönster om /name utelämnas. Om man har valt ut ett plcpgm visas refererade signaler i samtliga fönster i plcprogrammet.

/file

Filnamn på listfilen.
Default pwrp_lis:pwrp_plcmodulelist.plis.

/name

Window objekt under vilket refererade signaler ska visas.
Wildcard är tillåtet.
Default det window objekt som är utvalt.

Exempel

```
pwr_rtt> show signals
pwr_rtt> show signals /file = 3.9::pwrp_lis:pwrp_plcmodulelist.plis
pwr_rtt> show signals /name=sh-fsdn-plc-sekvens*
```

show symbol

Visar värdet av en symbol

Exempel

```
show sym rtt_os
%RTT-I-MSG, Symbol rtt_os = ELN
```

show time

Visar systemtiden.

show version

Visar rtt version.

store

Store lagrar en collection bild eller en debug bild på en fil med rtt kommando format.
Bilden kan återskapas genom att exekvera kommandofilen. Om qualifieraren /collect adderas kommer bilden att återskapas i collection bilden, annars återskapas den som en debug bild.

/file

Filnamn. /file= kan utelämnas.

/collect

Bilden återskapas i collection bilden.

Exempel
Hämta upp lämpliga parametrar i collection bilden och ta upp bilden. Ge kommandot

```
pwr_rtt> store testbild
```

Bilden kan nu återskapas med

```
pwr_rtt> @testbild
```

Bilden kan kopplas till en meny genom att men i pwr_rtt_edit skapar en meny item av typ command som exekverar kommandofilen (rtt_edit> create /command=@testbild testbild). På så sätt kan man skapa en dynamisk bild som kan förändras efter olika testbehov.

/symbols

Lagrar definierade symboler i en symbolfil. Om /file utelämnas lagras symbolerna i den symbolfil som finns angiven i RttConfig-objektet.

top

Gå till början på bilden, dvs visa första sidan och välj ut första objektet.

```
pwr_rtt> top
```

wait

Wait är avsett att användas i rtt kommandofiler där man vill utföra kommandon med en viss fördröjning

```
pwr_rtt> wait /time=
```

/time

Tid i sekunder (endast heltal).

Exempel

```
pwr_rtt> wait /time=20
```

/plcpgm

Väntar tills plcprogrammet är laddat och initierat.
Används för att ändra värden på parameterar i rtdb vid systemstart mha rtt kommandofil, när dessa parameterar initieras av plcprogrammet.

Exempel

```
!
! Kommandofil som startas vid systemstart
! och initierar parameter som tidigare blivit initierade
! av plcprogrammet
!
wait /plcpgm
set
parameter/parameter=sh-fsdn-börv_varvtal.actualvalue/value=
exit
!
! slut
!
```

Se även appendix C exempel 4.

Script

Rtt-script är ett sätt att programmera rtt-kommandon. Scripthanteraren ger dessutom möjlighet till att göra beräkningar, utföra vilkors-satser, loop-satser, deklarerara varibler och funktioner. Mha rtt-script kan man bl a navigera i rtt, bygga menyer, leta efter objekt och hämta och tilldela attribut i realtidsdatabasen.

Ett script startas med '@' följt av scriptfils namnet och eventuella argument. Man kan även starta ett script från en menu-entry av typen command, eller från en tryckknapp av typen command.

Exempel

```
pwr_rtt> @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.
- Rtt-kommandon, skickas vidare till rtt´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 '.rtt_com', och default filkatalog är samma som övriga rtt-filer och anges i RttConfig-objektet.

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 rtt-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.
- Likamed-tecken följt av initialvärde, om detta ej anges sätts värdet till 0 eller null-string.
- 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";
extern  int     jakob;
global  float   ferdinand = 1234;
```

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 ett uttryck som en variabel 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”

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.

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
-=	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.

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!");
```

Rtt-kommandon

Samtliga rtt-kommandon är tillgängliga i script-koden. En rtt-kommandorad ska *inte* avslutas med semikolon.
Variabler kommer att substitueras i kommandoraden om de omgärdas av enkel-apostrofer.

Exempel

```
  string name;
  string parname;
  int j;
  int i;

  for ( i = 0; i < 3; i++)
    parname = "vkv-test-obj" + (i+1);
    create obj/name='parname'
    for ( j = 0; j < 3; j++)
      name = parname + "-obj" + (j+1);
      create obj/name='name'
    endfor
  endfor
```

Simulering av tangenttryckningar

Det finns ett antal kommandon för att simulera tangent-tryckning i ett script. Dessa måste placeras i första position på kommandoraden.

Kommandona motsvarar de funktioner knapptryckningar på motsvarande tangent har i aktuell bild. Noquiet är ett hjälpmedel för felsökning och innebär att alla operationer är synliga på skärmen.

Efter kommandot PF3 kan ett värde anges och aktuellparameter datasätt med värdet.

I normala fall bör man undvika navigerings kommandon eftersom kommandofilen måste modiferas om objekt tas bort eller om nya skapas. Använd i stället kommandot ‘show menu’ för att visa en viss bild, och ‘show children’ för att visa en viss syskonskara.

- K_RETURN
- K_PF1
- K_PF2
- K_PF3
- K_PF4
- K_ARROW_UP
- K_ARROW_DOWN
- K_ARROW_RIGHT
- K_ARROW_LEFT
- K_PREVIOUS_PAGE
- K_NEXT_PAGE
- K_CTRLV
- K_DELETE
- K_CTRLN
- K_CTRLZ
- K_CTRLW
- NOQUIET

Symboler

I rtt’s kommandotolk man man skapa symboler som främst används för kortkommandon. Symbolerna lagras i form av strängar. Symboler definieras och tilldelas värden med define kommandot.

I samtliga fall när man använder en symbol ska namnet omgärdas med enkelapostrofer, utom i kommandona define och show symbol, och om symbolen används som ett kortkommando. Om en symbol innehåller space måste man dessutom i de flesta fall omgärda den med dubbelapostrofer.

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
MessageError	Skriver ut ett felmeddelande på rtt's meddelande-rad
MessageInfo	Skriver ut ett informationsmeddelande på rtt's meddelande-rad
printf	Formaterad utskrift
say	Skriver ut en sträng
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
GetClassList	Hämta första objektet av en klass
GetNextObject	Hämta nästa objekt av en klass
GetNextSibling	Hämta nästa syskon till ett objekt
GetNodeObject	Hämta $Node-objektet
GetParent	Hämta förälder till ett objekt
GetObjectClass	Hämta klassen till ett objekt
GetRootList	Hämta första objekt i rot-listan

Bildhanterings funktioner

Funktion	Beskrivning

GetCurrentObject	Hämta namn på objekt förknippat med utvalt menyentry.
GetCurrentText	Hämta text på utvalt menyentry eller fält
GetCurrentTitle	Hämta titel på aktuell bild eller meny
GetInput	Hämta inmatning med bibehållen uppdatering av fält och larm.
LineErase	Radera en rad från cursor positionen
PlaceCursor	Hämta nästa objekt av en klass

System funktioner

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

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");
```

GetClassList()

string GetClassList(string class)

Beskrivning

Hämta det första objektet av en specificerad klass. Nästföljande objekt kan hämtas med GetNextObject().
Returnerar namnet på objektet. Om det inte finns några instanser av klassen, returneras en null-sträng.

Argument

string name namn på klassen.

Exempel

```
string name;

name = GetClassList("Dv");
```

GetCurrentObject()

string GetCurrentObject( )

Beskrivning

Returnerar name på objekt som är förknippat med aktuellt menyentry.
Om det inte finns något förknippat objekt returneras en Null-stäng.

Exempel

```
string name;

name = GetCurrentObject();
```

GetCurrentText()

string GetCurrentText( )

Beskrivning

Returnerar texten på aktuellt menyentry eller fält.

Exempel

```
string text;

text = GetCurrentText();
```

GetCurrentTitle()

string GetCurrentTitle( )

Beskrivning

Returnerar titel på aktuell bild eller meny.

Exempel

```
string title;

title = GetCurrentTitle();
```

GetInput()

int GetInput(string format, (arbitrary type) input_variable [,string prompt] [,int terminator])

Beskrivning

Formaterad inmatning av en variabel av typen float, int eller string. GetInput uppdaterar bilder och larmtexter under inmatning-sekvensen (till skillnad från scanf och ask). Genom att undersöka terminator-tecknet kan man detektera tangentryckning av PF-tangenter mm.
Returnerar antal inmatade element.

Argument

string	format	format, c-syntax.
arbitrary type	input_variable	variabel som inmatat värde läggs i. Kan vara int, float eller string.
string	prompt	optional. Specificerar den prompt som ska visas.
int	terminator	optional. Returnerar ascii-koden för terminerings-tecknet. För tangenter med utan ascii-kod, dvs PF-, pil-, funktions-tangeter mf, finns den returnerade koden definierad i filen pwr_inc:rt_rtt.h (RTT_K_…). T ex PF1-PF4 har koden 278-281.

Exempel

```
int value;
string text;

PlaceCursor( 1, 23);
LineErase();
PlaceCursor( 1, 23);
GetInput( "%d", value, "Select coolingbed: ");
PlaceCursor( 1, 23);
LineErase();

sprintf( text, "You have selected coolingbed %d...", value);
MessageInfo( text);
```

GetNextObject()

string GetNextObject(string name)

Beskrivning

Hämta nästa objekt i klasslistan.
Returnerar namnet på objektet. Om det inte finns något nästa objekt returneras en null-sträng.

Argument

string	name	name på objektet.

Exempel

```
string name;

name = GetClassList("Di");
while ( name != )
  printf("Di object found: %s", name);
  name = GetNextObject(name);
endwhile
```

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 nname;
int not_first;

name = GetChild("Rt");
not_first = 0;
while ( name != "")
  if ( !not_first)
    create menu/title="The Rt objects"/text="'name'"/object="'name'"
  else
    add menu/text="'name'"/object="'name'"
  endif
  not_first = 1;
  nname = GetNextSibling(nname);
endwhile
if ( !not_first )
  MessageError("No objects found");
```

GetNodeObject()

string GetNodeObject()

Beskrivning

Hämta nod-objeketet.
Returnerar namnet på nodobjektet.

Exempel

```
string node;
node = GetNodeObject();
```

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");
```

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");
```

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
```

LineErase()

int LineErase( )

Beskrivning

Radera rad på bildskärment fr o m aktuell cursor-position.

Exempel

```
PlaceCursor( 1, 23);
LineErase();
```

MessageError()

string MessageError(string message)

Beskrivning

Skriv ett felmeddelande på skärmen.

Exempel

```
MessageError("Something went wrong");
```

MessageInfo()

string MessageInfo(string message)

Beskrivning

Skriv ett informations-meddelande på skärmen.

Exempel

```
MessageInfo("Everything is all right so far");
```

PlaceCursor()

int PlaceCursor(int column, int row)

Beskrivning

Flytta cursorn till specificerad rad och kolumn.
Bildskärmens övre vänstra hörn har positionen (1, 1).

Argument

int	column	kolumn (1-80).
Int	row	rad (1-24).

Exempel

```
PlaceCursor( 1, 23);
```

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 filrader 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…)

Beskrivning

Formaterad utskrift till en sträng-variabel. C-syntax. Sträng argumant, format argument samt ytterligare argument av godtyckligt antal och av godtycklig typ.
.
Returnerar antal utskrivna tecken.

Argument

string	str	Sträng som ska skrivas i.
string	format	Uttskifts format.
arbitrary type	arg1	Värde argument. Optional. Godtyckligt antal. 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 = element( pos, 10, str);
```

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”);
```

time()

string time()

Beskrivning

Returnerar aktuell tid i strängformat.

Exempel

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

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);
```

Kända fel

Här presenteras en lista på kända fel i rtt.

Start av rtt från ECL med kommandofil utan /wait

Vid start av loggning kan man starta rtt från ECL med en kommandofil som argument utan /wait. Rtt blir då hängande till nästa till nästa ECL-kommando. Dvs logga omedelbart ut (Log är ett kommando som släpper loss rtt) eller skriv t ex kommando ECL> !

Ex

```
ECL> ex rs_rtt/arg=(sys, 3.98::test)
ECL> !
```

Pil vänster i rtt-bild på position rad 1 kolumn 2

Om cursorn i en rtt-bild befinner sig på rad 1 kolumn 2, om man mha pil vänster vill flytta pilen till rad 1 kolumn 1, flyttas cursorn istället till fältet längst med till höger.

Larmhantering ej tidssorterad

Larmhanteringen är ej tidssorterad utan ordnas i den följd de inkommer till rtt. Om man har larmhantering från flera noder i system fungerar den ej tillfredställande, eftersom de lagrade larmen i de olika noderna vid uppstart av rtt kommer skurvis till rtt.

Appendix A

Användning av DECchart

Med DECchart kan man presentera data i grafisk form. Data som loggas med loggningsfunktionen (typ kontinuerlig logging) i pwr_rtt är anpassade för att kunna läsas in i DECchart.

Här följer en beskrivning av hur man hur man läser in data från rtt i DECchart.

- Kopiera filerna ssab_inc:chart$defaults.dat och ssab_inc:rs_rtt.chart_style till sys$login.
- Starta DECchart med kommandot
  @vue$library:chart$vue
- Ta upp ett worksheet genom att välja 'Worksheet...' under 'File' i menyn.
- Läs in loggfilen genom att välja 'Import...' under 'File' i worksheet menyn. Välj formatet ASCII_TABULAR och välj ut eller mata in loggfilen i Import-fönstret. Observera att filnamnet måste innehålla hakparenteser ([]) om sådana förekommer i filnamnet. Byt ut < och > mot [och]!!! Tryck på OK.
- Loggningsdata ska nu dyka upp i worksheet bilden. Välj ut samtliga data genom att klicka på fältet markerat '*'.
- Genom att klicka på 'Draw Chart' ritas ett diagram ut i DECchart fönstret.
- Skriv ut det på lämplig skrivare med 'Print'.

Appendix B

Exempel på loggning

Exempel på loggning med pwr_rtt (type=EVENT).

```
RTT LOGGING STARTED AT 23-JUL-1992 10:58:40.14
Parameter: VHX-IL-KRAN-DI_ORDERIN.ActualValue
Parameter: VHX-IL-KRAN-DI_SVARUT.ActualValue
Parameter: VHX-IL-KRAN-DO_ORDERUT.ActualValue
Parameter: VHX-IL-KRAN-DO_SVARIN.ActualValue
Parameter: VHX-IL-KRAN-SENAST_ORDER.ActualValue
23-JUL-1992 10:58:40.15 VHX-IL-KRAN-DI_ORDERIN.ActualValue   0
23-JUL-1992 10:58:40.15 VHX-IL-KRAN-DI_SVARUT.ActualValue    0
23-JUL-1992 10:58:40.15 VHX-IL-KRAN-DO_ORDERUT.ActualValue   0
23-JUL-1992 10:58:41.54 VHX-IL-KRAN-DO_ORDERUT.ActualValue   1
23-JUL-1992 10:58:41.54 VHX-IL-KRAN-DO_SVARIN.ActualValue    0
23-JUL-1992 10:58:41.94 VHX-IL-KRAN-DI_SVARUT.ActualValue    1
23-JUL-1992 10:58:44.04 VHX-IL-KRAN-DI_SVARUT.ActualValue    1
23-JUL-1992 10:58:44.34 VHX-IL-KRAN-DI_ORDERIN.ActualValue   1
23-JUL-1992 10:58:45.44 VHX-IL-KRAN-DO_ORDERUT.ActualValue   0
23-JUL-1992 10:58:45.44 VHX-IL-KRAN-DO_SVARIN.ActualValue    1
23-JUL-1992 10:58:45.94 VHX-IL-KRAN-DI_ORDERIN.ActualValue   0
23-JUL-1992 10:58:45.94 VHX-IL-KRAN-DI_SVARUT.ActualValue    0
23-JUL-1992 10:58:54.94 VHX-IL-KRAN-DO_ORDERUT.ActualValue   1
23-JUL-1992 10:58:54.94 VHX-IL-KRAN-DO_SVARIN.ActualValue    0
23-JUL-1992 10:58:55.24 VHX-IL-KRAN-DI_SVARUT.ActualValue    1
23-JUL-1992 10:58:55.74 VHX-IL-KRAN-DI_ORDERIN.ActualValue   1
23-JUL-1992 10:58:56.74 VHX-IL-KRAN-DO_ORDERUT.ActualValue   0
23-JUL-1992 10:58:56.74 VHX-IL-KRAN-DO_SVARIN.ActualValue    1
23-JUL-1992 10:58:57.34 VHX-IL-KRAN-DI_ORDERIN.ActualValue   0
23-JUL-1992 10:58:57.84 VHX-IL-KRAN-DI_SVARUT.ActualValue    1
```

Appendix C

Exempel på kommandofiler

Exempel 1

```
!
! Denna kommandofilen läser in tre parameterar i collection picture
! och sätter värdet 33.3 på den tredje
!
ctrlz
collect show
collect clear
collect/name=sh-fsdn-plc-fdv-auto
collect/name=sh-fsdn-plc-fdv-manu
collect/name=sh-fsdn-reserv1904
show collect
arrow_down
arrow_down
pf3 33.3
```

Exempel 2

```
!
! Denna kommandofil visar alla trigflaggor i transportobjekt
! under hierarkin sh-fsdn-mps
! samt en dv
!
show
parameter/parameter=triggflag/class=mpstrp/hier=sh-fsdn-mps
add debug/name=ds-fsdn-mps-reset
```

Exempel 3

```
!
! Läs in ett antal parameterar i collection picture
! och starta loggning på dessa
!
ctrlz
collect show
collect clear
collect/name=sh-fsdn-plc-ai33
collect/name=sh-fsdn-plc-fdv-manual
collect/name=sh-fsdn-reset_motor
collect show
!
! Starta loggning
```

```
logging create/insert/file=3.9::log.dat/type=event
logging start/entry=1
!
! slut
!
```

Exempel 4

```
!
!temp_loggning.rtt_com
!
! Denna kommandofil startar logging av två avobjekt och stoppar
! loggningen efter 60 minuter
! Rtt jobbet startas med kommandot
!
!ECL> ex rs_rtt/arg=(sys, 3.9::temp_loggning)
!
! Efter att ha startat jobbet kan man logga ur ECL och låta rtt sköta
! sig själv.
!
logg create /file=3.9::temperatur.log /time=60000
logg set /entry=1 /parameter=sh-fsdn-plc-temp1.actualvalue
logg set /entry=1 /parameter=sh-fsdn-plc-temp2.actualvalue
logg start /entry=1
!
! Vänta i 60 min och stoppa loggingen
!
wait /time=3600
logg stop /entry=1
!
! stoppa rtt
!
exit
!
! slut
```

Exempel 5

```
!
! Skapa ett objekt med löpnummer som namn, fyll det med data
! och lägg in objektet i en NMpsCell
!
! Hämta upp nästa löpnummer
!
if defined lopnum
  define lopnum ++
else
  define lopnum '#SIM-END-PF-LöpnrRäknare.ActualValue'
  define lopnum ++
endif
!
set par/name=SIM-END-PF-LöpnrRäknare.ActualValue/value='lopnum'
!
! Skapa objeket
!
create object/class=vkvend_plat/name=vkv-end-pf-plåtar-'lopnum'
!
! Datasätt objektet
!
```

```
set par/name=vkv-end-pf-plåtar-'lopnum'.Lopnr/val='lopnum'
set par/name=vkv-end-pf-plåtar-'lopnum'.Langd/val=12
set par/name=vkv-end-pf-plåtar-'lopnum'.Bredd/val=2.2
set par/name=vkv-end-pf-plåtar-'lopnum'.Tjocklek/val=5
define offsetr30 '#VKV-END-R31-OffsetR30.ActualValue' + 2
set par/name=vkv-end-pf-plåtar-'lopnum'.Koord_R30/val='offsetr30'
set par/name=vkv-end-pf-plåtar-'lopnum'.Koord_xf_Svb/val=13
set par/name=vkv-end-pf-plåtar-'lopnum'.Koord_xB_Svb/val=1
set par/name=vkv-end-pf-plåtar-'lopnum'.Koord_f_R31/val=13
set par/name=vkv-end-pf-plåtar-'lopnum'.Koord_b_R31/val=1
set par/name=vkv-end-pf-plåtar-'lopnum'.HogNr/val=289
!
! Lägg in i en NMpsCell
!
set par/name=VKV-END-PF-PlåtFöljning-W-FrånTRP.ExternObjId
           /val=vkv-end-pf-plåtar-'lopnum'
set par/name=VKV-END-PF-PlåtFöljning-W-FrånTRP.ExternOpType/val=0
set par/name=VKV-END-PF-PlåtFöljning-W-FrånTRP.ExternFlag/val=1
```