

# RT\_XTT

## Användarhandledning

|           |          |               |
|-----------|----------|---------------|
| Revision: | 02 08 13 | Claes Sjöfors |
| Version:  | V4.0-1   |               |

|  |    |
|--|----|
| Inledning .....                            | 6  |
| Operatörs-fönster.....                     | 7  |
| Okvitterade larm.....                      | 7  |
| Knappar för att öppna bas-funktioner.....  | 7  |
| Funktions-knappar.....                     | 7  |
| Processbilder.....                         | 8  |
| Öppna en processbild .....                 | 8  |
| Stänga en processbild .....                | 8  |
| Ändra i databasen.....                     | 8  |
| Layout.....                                | 8  |
| Distribution .....                         | 8  |
| Larmlista .....                            | 9  |
| Öppna larmlistan .....                     | 9  |
| Kvittering av larm.....                    | 9  |
| Larm raden.....                            | 9  |
| Meny.....                                  | 9  |
| Konfigurering .....                        | 9  |
| Händelselista .....                        | 10 |
| Öppna händelselistan.....                  | 10 |
| Händelse raden.....                        | 10 |
| Menu.....                                  | 10 |
| Runtime navigatören .....                  | 11 |
| Navigera med tangentbordet.....            | 11 |
| Navigera med musen .....                   | 11 |
| Trace .....                                | 12 |
| Öppna trace.....                           | 12 |
| Trace-fönstret .....                       | 12 |
| View.....                                  | 12 |
| Trace .....                                | 12 |
| Simulate .....                             | 12 |
| Menu.....                                  | 12 |
| Systembilder.....                          | 14 |
| System/Nethandler/Link.....                | 14 |
| System/Nethandler/Subscription Client..... | 14 |
| System/Nethandler/Subscription server..... | 14 |
| System/Communication/RemNode.....          | 14 |
| System/Communication/RemTrans.....         | 14 |
| System/Device.....                         | 15 |
| System/PlcThread.....                      | 15 |
| System/PlcPgm.....                         | 15 |
| System/Logging.....                        | 15 |
| Klassbilder.....                           | 16 |
| Användarklasser.....                       | 16 |
| Hierarkibilder.....                        | 16 |
| Trendkurvor.....                           | 17 |
| Kurvfenstret .....                         | 17 |
| Logging .....                              | 18 |
| Starta loggningen.....                     | 18 |
| Kontinuerlig logging .....                 | 18 |
| Händelsestyrd logging.....                 | 18 |
| Buffer.....                                | 18 |
| Lagra .....                                | 18 |
| Kommandon.....                             | 18 |
| Villkorlig logging .....                   | 18 |
| Attribut.....                              | 18 |
| Javabilder.....                            | 20 |

|  |    |
|--|----|
| Öppna en javabild .....                    | 20 |
| Stänga en javabild .....                   | 20 |
| Konfigurering .....                        | 20 |
| Behörighetskontroll .....                  | 21 |
| Runtimenavigatören .....                   | 21 |
| Trace .....                                | 21 |
| Klassbilder .....                          | 21 |
| Ge och java-bilder .....                   | 21 |
| Funktionstangenter .....                   | 22 |
| Funktionsanrop .....                       | 22 |
| SetDig() .....                             | 22 |
| ResetDig() .....                           | 22 |
| ToggleDig() .....                          | 22 |
| Command() .....                            | 22 |
| Resursfil .....                            | 22 |
| Hjälp .....                                | 23 |
| Få hjälp .....                             | 23 |
| Operatörsfönstret .....                    | 23 |
| Meny .....                                 | 23 |
| Kommando .....                             | 23 |
| Tryckknappar i bilder .....                | 23 |
| Navigera i hjälptexten .....               | 23 |
| Hjälpfiler .....                           | 23 |
| Syntax .....                               | 23 |
| Exempel .....                              | 25 |
| Generera hjälpfil från wb_load-filer ..... | 26 |
| Generera html-filer från hjälpfil .....    | 26 |
| Popup meny .....                           | 27 |
| Konfigurering .....                        | 27 |
| Konfigurering .....                        | 29 |
| Operatörsmod .....                         | 29 |
| Underhållsmod .....                        | 29 |
| Symbolfil .....                            | 29 |
| Kommandon .....                            | 30 |
| Allmänt .....                              | 30 |
| add .....                                  | 31 |
| add parameter .....                        | 31 |
| add menu .....                             | 31 |
| collect .....                              | 32 |
| collect .....                              | 32 |
| collect show .....                         | 32 |
| collect clear .....                        | 32 |
| close .....                                | 33 |
| close alarmlist .....                      | 33 |
| close eventlist .....                      | 33 |
| close graph .....                          | 33 |
| close navigator .....                      | 33 |
| create .....                               | 34 |
| create item .....                          | 34 |
| create object .....                        | 34 |
| crossreference .....                       | 36 |
| delete .....                               | 38 |
| delete object .....                        | 38 |
| exit .....                                 | 39 |
| define .....                               | 40 |
| delete .....                               | 41 |
| delete item .....                          | 41 |
| help .....                                 | 42 |

|                                       |    |
|---------------------------------------|----|
| login .....                           | 43 |
| logout.....                           | 44 |
| logging.....                          | 45 |
| logging create.....                   | 45 |
| logging set.....                      | 45 |
| logging start .....                   | 46 |
| logging stop .....                    | 46 |
| logging store .....                   | 46 |
| logging show.....                     | 47 |
| open .....                            | 48 |
| open graph.....                       | 48 |
| open jgraph.....                      | 48 |
| open trace .....                      | 48 |
| open trend.....                       | 49 |
| open operatorwindow.....              | 49 |
| open url .....                        | 49 |
| search.....                           | 50 |
| set .....                             | 51 |
| set advanceduser.....                 | 51 |
| set parameter.....                    | 51 |
| setup .....                           | 52 |
| show.....                             | 53 |
| show file .....                       | 53 |
| show object .....                     | 53 |
| show parameter.....                   | 54 |
| show version.....                     | 54 |
| show symbol.....                      | 54 |
| show plcpgm.....                      | 55 |
| show plcthread.....                   | 55 |
| show links .....                      | 55 |
| show logfiles .....                   | 55 |
| show alarmlist.....                   | 55 |
| show eventlist.....                   | 55 |
| show time .....                       | 55 |
| show user.....                        | 55 |
| store .....                           | 56 |
| Script.....                           | 57 |
| Programsatser.....                    | 57 |
| Include.....                          | 57 |
| Datatyper .....                       | 57 |
| Deklaration av variabler.....         | 58 |
| Datatypkonvertering.....              | 58 |
| Konstanter.....                       | 58 |
| Uttryck.....                          | 59 |
| Funktioner.....                       | 60 |
| main.....                             | 60 |
| Argument .....                        | 61 |
| Villkors-satser.....                  | 61 |
| Loop-satser.....                      | 61 |
| Hoppsatser .....                      | 62 |
| Rtt-kommandon.....                    | 62 |
| Simulering av tangenttryckningar..... | 63 |
| Symboler.....                         | 63 |
| Sammanfattning .....                  | 63 |
| Inbyggda funktioner.....              | 65 |
| In och utmatning .....                | 65 |
| Filhantering.....                     | 65 |
| Hantering av strängar.....            | 65 |

|                                 |     |
|---------------------------------|-----|
| Databas funktioner .....        | 65  |
| Bildhanterings funktioner ..... | 66  |
| System funktioner .....         | 66  |
| CutObjectName() .....           | 67  |
| GetAttribute() .....            | 68  |
| GetChild() .....                | 69  |
| GetClassList() .....            | 70  |
| GetCurrentObject() .....        | 71  |
| GetCurrentText() .....          | 72  |
| GetCurrentTitle() .....         | 73  |
| GetInput() .....                | 74  |
| GetNextObject() .....           | 75  |
| GetNextSibling() .....          | 76  |
| GetNodeObject() .....           | 77  |
| GetObjectClass() .....          | 78  |
| GetParent() .....               | 79  |
| GetRootList() .....             | 80  |
| LineErase() .....               | 81  |
| MessageError() .....            | 82  |
| MessageInfo() .....             | 83  |
| PlaceCursor() .....             | 84  |
| edit() .....                    | 85  |
| element() .....                 | 86  |
| exit() .....                    | 87  |
| extract() .....                 | 88  |
| fclose() .....                  | 89  |
| felement() .....                | 90  |
| fgets() .....                   | 91  |
| fopen() .....                   | 92  |
| fprintf() .....                 | 93  |
| printf() .....                  | 94  |
| scanf() .....                   | 95  |
| sprintf() .....                 | 96  |
| strchr() .....                  | 97  |
| strlen() .....                  | 98  |
| strchr() .....                  | 99  |
| strstr() .....                  | 100 |
| say() .....                     | 101 |
| system() .....                  | 102 |
| time() .....                    | 103 |
| toupper() .....                 | 104 |
| Appendix A .....                | 105 |
| Exempel på loggning .....       | 105 |

# Inledning

rt\_xtt är ett verktyg för operatörs-kommunikation, underhåll och felsökning. Xtt innehåller bl a följande komponenter:

- Operatörs-fönster. Ett fönster med de senaste larmen och snabbknappar för t ex bilder.
- Processgrafik. Processgrafik ritas i Ge-editorn och öppnas i xtt från operatörsfönstret eller från kommando-raden.
- Larmlista och händelselista.
- Navigator. Med runtime-navigatören kan man titta i databasen och navigera i menyer och hjälp-filer.
- Trace. Felsökning i plc-programmet.
- Kommando-tolk. I xtt finns en kommandorad från vilken man kan exekvera en mängd kommandon.
- Script-hanterare. Man kan skriva script i ett c-liknande script-språk.
- System-bilder. Det finns ett antal bilder som visar information om näthanterare, plc-trådar mm.
- Objektsbilder. För vissa klasser finns objektsbilder (t ex PID, Av). För användar-klasser kan objekts-bilder ritas i Ge-editorn.
- Trend-kurvor. Visning av innehållet i trend-objekt.
- Logging. Attribut i databasen kan loggas på fil och visas i kurvform.
- Java-bilder. Man kan göra bilder i java (i t ex JBuilder) och öppna dessa från xtt.
- Behörighetskontroll.
- Funktionstangentbord. Funktions-tangenter kan programmeras för att öppna bilder och exekvera kommandon.
- Hjälp. Xtt kan visa dokumentation om xtt-funktioner och proview-klasser. Det går även att skriva hjälp-texter om projektet, process-bilder, plc-program och larm.

Xtt kan startas i operatörs-mod eller i underhålls-mod. I operatörs-mod visas ett operatörsfönster med bl a larmtexter och snabb-knappar. I underhålls-mod visas runtime-navigatören.

# Operatörs-fönster

Operatörsfönstret är det fönster som operatören använder för att öppna process-bilder, larm-fönster mm.

## ***Okvitterade larm***

Till vänster i operatörsfönstret visas senaste okvitterade larm. I övre delen visas A-larm med röd-makering. Genom att ändra på fönster-storleken kan man se mellan 2 och 5 A-larm. Här finns även en knapp för att kvittera det senaste A-larmet och en siffra som visar antalet rådande larm. I den undre delen visas B- C- D-larm samt info-meddelanden.

B-larm är gul-markerade, C-larm blå, D-larm violett och info-meddelanden gröna. Larmen visas i prioritets-ordning snarare än tids-ordning, dvs om det finns B-larm visas det senaste okvitterade B-larmet. Endast om det inte finns några okvitterade B-larm visas eventuella C-larm, osv. Det finns även en knapp för att kvittera det visade larmet.

## ***Knappar för att öppna bas-funktioner***

I mitten av operatörsfönstret finns ett antal knappar för att öppna vissa bas-funktioner i xtt. En del knappar syns ej vid minimal fönsterstorlek.

- Alarmlist. Öppnar larm-fönstret som visar rådande och/eller okvitterade larm.
- Eventlist. Öppnar händelselistan.
- Navigator. Öppnar runtime-navigatorn.
- Help. Visar hjälp-texter för projektet.

## ***Funktions-knappar***

Till höger finns det plats för upp till 15 knappar. Dessa konfigureras med XttGraph-objekt (se nedan) och kan användas för att öppna process-bilder eller sätta signaler i databasen.

# Processbilder

Processbilder visar tillståndet i en anläggning, dvs hämtar värden från realtids-databasen och presenterar dessa i en bild som staplar, kurvor eller siffer-värden, eller genom att ändra färg och form på objekt i bilden. Operatören ges även möjlighet att påverka processen genom att klicka på tryckknappar och mata in värden.

## Öppna en processbild

En process bild kan öppnas på följande sätt:

- med en funktions-knapp i operatörsfönstret.
- med en funktions-tangent på tangentbordet.
- med xtt-kommandot *'open graph'*.
- med xtt-kommandot *'show graph'* som listar alla .pwg-filer. Genom att dubbel-klicka på en fil i listan kan man öppna bilden.
- med en tryckknapp i en bild av typen *'Command'* som innehåller kommandot *'open graph'*.
- med ett meny-entry i xtt av typen *'Command'* som innehåller kommandot *'open graph'*.

## Stänga en processbild

En processbild stängs genom att man klicka med MB3 någonstans i bilden, eller med *'Close'* i fönstrets meny (Alt+F4). Man kan även stänga en bild med *'close graph'* kommandot.

## Ändra i databasen

En processbild är uppbyggd av objekt i form av fyrkanter, cirklar, ventiler, ramar osv. På objekten kan man sätt olika typer av dynamik så att de ändrar färg eller form, beroende på värden i databasen. Från vissa objekt kan man ändra värden i databasen genom att klicka eller dra på objekten. De objekt som man kan klicka eller dra på, *'hot'*-markeras när markören är placerade på dem. Det innebär att markören ändras till ett hårkors, samt att grafiken hos objektet förändras något. Vanligtvis ritas det med något tjockare linjebredd. Innan ändringen i databasen verkställs, kontrolleras att användaren har privilegier att ändra på det här objektet.

## Layout

När en bild öppnas mha kommandot *'open graph'* kan man påverka hur mycket av bilden som visas, och hur bilden påverkas när man ändrar storlek på den.

Normalt anger man vid editering av bilden hur stor del av bildarean som ska visas, genom att ange övre vänstra och under högra hörnet för bilden. Har man gjort detta anpassas bilden skalning så att den alltid visar detta område. Dessutom låses förhållandet mellan fönstrets höjd och bredd så att det motsvarar utsnittet.

Om man inte anger ett bild-utsnitt kommer inte bilden att skalas om vid ändring av fönsterstorlek. Man kan då öppna bilden med

- scroll-lister.
- ett navigations-fönster med vilket man kan scrolla och zooma.
- en meny med zoom-funktioner.

Se kommandot *'open graph'* och kvalifierarna */scrollbar*, */navigator* och */menu*.

## Distribution

Bilderna ritas i Ge-editor. Denna genererar en .pwg som ska kopieras till målnodens pwrp\_exe. Eventuella externa submodeller måste också kopieras.



# Larmlista

Larmlistan visar rådande och okvitterade larm.

## Öppna larmlistan

Larmlistan kan öppnas

- Från knappen 'Alarmlist' i operatörsfönstrets mittsektion.
- Från meny-entryt 'Alarm/Alarm list' i runtimenavigatörens standardmeny.
- Med kommandot 'show alarmlist'.

## Kvittering av larm

Från larmbilden kan senaste okvitterade larm kvitteras från menyn 'Functions-Acknowledge' eller *Ctrl+K*. Kvittering kan även ske från operatörsfönstrets kvittens-knappar, och med kommandot 'alarm acknowledge'. Kommandot kan t ex läggas ut på funktions-tangenter.

## Larm raden

På en rad i larmfönstret visas ett rådande och/eller okvitterat larm.

I början på raden finns en färgmarkering och en bokstav som markerar larmets prioritet. A-larm markeras med rött, B-larm med gult. C-larm och D-larm markeras enbart med bokstav och Info-meddelanden med grönt utan bokstav. Därefter följer två symboler, en klocka som markerar att larmet är okvitterat, samt ett utropsteckat i en varnings-triangel som markerar att larmet är rådande. Efter detta följer tiden, larmtexten samt larm-namnet. Larm-namnet är vanligtvis objektet som ger upphov till larmet. Man kan välja i menyn om man vill dölja larmtexten eller larm-namnet.

## Meny

| File/Close                            | Stänger larmfönstret.   |
|---------------------------------------|---|
| Functions/Acknowledge                 | Kvitterar senaste okvitterade larm.   |
| Functions/Open Plc                    | Öppnar trace för larm-objektet. Detta kräver att larm-objektet är ett plc-objekt.   |
| Functions/Display objekt in navigator | Visar larm-objektet i runtime-navigatorn.   |
| View/Zoom in                          | Zoomar in.  |
| View/Zoom out                         | Zoomar ut.  |
| View/Zoom reset                       | Återställer till ursprunglig zoom-faktor.   |
| View/Display hundredth                | Visar hundradels sekund i larm-tiden.   |
| View/Hide event name                  | Döljer larm-namnet.   |
| View/Hide event text                  | Döljer larm-texten.   |
| Help/Alarmlist                        | Visar hjälptext om larmlist-fönstret.   |
| Help/Help on selected event           | Visar hjälptext för utvalt larm. Detta kräver att det Dsup eller Asup-objektet för larmet finns som ett ämne (topic) i projektets xtt-hjälpfil. |

## Konfigurering

I user-objektet konfigurerar man en del saker som hör till larm-listan: MaxNoOfAlarms och SelectList. Vilket user-objekt som xtt läser beror på i vilken mod xtt startas. I operatörs-mod kopplas OpPlace-objektet till ett User-objekt via OpNumber-attributet. I underhålls-mod används det User-objekt som finns angivet i RttConfig-objektet.

# Händelselista

Händelselistan visar larm och händelser. Händelser kan vara när ett larm blir rådande, när det kvitteras och när det upphör att vara rådande. Via Dsup och Asup-objekt kan man även skicka händelser som enbart visas i händelselistan och inte i larmlistan.

## **Öppna händelselistan**

Händelselistan kan öppnas

- Från knappen *'Eventlist'* i operatörsfönstrets mittsektion.
- Från meny-entryt *'Alarm/Event list'* i runtimenavigatörens standardmeny.
- Med kommandot *'show eventlist'*.

## **Händelse raden**

Händelseraden är ganska lik larm-raden i larmlistan. Markering av prioritet, tid, text och namn är densamma. Skillnaden ligger i att det finns tre olika typer av händelser, när ett larm går till, när det kvitteras och när larmtillståndet försvinner. Detta markeras med olika symboler, ett utropstecken när larmet går till, en bock för kvittens och ett korsat utropstecken när larmtillståndet upphör.

## **Menu**

Funktionen hos menyn är densamma som för larmlistan.

# Runtime navigatören

Med runtime navigatören navigerar man i en trädstruktur bestående av mappar och löv. Man kan navigera med musen eller med tangentbordet. Ur ergonomisk synvinkel är tangentbordet att föredra.

## ***Navigera med tangentbordet***

Här använder man framför allt piltangenterna för att navigera.

Med 'Upp' och 'Ner' väljer man ut en map eller ett löv. Mappen öppnas med 'Pil-höger' och stängs med 'Pil-vänster'. För att stänga en mapp kan man antingen välja ut den öppna mappen, eller välja ut ett stängt barn, under mappen.

Ett objekt i realtidsdatabasen öppnar man med 'Shift Pil-höger', dvs man trycker först på Shift-tangenten och håller den nedtryckt medan man trycker på 'Pil-höger'. Om man har satt 'advanced user' i setup'en räcker det med att trycka på 'Pil-höger' för att öppna objektet, om att objektet inte har några barn.

Om man har satt 'advanced user' kan man även ändra värdet på attribut med 'Pil-höger'. Detta har då samma funktion som 'Change value' i menyn.

## ***Navigera med musen***

Välj ut en hieraki genom att klicka på texten.

Öppna/stäng en map genom att klicka på mappen eller dubbelklicka på texten till höger om mappen.

Öppna ett objekt genom att klicka på mappen/lövet med Shift MB1, eller dubbelklicka på texten med Shift MB1.

# Trace

Trace används för att felsöka i plc-program.

Med trace kan man titta på koden i ett plc-program och se status för olika plc-objekt.

## Öppna trace

Trace öppnas genom att välja ut ett PlcPgm objekt i runtime-navigatören och aktivera 'Open Plc' i menyn. Ett visst PlcPgm-objekt hittar man genom att navigera i databasen, i menyentry't *System//PlcPgm* i xtt standard-menyn som listar alla plc-program, eller med kommandot '*show plcpgm*'. Man kan även öppna trace med kommandot '*open trace*'.

## Trace-fönstret

När man har öppnat trace för ett PlcPgm-objekt får man upp ett huvud-fönster med plc-koden, och ett navigations-fönster. Plc-koden kan vara ganska omfattande, vilket gör att man inte kan titta på all kod på en gång i huvudfönstret. Med hjälp av navigations-fönstret kan man zooma in den del av koden man är intresserad av. Den del av koden som visas i huvudfönstret visas som en fyrkant i navigations-fönstret, och denna kan man flytta på genom att dra med MB1, eller ändra storlek med MB2.

Det finns tre olika moder i trace-fönstret: view, trace och simulate.

## View

I view-mode visas plc-koden. Vissa plc-objekt innehåller under-fönster med plc-kod. Dessa öppnas med *Shift/DubbelKlick/MB1*.

## Trace

I trace-mode kopplas logiska plc-objekt till aktuella värden i databasen. Om värdet är 1 röd-markeras objektet. För analoga objekt och lite mer komplexa logiska objekt finns det inte något självklart attribut som kan kopplas till röd-markeringen. Dem kan man koppla sk analysobjekt till. Genom att dra med MB2 från ett utgång på ett objekt, sedan släppa MB2 på något fritt utrymme i fönstret, skapas ett analys-objekt som visar värdet på utgången. En uppsättning analys-objekt kan sparas genom att aktivera *File/SaveTrace* i menyn och återskapas vid ett senare tillfälle med *File/RestoreTrace*.

## Simulate

Simulate-mod kan användas vid simulering av ett PlcPgm. Förutom de funktioner som finns i trace-mod kan man även sätta värden på logiska plc-objekt med *Shift/Ctrl/DubbelKlick MB1*. Simulate-mod kräver att användaren har privilegierna *System* eller *RtWrite*.

## Menu

| File/Close                            | Stänger fönstret.   |
|---------------------------------------|---|
| File/SaveTrace                        | Sparar samtliga analys-objekt i fönstret.   |
| File/RestoreTrace                     | Återskapar tidigare sparade analys-objekt.  |
| Functions/Display object in navigator | Visar utvalt plc-objekt i runtime-navigatören.  |
| Functions/Collect insert              | Lägger in objektet i runtime-navigatörens samlingsbild (det attribut som objektet kopplats till i databasen). |
| View/Zoom in                          | Zoomar in.  |
| View/Zoom out                         | Zoomar ut.  |
| View/Zoom reset                       | Återställer till ursprunglig zoom-faktor.   |
| Mode/View                             | Visar plc-koden utan koppling till databasen.   |
| Mode/Trace                            | Trace-mod.  |

|               |                  |
|---------------|------------------|
| Mode/Simulate | Simulerings-mod. |
|---------------|------------------|

# Systembilder

Det finns ett antal systembilder som sammanställer information ur databasen och pooler. Systembilderna kan öppnas från standard-menyn i xtt eller med 'show' kommandot.

## ***System/Nethandler/Link***

Bilden visar de noder aktuell nod har QCOM kontakt med.

| Kolumn  | Beskrivning   |
|---------|---|
| Node    | Nodnamn   |
| Os      | Operativsystem och hårdvara för noden.                        |
| Link    | Status för länken. Kan vara Up, Active, Connected eller Down. |
| UpCount | Antal gånger länken till noden har gått upp.                  |
| TimeUp  | Senast länken gick upp.                                       |

## ***System/Nethandler/Subscription Client***

Visar prenumerationer som aktuell node har begärt från andra noder.

På första raden visas antalet prenumerationer på den egna noden, på andra noder och på okända objekt. Observera att antalet okända prenumerationer, om denna är större än noll när samtliga noder är uppe, indikerar att det finns prenumerationer på objekt som inte existerar.

| Kolumn    | Beskrivning                                |
|-----------|--|
| Subid     | Identitet på prenumerationen.              |
| Time      | Senaste ankomst för prenumerations-data.   |
| Count     | Antal gånger data har mottagits.           |
| Node      | Node som skickar data för prenumerationen. |
| Size      | Storlek i bytes på prenumerations-data.    |
| Attribute | Attribut för prenumerationen.              |

## ***System/Nethandler/Subscription server***

Visar prenumerationer som andra noder har begärt från aktuell nod.

| Kolumn    | Beskrivning   |
|-----------|---|
| Subid     | Prenumerations identitet.                           |
| Count     | Antal gånger data för prenumerationen har skickats. |
| Node      | Nod som data skickas till.                          |
| Size      | Storlek på data byte.                               |
| Offset    | Offset för prenumerations-attributet i objektet.    |
| Attribute | Attribut som prenumereras på.                       |

## ***System/Communication/RemNode***

Visar data för all RemNode-objekt på noden.

## ***System/Communication/RemTrans***

Visar data för alla RemTrans-objekt på noden.

När man öppnar en remtrans kan man se transbuffrarna i form av en c-struct. Detta kräver att struct-namnet och h-filen där structen finns definierad är inlagd in i remtrans-objektets StructName- resp StructFile-attribut. Se kommandot `show object /type` för de begränsningar som gäller vid tolkning av c-structar.

## **System/Device**

Visar data för alla kort-objekt på noden.

## **System/PlcThread**

Visar data för alla plc-trådar på noden.

## **System/PlcPgm**

Listar alla PlcPgm på noden.

## **System/Logging**

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 xtt's samling-bild.
- Gå in i loggnings-bilden för ett ledigt entry.
- Lägg in värden på tid, loggfil och typ, om default-värdena inte duger. Typen kan vara Cont(1) eller Event(2). Cont innebär att värden loggas kontinuerligt med angiven periodtid, Event innebär att värdena loggas när de ändrar värde.
- Klicka på *Insert* för att lägga in de attribut som valts ut i samlingsbilden.
- Klicka på *Start* för att starta loggningen. När loggningen är startad sätts 'Active' till 1.
- Stoppa loggningen genom att klicka på *Stop*.
- Titta på loggningen genom att klicka på *ShowFile*.

För övriga parametrar se kapitlet Logging och kommandot '*logging*'.

# Klassbilder

En klassbild presenterar innehållet i ett objekt av en viss klass. Alla objekt som tillhör klassen kan visas med klassbilden. En klassbild öppnas genom att man väljer ut objektet i navigatören och aktiverar *Functions/Open object graph* (Ctrl/G), eller med kommandot *'open graph /instance'*.

Klassbilder finns bl a för klasserna PID, Mode, DsTrend, PlotGroup samt för signaler och kanaler.

Behörigheten för att ändra i en klass-bild varierar mellan olika klasser. I avsnittet behörighetskontroll finns beskrivet vilka privilegier som krävs.

## Användarklasser

Man kan göra klassbilder till användarklasser på följande sätt. Skapa en Ge-bild med samma namn som klassen, men med små bokstäver. I kopplingar till olika attribut i bilden skriver man '\$object' där man normalt skriver ett objekts-namn, t ex \$object.Length##Float32. Kopiera .pwg-filen till \$pwrp\_exe och öppna bilden från menyn med *Functions/Open object graph* (Ctrl/G) eller med kommandot *'open graph /instance'*.

## Hierarkibilder

Om det finns flera hierarkier som liknar varandra kan man göra en gemensam bild för dem. Bilden för en hierarki öppnas med kommandot *'open graph/instance'* genom att skicka med hieraki-namnet som instans.



# Trendkurvor

Det finns olika typer av trendkurvor i xtt. Här beskrivs de som konfigureras med DsTrend- och PlotGroup-objekt.

En trendkurva visar värdet av en signal under en den senaste tiden i form av en kurva. Tidsperioden kan variera beroende på hur ofta nya värden lagras. Värden för en signal lagras i ett DsTrend-objekt som har plats för 478 värden.

Man kan visa flera trender i samma diagram genom att kombinera ett antal DsTrend-objekt i en PlotGroup. I ett PlotGroup-objekt finns det plats för 20 DsTrend-objekt och här kan man även ange skal-områden för koordinat-axlar.

En trendkurva öppnas genom att välja ut ett DsTrend-objekt eller ett PlotGroup-objekt och välja *'Functions/Open Object Graph'* i menyn, eller med kommandot `'open trend'`.

## Kurvfönstret

Kurvfönstret visar trendkurvorna i diagramform, med tiden på x-axeln. Nutid (0) ligger längst till höger i diagrammet och kurvorna förskjuts åt vänster när nya mätvärden registreras.

Till vänster visas axlar för skalområden för de olika trenderna. Bakgrundsfärgen för en axel visar vilken trendkurva den motsvarar i diagrammet. I Diagrammet finns horisontella linjer, och genom att klicka på en koordinataxel kan man få de horisontella linerna att anpassa sig efter just denna axel. Med dubbelklick på en axel kan man ändra min och max-värde för denna.

Med *View/Show names* (Ctrl/W) i menyn visas signalnamnet för respektive kurva. Genom att klicka i den färgade rutan framför namnet kan man påverka om kurvan ska döljas eller ritas ut. I diagrammet kan maximalt 10 kurvor ritas ut åt gången, dvs max 10 kurvor kan vara markerade för att ritas samtidigt.

Under diagrammet finns ett navigations-fönster. I detta visas trendkurvorna i hela sin utsträckning i x- och y-led. Den del som visas i huvud-diagrammet markeras med en svart rektangel. Genom att flytta på rektangeln med MB1 och ändra storlek med MB2 kan man påverka vilket område som visas i huvud-diagrammet.

Man kan även zooma från menyn med *View/Zoom in* (Ctrl/i) resp *View/Zoom out* (Ctrl/o). Från menyn finns också funktioner för att visa kurvorna som fyllda, och för att få ljus bakgrundsfärg i diagrammet.

# Logging

Loggingsfunktionen i xtt för det möjligt att logga värden i realtids-databasen på en fil. Detta kan användas t ex vid felsökning av snabba förlopp där man inte hinner se samspelet mellan olika signaler på annat sätt.

## Starta loggningen

Xtt kan hantera upp till tio loggfiler samtidigt och i varje loggfil kan hundra parametrar loggas. En logging hanteras från loggingsbilden under System/Logging. Här finns tio olika entryn att välja mellan, där varje entry hanterar en loggfil.

Man startar en logging genom att samla ihop de parameterar som ska loggas i xtt's samling-bild. Därefter går man in i ett ledigt entry och klickar på insert-kappen. De insamlade parametrarna kopieras då in i loggnings-entryt. Lägg in cykeltid och klicka på start-kappen. Loggningen är nu igång och detta markeras med en etta i Active.

Loggningen stoppas genom att klicka på stopp-kappen. Man kan nu titta på resultatet genom att klicka på *ShowFile*. ShowFile öppnar ett kurvfönster som visar loggningen i ett diagram.

Loggningen kan vara av två typer, antingen loggas värdet av parametrarna kontinuerligt med en viss frekvens (kontinuerlig logging), eller loggas en parameter varje gång dess värde ändras (händelsestyrd logging).

### **Kontinuerlig logging**

Värdet av parametrarna i entryt loggas vid varje cykel. På filen skrivs dessutom tiden sedan loggningen startades. Filen kan visas i xtt's kurv-fönster med 'ShowFile' eller läsas in av tex Excel för vidare bearbetning.

### **Händelsestyrd logging**

En parameter i entryt loggas om värdet av parametern har ändrats. På loggfilen skrivs tiden för ändringen och det nya värdet. Denna typ av logging kan inte visas med 'ShowFile'.

### **Buffer**

Loggningen sker först intern i en buffer. När bufferten är full töms den på fil. Vid snabba tidskritiska förlopp är det lämpligt att öka bufferstorleken för att undvika störningar som kan uppstå när bufferten skrivs på fil.

### **Lagra**

Med *Store/Restore* funktionen kan man lagra och återskapa uppsättningen i ett loggings-entry.

### **Kommandon**

Loggingsfunktionen kan även styras med kommandot 'logging' och därmed initieras med ett script. Observera att *Store*-funktionen lagrar entryt i form av ett script som kan användas utgångspunkt för ev modifieringar.

### **Villkorlig logging**

I *ConditionParameter* kan man ange en digital parameter som styr loggningen. Logging sker enbart om ConditionParameter'n är true.

## Attribut

| Attribut | Beskrivning |
|----------|-------------|
|----------|-------------|

|                    |  |
|--------------------|--|
| Active             | Visar om loggningen är startad.  |
| Insert             | Kopierar parameterar från samlingsbilden till loggings-entryt.   |
| Start              | Startar loggningen.  |
| Stop               | Stoppa loggningen.   |
| Store              | Lagrar uppsättningen och parameterar i loggings-entryt.  |
| Restore            | Läser in senaste lagrade uppsättning.  |
| ShowFile           | Visar loggings-resultatet i kurv-diagram. Endast kontinuerlig logging kan visas.                       |
| Time               | Cykeltid i ms.   |
| File               | Namn på loggfil.   |
| Type               | Typ av logging. 1: kontinuerlig, 2: händelsestyrd.   |
| BufferSize         | Storlek på buffer i 512 bytes block.   |
| FillBufferStop     | Stoppa loggningen när buffern är full.   |
| ShortName          | Dölj hieraki-leden i namnet på loggning-parameterarna.   |
| ParameterX         | Parameterar som ska loggas.  |
| ConditionParameter | Villkors parameter. Om en parameter har angetts här sker enbart loggningen om parametern har värdet 1. |

# Javabilder

Bilder gjorda i JBuilder eller någon annan java-IDE kan visas i xtt.

## ***Öppna en javabild***

Javabilder öppnas med kommandot 'open jgraph' där man skickar med namnet på den java-klass som ska startas.

För kunna öppna java-klasser startar xtt java i en subprocess. Vidare öppnas en QCOM-länk mellan xtt och java-processen, och över denna skickas information i form av xtt-kommandon. Java processen exekverar kommandot 'open jgraph' medan xtt exekverar övriga kommandon. Om ett kommando dyker upp i java-processen, t ex från en tryckknapp av kommando-typ, slussas detta till xtt. Detta gör att man t ex kan öppna en ge-bild från en java-bild.

## ***Stänga en javabild***

En javabild stängs med *Close* i menyn.

## ***Konfigurering***

Java processen startas genom att i OpPlace-objektets OpWinProgram lägga in "jop". Dessutom måste man se till att klasser som ska öppnas finns i CLASSPATH.

# Behörighetskontroll

Varje gång xtt utför en operation som förändrar något värde i databasen, sker det först en behörighetskontroll. Om användaren har behörighet att utföra en ändring, genomförs ändringen, annars refuseras den. Behörigheten beror den aktuella användarens privilegier, som hämta ifrån proview's användar-databas. De privilegier som är aktuella i runtime-miljön är RtRead, RtWrite, System, Maintenance, Process, Instrument, Operator1, Operator2, ..., Operator10.

## ***Runtimenavigatören***

För att göra förändringar från runtimenavigatören krävs *RtWrite* eller *System*. Detta gäller även kommandot 'set parameter'.

## ***Trace***

För att göra ändringar från trace i simulerings-mod krävs *RtWrite* eller *System*.

## ***Klassbilder***

Generellt för klassbilder krävs det *RtWrite* eller *System* för att göra ändringar från dem. Det finns några undantag. I ChanAi och ChanAo har även *Instrument* behörighet att ändra signal-områden, och att sätta en Ao i test-mod. I PID- och Mode-bilderna har *Process* behörighet att ändra förutom *RtWrite* och *System*. I Mode-bilden kan man dessutom styra behörigheten genom att öppna den med kommandot 'open graph' och använda /access funktionen.

## ***Ge och java-bilder***

I Ge och java-bilder kan konstruktören i varje inmatingsfält eller tryckknapp bestämma vilka vilka privilegier som krävs för att göra en ändring.

# Funktionstangenter

Man kan använda funktionstangenterna för att snabbt ta upp bilder eller påverka objekt i databasen.

## Funktionsanrop

Följande funktioner kan anropas när en funktionstangent trycks ner.

### ***SetDig()***

Sätter ett attribut av typen Boolean. Namnet på attributet ges som argument.

### ***ResetDig()***

Återställer ett attribut av typen Boolean. Namnet på attributet ges som argument.

### ***ToggleDig()***

Togglar värdet på ett attribut av typen Boolean. Namnet på attributet ges som argument.

### ***Command()***

Exekverar ett xtt-kommando. Kommandot ges som argument.

De flesta xtt-kommandon kan användas här, t ex kommandon för att öppna bilder, öppna larm och händelse-listor, kvittera larm.

## Resursfil

Kopplingen till funktionerna läggs i filen Rt\_xtt (OBS! case sensitive) på inloggnings-katalogen. Detta är en X-windows resurs-fil och för närmare beskrivning av syntax se *X Windows System User's Guide*. I resursen hotkeys lägger man kopplingen mellan tangenter och funktions-anrop.

### Exempel

```
! Configuration of Global function keys
*hotkeys: \
  <Key>F1: Command(event ack /prio=A) \n\
  <Key>F2: Command(event ack /prio=NOA) \n\
  <Key>F3: Command(show alarm) \n\
  <Key>F4: Command(show event) \n\
  <Key>F5: Command(close all/except=nu4_oversikt) \n\
  <Key>F6: Command(open graph /object=-OpNu4-nu4_översikt) \n\
  <Key>F7: Command(open graph /object=-OpNu4-nu4_status_trp) \n\
  <Key>F8: Command(open graph /object=-OpNu4-nu4_alla_plåtar) \n\
  <Key>F9: Command(open graph /object=-OpNu4-nu4_vägval) \n\
  <Key>F10: SetDig(vhx-nu4-pf-InTillRikt.ActualValue) \n\
  <Key>F11: SetDig(vhx-nu4-pf-EjRikt.ActualValue) \n\
  <Key>F12: SetDig(vhx-nu4-pf-InMellanHög.ActualValue) \n\
  Shift<Key>F7: Command(open graph /object=-OpNu4-nu4_sts)
```

# Hjälp

Xtt kan användas för att visa upp hjälp-texter, och navigera mellan olika ämnen i hjälptexterna. Här beskrivs hur man hittar ett ämne, hur man navigerar, och hur man skriver en hjälp-fil.

## Få hjälp

Hjälp kan öppnas på följande sätt

### **Operatörsfönstret**

I operatörsfönstrets mitt-del finns en hjälp-knapp som visar hjälp-texter för projektet.

### **Meny**

*Help/Overview* ger hjälp om Proview och operatörs-miljön. *Help/Projekt* ger hjälp om projektet.

### **Kommando**

En hjälp-text öppnas med rtt-kommandot `help`. Till `help` skickar man med ämnet man är intresserad av. Kommandot `help help` visar `text` hur man använder kommandot `help`.

### **Tryckknappar i bilder**

Tryckknappar i bilder kan kopplas till hjälp-texter. Det sker mha en tryckknapp av Command-typ och help-kommandot.

## Navigera i hjälptexten

Rader som innehåller länkar till andra ämnen markeras med en pil. Genom att klicka på pilen eller dubbelklicka på raden öppnar man texten som länken pekar på.

Den översta radan är markerad med en vänster-pil. Genom att klicka på den, eller dubbelklicka på raden, kommer man tillbaka till föregående sida.

Man kan även navigera med tangentbordet genom att välja ut en länk med pil-upp och pil-ner. Öppna länken med pil-höger och återgå med pil-vänster.

## Hjälpfiler

Xtt läser hjälptexter från hjälpfiler. För skriva hjälptexter för ett projekt skapar man filen `$pwrp_exe/xtt_help.dat`.

### **Syntax**

Hjälpfilen byggs upp med taggar, som påverkar hur hjälptexten visas. Det finns även taggar för att skapa länkar till ämnen i hjälpfilen.

### **<topic>**

<topic> inleder ett ämne och ska skrivas på radens första position. Topic-taggen ska följas av det sökord som hjälpfunktionen ska söka efter. Alla följande rader fram till </topic> taggen kommer att visas som text för ämnet.

<topic> 'key'

## **</topic>**

Avslutar ett ämne. </topic> ska skrivas på radens första position.

### **Exempel**

```
<topic> start engine
The engine will be started by...
</topic>
```

Följande kommando visar texten för detta ämne

```
x tt> help start engine
```

## **<bookmark>**

Ett bokmärke är en rad inom ett ämne som kan visas med link-taggen eller med /bookmark kvalifieraren i help-kommandot. Bookmark-taggen ska placeras i slutet på en rad och ska följas av ett namn.

```
'some text' <bookmark> 'name'
```

### **Exempel**

```
This is a bookmark <bookmark> first_engine
```

Följande kommando kommer att visa texten för ämnet och skrolla till bokmärket

```
x tt> help start engine /bookmark=first_engine
```

## **<link>**

<link> taggen är en länk till ett annat hjälp-ämne. <link> taggen ska placeras i slutet på en rad. När raden med länken aktiveras med dubbel-klick, pil-höger eller return, kommer texten för ämnet att visas. Link-taggen ska följas av ämnet, och kan även följas av ett bokmärke och hjälpfilen som innehåller ämnet, separerade med komma-tecken. En rad som innehåller en länk kommer att markeras med en pil.

```
'some text' <link> 'topic'[, 'bookmark'][, 'helpfile']
```

I <link> kan man även ange en URL till en .html eller .pdf fil. I så fall startas en webbläsare med den angivna URL'en. URL'en får inte innehålla några av de URL-symboler som kan konfigureras i WebBrowserConfig objektet.

### **Exempel**

```
Link to first engine <link> show engine, first_engine
Link to datasheet <link> http://platon:8080/pwr/fotocell.html
```

## **<index>**

<index> taggen är en speciell länk som visar innehållet i hjälpfilen, dvs en lista på alla ämnen i alfabetisk ordning.

```
'some text' <index>
```

## **<h1>**

<h1> taggen visar en rad som en rubrik med större text-storlek. Taggen ska placeras i början på raden.



En rubrik-rad kan inte innehålla några länkar.

### Exempel

```
<h1>This is a h1 header
```

### <h2>

<h2> taggen visar en rad som en rubrik med fet text omgiven av gråa linjer. Taggen ska placeras i början på raden. En rubrik-rad kan inte innehålla några länkar.

### <b>

<b> taggen visar en rad med fet text. Taggen ska placeras i början på raden.

### Exempel

```
<b>This is a bold line
```

### <t>

<t> taggen gör det möjligt att skriva kolumner. Endas tre kolumner (två <t> taggar) tillåts.

### Exempel

```
Col1 <t> Col2 <t> Col3
```

### <include>

Inkluderar en annan hjälp-fil. <include> taggen ska inte placeras i ett ämne.

```
<include> 'filename'
```

### <i>

<i> skriver ut en rad utan att ta hänsyn till eventuella taggar i raden. Ska placeras i början på raden.

### Exempel

```
<topic> helpfile_example  
Start and stop of engines.
```

```
Engine 1 <link> helpfile_example, bm_engine_1  
Engine 2 <link> helpfile_example, bm_engine_2  
Characteristics <link> helpfile_example, bm_char
```

```
<h1>Engine 1 <bookmark> bm_engine_1  
Start engine one by pressing the start button.  
Stop engine one by pressing the stop button.
```

```
<h1>Engine 2 <bookmark> bm_engine_2  
Start engine two by pressing the start button.  
Stop engine two by pressing the stop button.
```

```
<h2>Characteristics <bookmark> bm_char
```

```
<b><t>Engine1 <t>Engine2
Max speed <t> 3200 <t> 5400
Max current <t> 130 <t> 120
</topic>
```

### **Generera hjälpfil från wb\_load-filer**

Med programmet co\_convert kan man generera en xtt hjälpfil från en wb\_load-fil

Hjälpfiler för xtt genereras med kommandot

```
co_convert -xv -d $pwrp_exe/ $pwrp_db/userclasses.wb_load
```

Kommandot genererar en hjälpfil \$pwrp\_exe/'volymnamn'\_xtthelp.dat som man lämpligt-vis lägger en länk till i projektets xtt-hjälpfil \$pwrp\_exe/xtt\_help.dat:

Exempel för klassvolyman vhn2r:

```
<topic> index
...
<b>Användar klasser<link>cvolvhn2r,"",$pwrp_exe/cvolvhn2r_xtthelp.dat
</topic>
...
<include> $pwrp_exe/cvolvhn2r_xtthelp.dat
```

### **Generera html-filer från hjälpfil**

co\_convert kan även konvertera en xtt-hjälpfil till html. Detta gör att knappar i en process-bild som använder help-kommandot även fungerar när bilden tas upp på webben. Man kan även lägga en länk i menyn till html-filerna mha ett WebLink objekt.

Kommandot

```
co_convert -d $pwrp_web -t $pwrp_exe/xtt_help.dat
```

konverterar filen \$pwrp\_exe/xtt\_help.dat och alla filer som den inkluderar. Html-filerna läggs på \$pwrp\_web. Kommandot exekveras av *Generate web* i Ge-editor.

# Popup meny

Om man klickar med MB3 på ett objekt öppnas en popup-meny. Meny-alternativen varierar beroende på objekts-typ och konfigurerings. Här visas exempel på de vanligaste meny-alternativen.

| Meny-alternativ | Beskrivning   |
|-----------------|---|
| Graph           | Visar process-bild för objektet   |
| Trend           | Visar trendkurva för objektet   |
| Help            | Visar hjälptext för objektet  |
| Open Object     | Öppnar attribut-editorn som visar värdet på de olika attributen i objektet. |
| Open Plc        | Öppnar plc-fönstret där objektet förekommer.                                |
| Help Class      | Visar hjälptext för klassen som objektet tillhör.                           |
| Crossreferenses | Visar var objektet refereras i plc-koden.                                   |
| DataSheet       | Visar datablad för objektet   |
| Collect         | Lägger in objektet i xtt's samlings-bild.                                   |

## Konfigurerings

Vissa meny-entryn i popup-menyn kräver en viss konfigurerings. För varje entry finns dels en filter-funktion som avgör om ett entry ska visas eller ej, samt en metod som anropas när ett entry har aktiverats. Här beskrivs de metoder som är generella för alla klasser. Det finns även metoder som är dedicerade till objekt av en viss klass. Metoderna kan fungera även på användar-klasser om de innehåller de attribut som de olika metoderna använder (DefGraph, DefTrend, HelpTopic och DataSheet).

### Graph

Öppnar en Ge-graf. Metoden letar efter attributet 'DefGraph' i objektet eller i något objekt som ligger ovanför objektet i hierarkin. DefGraph attributet ska innehålla ett XttGraph objekt som öppnar Ge-grafen.

DefGraph-attributet ska vara av typen pwr\_tObjid, och finns bl a i \$PlantHier och signal-objekt.

### Help

Visar en xtt hjälp-text. Metoden letar efter attributet 'HelpTopic' i objektet och visar hjälptexten för detta ämne.

HelpTopic-attributet ska vara av typen pwr\_tString40 och finns bl a i \$PlantHier och signal-objekt.

### Trend

Öppnar en trendkurva. Metoden letar efter attributet 'DefTrend' i objektet. DefTrend attributet ska innehålla ett DsTrend eller PlotGroup objekt.

DefTrend-attributet ska vara av typen pwr\_tObjid och finns bl a i \$PlantHier och signal-objekt.

### DataSheet

Startar en web-browser som visar att datablad för objektet. Metoden letar efter attributet 'DataSheet' och öppnar den URL som attributet innehåller. URL'en kan innehålla symboler definierade i ett WebBrowserConfig-objekt.

DataSheet-attributet ska vara av typen pwr\_tURL och finns bl a i \$PlantHier och signal-objekt.

### Help Class

Metoden fungera automatiskt för bas och systemklasser. Vill man att de ska fungera även på användarklasser ska man genererar xtt hjälp-filer från wb\_load-filen och göra denna tillgänglig för help-funktionen i xtt. Se avsnittet *Hjälp*.

**Open Plc**

Open plc kräver att flw-filer är distribuerade (flw-filer skapas på \$pwrp\_load när ett plc-fönster sparas).

**Crossreferences**

Korsreferenser kräver att rtt-filer är skapade och distribuerade (skapas med 'create rttfiles' i Utilities).

# Konfigurering

## Operatörsmod

En operatörsplats konfigureras med ett OpPlace-objekt och ett User-objekt. För att konfigurera process-bilder används XttGraph-objekt.

## Underhållsmod

I underhålls-mod konfigureras xtt med ett RttConfig-objekt, med namnet RttConfig. Objektet ska ligga under nod-objektet.

### ***Symbolfil***

I RttConfig-objektet kan man ange en symbol-fil. Denna exekveras när xtt startas, och man kan t ex modifiera menyn eller definiera symboler där.

Här följer några användbara kommandon.

Genväg till en databas-hieraki

```
define rb9 "show children /name=hql-rb9"
```

Genväg till en ge-bild

```
define my_graph "open graph my_graph"
```

Skapa meny-alternativ i xtt menyn

```
create item /text="Maintenance"/menu/destination="DataBase"/before  
create item /text="My graph"/command="open graph my_graph"/dest=Maintenance/first
```

Ta bort meny-alternativ ur standard-menyn

```
delete item /name=exit  
delete item/name=system-nethandler
```

# Kommandon

## Allmänt

Kommando-tolken i xtt fungerar som en knutpunkt i xtt, eftersom det är genom kommandon de flesta funktioner i xtt aktiveras.

Kommandon kan matas in interaktivt från kommandoraden som öppnas med *Functions/Command* (Ctrl/B) i meny. Det finns en recall-funktion så att men med PilUpp och PilNer kan välja mellan tidigare inmatade kommandon.

Kommandon kan även exekveras från tryckknappar i Ge-bilder, de konfigurerbara knapparna i operatörs-fönstret och från funktionstangenter.

# add

Adderar objekt eller parameterar till aktuell lista.

## add parameter

Adderar ett attribut till en lista av attribut och visar innehållet i attributet.

En lista med attribut skapas med kommandot 'show parameter' och fler attribut kan adderas till listan med 'add parameter'.

```
x tt> add parameter /name=
```

### **/name**

Namn på ett attribut.

Namnet kan skrivas utan '/name='.

## add menu

Adderar ett meny-entry till xtt menyn. Meny-entryt läggs sist i aktuell meny.

```
x tt> add menu /text= /command=
```

```
x tt> add menu /text= /object=
```

### **/text**

Text på menyentry.

### **/object**

Objekt som visas när menyentry aktiveras.

### **/command**

Xtt-kommando som exekveras när meny-enryt aktiveras.

# collect

## collect

Addera ett attribut till samlings-listan.

Om name-kvalifieraren utelämnas adderas det utvalda attributet.

```
xtt> collect [/name=]
```

### **/name**

Namn på attributet.

### **Exempel**

```
Xtt> collect /name=hql-hvk-Start.ActualValue
```

## collect show

Visa samlingslistan.

```
xtt> collect show
```

## collect clear

Tömmer samlingslistan.

```
xtt> collect clear
```



# close

## close alarmlist

Stänger larmlistan.

```
xтт> close alarmlist
```

## close eventlist

Stäng händelselistan.

```
xтт> close eventlist
```

## close graph

Stäng en ge-bild.

Ge-bilden kan anges med filnamn på pwg-filen eller som XttGraph objekt.

```
xтт> close graph 'filename' [/instance=]  
xтт> close graph /object=
```

### **/object**

Ett XttGraph objekt.

Om namnsträngen inleds med '\*' kommer \* att ersättas av nod-objektet.

### **/instance**

Stänger en objekts- eller hieraki-bild för denna objekt/hierarki.

## close navigator

Stäng navigatorn.

```
xтт> close navigator
```

# create

## create item

Skapa ett meny-entry i xtt.

När xtt navigatören startas innehåller den ett antal meny-entryn för att öppna databas-hierakin, visa larmlist mm. Till den här menyn kan man addera nya meny-entryn med 'create item' kommandot.

Man kan även ta bort existerande entryn med 'delete item' kommandot.

Ett meny-entry kan antingen exekvera ett kommando, eller öppna en meny.

```
xtt> create item /text= /command= /destination= /after /before  
      /firstchild /lastchild  
xtt> create item /menu /text= /after /before /firstchild /lastchild
```

### **/text**

Meny-entrys text.

### **/command**

Xtt-kommando som exekveras när entryt aktiveras.

### **/menu**

Skapar ett meny-entry för att öppna en meny.

### **/destination**

Anger var i meny-hierakin entryn ska skapas. Entryt skapas som barn eller som syskon till destinations-entryt beroende på kvalifierarna /after, /before, /firstchild och /lastchild.

### **/after**

Menyentryt läggs efter destinations-entryt.

### **/before**

Menyentryt läggs före destinations-entryt.

### **/firstchild**

Menyentryt läggs som första barn till destinations-entryt.

### **/lastchild**

Menyentryt läggs som sista barn till destinations-entryt.

### **Exempel**

```
xtt> create item /text="Motor1"/dest=Underhåll-Motorer  
      /command="open graph motor1"/first
```

## create object

Skapar ett objekt i rtdb.

Xtt måste köras i den nod som objektet ska tillhöra.

Om man i xtt tittar på den objektshierarki där objektet kommer att skapas, kommer denna ej att uppdateras förrän man har stängt hierarkin och öppnat den .

### **/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 xtt's kommandotolk översätter alla strängar till versaler som inte är omgärdade av 'dubbelfnuttar'.

## **Exempel**

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

```
xtt> cross /function="MyFunc"
```

Sökningen känslig för gemener och versaler.

```
xtt> 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]
```

# 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

Stäng navigatorn.

Om xtt är startat med operatörsfönster stängs navigatorn men xtt-processen lever vidare.

Om xtt är startad utan operatörsfönster avslutas processen.

# define

Definiera en symbol.

Symboler kan användas som kort-kommandon eller som sträng-variabler i kommandon.

```
xtt> define 'symbolnamn' 'text'
```

## Exempel

```
xtt> define p1 "show child /name=hql-hvk-pumpar-pump1"
```



# delete

## delete item

Ta bort ett meny-entry ur xtt-menyn.

Meny-entryt kan vara skapat av användaren med 'create item' kommandot, eller tillhöra xtt standardmeny.

```
xtt> delete item /name=
```

### **/name**

Namn på meny-entryt.

### **Exempel**

```
xtt> delete item /name=exit
```

# help

Visar hjälp-text för ett ämne.

Hjälp-texten hämtas från en xtt hjälp-fil.

Om ingen hjälpfil har angivits, letas först efter ämnet i hjälp-filen för proview's bassystem och sedan i hjälpfilen för projektet (\$pwrp\_exe/xtt\_help.dat).

Xtt använder navigatören för att visa hjälptexter.

```
xtt> help 'ämne' [/popnavigator] [/bookmark=] [/helpfile=]  
          [/returncommand=] [/width=] [height=]
```

## **/popnavigator**

Hjälpfönstret (navigatören) läggs överst på skärmen och ges inmatnings fokus. Detta bör användas om hjälp-kommandot används i en tryckknapp i en Ge-bild.

## **/bookmark**

Namnet på ett bokmärke i ämnes-texten. Visningen av texten kommer att positioneras på bokmärket.

## **/helpfile**

Hjälpfil som innehåller hjälp-texten för aktuellt ämne.

## **/returncommand**

Ett kommando som exekveras när man återgår från hjälp-texten.

Detta används normalt för att återgå till en huvud-hjälptext när hjälp är anropat från en kommando-knapp i Ge. Om man trycket på återgå-pilen på första raden i hjälptexten hamnar men normalt i xtt huvud-meny.

Med /returncommand kan man styra om detta så att man hamnar i en annan hjälptext.

## **/width**

Bredd på hjälpfönstret.

## **/height**

Höjd på hjälpfönstret.

# login

Inlogging med användare och passerord.

Privilegierna för användaren hämtas från användar-databasen och påverkar behörigheten.

Med 'logout' kommandot återgår man till den ursprungliga användaren.

```
xtd> login 'användare' 'passerord'
```

# logout

Loggar ut en användare och återgår till den ursprungliga användaren.

```
xtt> logout
```

# logging

Ett antal kommandon för att hantera xtt logg-funktion.

Loggnings funktionen i xtt 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.

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

Create kommandot är i xtt identiska med set-kommandot men finns kvar för att vara kompatibelt med rtt's logg-kommandon.

## 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 filspekifikation .

### **/parameter**

Namn på parameter som ska loggas ('objekts namn.attribut'). Maximalt 100 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 loggningen 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 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**

Stoppas 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.

```
xtt> 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.

# open

## open graph

Öppna en Ge bild.

Bilden kan specificeras med en .pwg fil eller som ett XttGraph objekt.

```
x tt> open graph 'filename' /width= /height= /scrollbar /menu  
      /navigator /instance= /focus= /inputempty  
x tt> open graph /object= /focus= /inputempty
```

### **/width**

Bredd i pixel.

### **/height**

Höjd i pixel.

### **/scrollbar**

Scrollbars kommer att visas i fönstret.

### **/menu**

En meny kommer att visas i fönstret.

### **/navigator**

Ett navigations-fönster kommer att öppnas.

### **/object**

Ett XttGraph-objekt. Om namnsträngen inleds med '\*', kommer \* att ersättas av nod-objektet.

### **/instance**

Öppna en objekts-bild för det här objektet.

### **/focus**

Namn på ett bild-objekt som ska få inmatnings-fokus när bilden öppnas.

### **/inputempty**

Inmatningsfältet som ges inmatnings-fokus är tomt.

## open jgraph

Öppna en java bild. Bilden kan vara exporterad från Ge eller ritad i JBuilder.

```
x tt> open jgraph 'namn'
```

## open trace

Öppna trace för ett plc-fönster. Om namn-kvalifieraren utelämnas, öppnas trace för utvalt PlcPgm eller PlcWindow.



```
x tt> open trace [/name=] [/center=]
```

### **/name**

PlcPgm eller PlcWindow.

### **/center**

Objekt i plc-fönstret som ska centreras och väljas ut.  
Endast sista ledet i objektsnamnet ska anges.

### **Examples**

```
x tt> open trace /name=hql-hvk-flow /center=Pid0
```

## **open trend**

Öppna en kurva för ett DsTrend eller PlotGroup objekt.

```
x tt> open trend [/name=] [/title=]
```

### **/name**

DsTrend eller PlotGroup. Kan vara en lista av DsTrend-objekt separerade med komma.

### **/title**

Titel på fönstret.

### **Examples**

```
x tt> open trend /name=hql-hvk-flow-Trend,hql-hvk-temperature-Trend
```

## **open operatorwindow**

Öppna the operatörs-fönstret för en operatörs-plats.

```
x tt> open operatorwindow 'opplace-objekt'
```

## **open url**

Starta en webbrowser med angiven URL. Webläsaren och eventuella symboler kan konfigureras med ett WebBrowserConfig-objekt. Om URL'en innehåller '/' måste den omgärdas av dubbelfnuttar.

```
x tt> open url 'url'
```

### **Exempel**

```
x tt> open url "$pwrp_websrv/pwrb_index.html"  
x tt> open url "http://www.java.sun.com/j2se"
```

# search

Letar efter ett objekt eller en sträng i database-hierakin.

```
xtt> search 'object'  
xtt> search /regexexpression 'expression'  
xtt> search /next
```

## **/regexexpression**

Sök-strängen är ett reguljärt uttryck.

## **/next**

Leta efter nästa förekomst av sök-strängen.

# set

## set advanceduser

Sätter eller återställer advanced user.

```
xrt> set advanceduser  
xrt> set noadvanceduser
```

## set parameter

Sätter värde på en parameter i databasen.

Kräver System eller RtWrite privilegier.

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

### **/name**

Fullständigt namn på parameter ('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
```

# setup

Xtt setup

Visar egenskaper för xtt-sessionen.

| Attribut         | Beskrivning  |
|------------------|--|
| ConfigureObject  | Visar aktuellt RttConfigure objekt.  |
| DefaultDirectory | Default filkatalog för lagrade script.   |
| Scantime         | Cykeltid. Bestämmer bl a uppdateringsfrekvensen vid visning av objekts-attribut.   |
| AlarmMessage     | -  |
| 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 vid start av xtt.                                  |
| Verify           | Script exekveras med verify, dvs utskrift av exekverade rader.                     |
| AdvancedUser     | Användaren är avancerad.   |

# show

## show file

Visar filer.

Som parameter kan en filspekifikation anges. Om filspekifikationen utelämnas visas rtt-kommandofiler på default-directoryt.

De filer som matchar filspekifikationen visas upp i en lista där man kan välja ut filerna. Om filen är ett xtt-script kan man exekvera scriptet genom att klicka på det, om det en en ge-graph kan man öppna bilden, är det en rtt\_log-fil öppnar man den in ett kurv-fönster.

```
xtt> show file
xtt> show file "$pwrp_exe/*.rtt_com"
```

### parameter

Filspekifikation. Kan innehålla wildcard.

## show object

Visar en transbuffer tolkad som en c-struct

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

### /name

Namn på objektet.

### /type

Visar ett transbuffer-objekt i form av en c-struct. /type anger namnet på den c-struct som ska användas. Xtt öppnar den include-fil som anges med /file och letar efter structen. För att xtt ska kunna tolka c-structen krävs följande begränsningar i syntaxen:

- Structen ska vara definerad med en typedef in den include-fil som anges.
- 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 ska finnas definierade i andra include-filer. Dessa include-filer måste vara inkluderade i den include-file som anges i /file.
- Arrayer får ha högst två dimensioner.
- En array av struct får ha en dimension.
- Tomma rader får ej finnas mellan elementen i structen.
- Kommentarsrader måste inledas med /\* på varje rad.
- Storlek på arrayer måste anges explicit, eller med en #define i den include-file som anges i /file.
- Logiska namn på filkataloger för filer inkluderade med #include måste vara kända i realtidsmiljön om dessa innehåller typdefinitioner som refereras i den ursprungliga structen. Include-filer angivna utan fil-katalog antas ligga på pwrp\_inc. För VAXELN antas att include-filer ligger på disk\$sys:<sys0.sysex>

### /file

Filnamn på den include-fil som innehåller definitionen av 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.

```
x tt> 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**

```
x tt> sho par /par=valueindex /class=ao
```

Visar parametern valueindex för alla ao-objekt i systemet.

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

## show version

Visa xtt version.

```
x tt> show version
```

## show symbol

Visa en, eller alla symboler.

Visa en symbol

```
x tt> show symbol 'symbol'
```

Visa alla symboler

```
x tt> show symbol
```

## show plcpgm

Lista alla PlcPgm-objekt i systemet.

```
x tt> show plcpgm
```

## show plcthread

Visa alla PlcThread-objekt.

PlcThread-objekt innehåller information och statistik för plc-trådarna.

```
x tt> show plcthread
```

## show link

Visa länkar med andra provsystem.

```
x tt> show link
```

## show logfile

Visa xtt logg-filer som ligger i aktuell filkatalog.

En logg-fil skapas av logging-funktionen i rtt eller xtt och har fil-typen .rtt\_log.

En logg-fil öppnas med dubbel-click, eller genom att välja ut filen och trycka på return.

```
x tt> show logfile
```

## show alarmlist

Öppna larmlistan.

```
x tt> show alarmlist
```

## show eventlist

Öppna händelselistan.

```
x tt> show eventlist
```

## show time

Visar systemtiden.

```
x tt> show time
```

## show user

Visa aktuell användare med privilegier.

```
x tt> show user
```

# store

Lagra en lista med objekts-attribut i form av ett xtt-script, eller lagra aktuell symboltabell i form av ett xtt-script.

Attributs-listan respektive symboltabellen kan återskapas genom att scriptet exekveras. Det sker antingen från kommando-prompten med @'filnamn' eller genom att scriptet kan återfinnas under 'Store' i xtt-menyn.

Om /collect används, kommer attribut-listan att återskapas i samling-listan.

```
xtt> store 'filename' [/collect]  
xtt> store 'filename' /symbols
```

## **/collect**

Attribut-listan kommer att återskapas i samlings-listan.

## **/symbols**

Aktuell symboltabell lagras.



# Script

Rtt-script är ett sätt att programmera rtt-kommandon. Scripthanteraren ger dessutom möjlighet till att göra beräkningar, utföra villkors-satser, loop-satser, deklarerara variabler 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 kan 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 deklarerars 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.

### Exempel

```
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 scriptet 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 eller break-satserna kan man hoppa till nästa loop eller hoppa ur loopen.

### Exempel

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

### break

En break-sats kommer att orsaka att exekveringen försätter 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 försätter 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 modifieras 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 en konstant 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.
- Ytterligare 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***

| <b>Funktion</b>  | <b>Beskrivning</b>   |
|------------------|--|
| 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***

| <b>Funktion</b> | <b>Beskrivning</b>                 |
|-----------------|------------------------------------|
| 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-tangenter 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-objektet.

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 = element( 1, ",", str);
```

# exit()

**int exit()**

## **Beskrivning**

Avslutar exekveringen av ett script.

## **Exempel**

```
exit();
```

# extract()

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

## Beskrivning

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

## Argument

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

## Exempel

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



# fclose()

**int fclose( int file)**

## Beskrivning

Stänger en öppnad fil.

## Argument

int        file        fil-id returnerad av fopen.

## Exempel

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

# felement()

**string felement( int number, string delimiter)**

## Beskrivning

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

Returnerar det extraherade elementet.

## Argument

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

## Exempel

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

# fgets()

**int fgets( string str, int file)**

## Beskrivning

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

## Argument

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

## Exempel

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

# fopen()

**int fopen( string filespec, string mode)**

## Beskrivning

Öppnar en fil för läsning eller skrivning.

Returnerar en fil-identitet. Om filen inte kan öppnas, returneras 0.

## Argument

|        |          |              |
|--------|----------|--------------|
| string | filespec | Namn på fil. |
| string | mode     | Access mod.  |

## Exempel

```
int infile;
int outfile;

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

# fprintf()

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

## Beskrivning

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

Returnerar antal utskrivna tecken.

## Argument

|                |        |  |
|----------------|--------|--|
| int            | file   | Fil-id returnerat av fopen.  |
| string         | format | Utskifts 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 | Utskifts 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 | Utskifts 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 strrchr( 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   Teckensekvens 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 );
```



# Appendix A

## Exempel på loggning

Exempel på loggning med xtt (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
```