Uppgradering till V4.2.0

Vägledning

Revision 0.1	**Revisionshistorik** **2006-04-01** **v0.1**	**cs**

Uppgradering till V4.2.0: Vägledning

av Claes Sjöfors

Innehållsförteckning

Kapitel 1. Introduktion

1.1. Uppgradering till Proview 4.2.0

Detta dokument beskriver nya funktioner i Proview V4.2.0, samt hur man uppgraderar från V4.1.3 till V4.2.0.

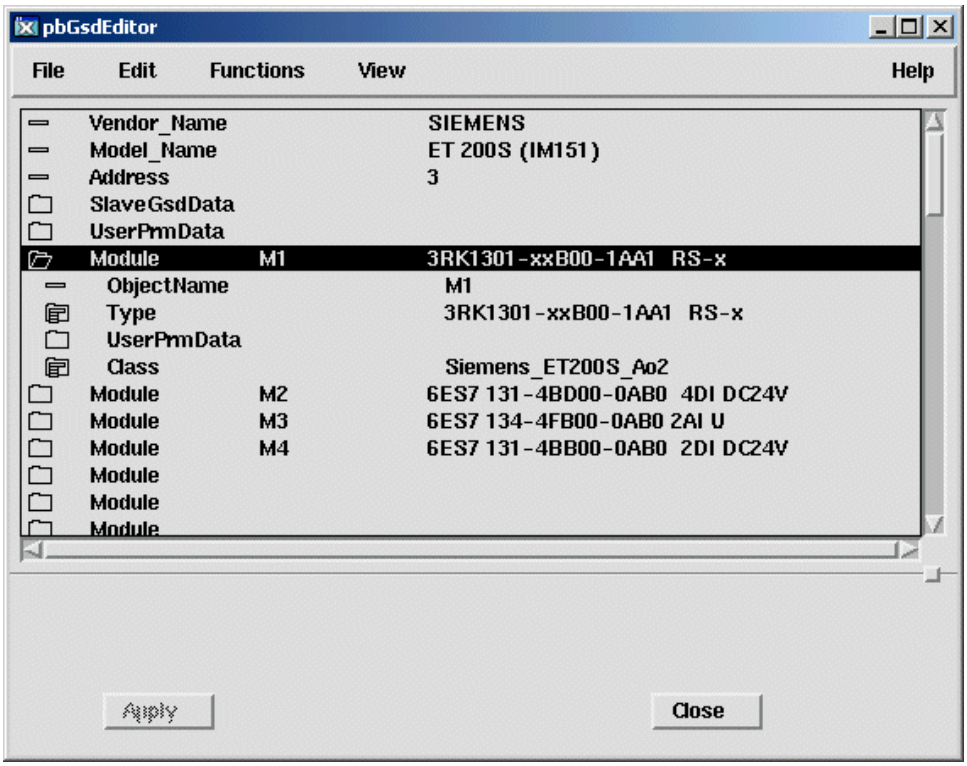
Kapitel 2. Nya funktioner

2.1. Profibus konfigurator

Konfigureringen av profibus är förändrad i V4.2.0, både när det gäller tillvägagångsätt och de objekt som används vid konfigureringen.

Man börjar genom att skapa ett masterobjekt i nodehierarkin, för Softing profiboard används **Pb_Profiboard**. Under detta läggs de slavar som finns på profibusslingan med **Pb_DP_Slave** objekt, eller objekt som är en subklass av **Pb_DP_Slave**. Använder man **Pb_DP_Slave** objektet lägger man in namn på gsd-fil, byteordning och ev flyttalsrepresentation. För vissa slavar finns specifika subclasser, t ex **Siemens_ET200S_IM151**, **Siemens_ET200M_IM153** och **ABB_ACS_Pb_Slave**. Här är gsd-filen redan specificerad i objektet, och följer även med proview-releasen.

Nästa steg är att öppna profibus-konfiguratören för varje slav, genom att aktivera Configure Slave i popupmenyn för slaven. Profibus konfiguratören läser gsd-filen och presenterar data och konfigurations alternativ för slaven. Under mappen `SlaveGsdData` visas diverse uppgifter om slaven, och under mappen `UserPrmData` konfigurationsdata för slaven.



På slaven finns plats för ett visst antal moduler, och för varje tänkbar modul finns ett modul-entry i konfiguratören. Genom att öppna ett modul-entry kan man ange typ, konfigureringsdata, objektsnamn och objektsklass för modulen.

Typ

Under Type listas alla möjliga typer finns för den aktuella slaven. Välj ut den önskade typen genom att klicka i checkboxen för typen.

Konfigurationsdata

Under UserPrmData visas konfigurerings alternativen för den valda modulen. Här anger man data och väljer mellan olika alternativ för att konfigurera modulen. Se databladet för modulen för närmare information om vad alternativen innebär.

Objektnamn

Vid konfigureringen skapar profibus konfiguratorn ett modulobjekt under slavobjektet. I ObjectName anger man ett namn på detta modulobjekt. Namnet måste vara unikt inom slaven.

Modulklass

Under ModuleClass listas möljliga klasser på det modul konfigurations-objekt som skapas under slavobjektet. Den klass som man väljer är beroende av hur dataarean som transporteras på profibussen ser ut. Det finns ett antal specifika klasser t ex **Siemens_ET200S_Ai2, Siemens_ET200SDi2, ABB_ACS_PPO4.** Dessa innehåller en förspecificerad dataarea i form av interna kanalobjekt. Om inte något specifikt modulklass passar, väljer man **Pb_Module** och specifierar senare dataarean genom att lägga kanal-objekt under modulobjektet.
När alla moduler är konfigurerade klickar man på apply, och de olika modul-objekten skapas. Nu lagras även PrmUserData konfigureringen för slaven och modulerna i attributet PrmUserData i slavobjektet, tillsammans med diverse andra data.

Nu återstår att ange Process och PlcThread för alla konfigurations objekt, samt konfiguerar eventuella kanalobjekt under Pb_Module objekten.

2.2. Bygg metoder

Kompilering av PlcPgm, skapande av laddatafiler och skapande av bootfil utförs numera av Build funktionen. Build funktionen utgörs av bygg-metoder för noder, volymer och objekt.

2.2.1. Byggmetoder för objekt

PlgPgm

Byggmetoden för ett PlcPgm kontrollerar om plc-koden är ändrad sedan senaste kompileringen. Om det är ändrad kompileras plcprogrammet med alla underfönster.

XttGraph

Byggmetoden för ett XttGraph kopierar .pwg filen från $pwrp_pop till $pwrp_exe om filen på $pwrp_pop är nyare än den på $pwrp_exe. Om grafen är en java-applet eller java-application exporteras den som java och kompileras.

WebHandler

Byggmetoden för ett WebHandler objekt skapar en hemsida för en nod (anropar Generate Web).

2.2.2. Byggmetod för volymer

Rotvolym

Rotvolymens byggmetod anropar byggmetoden för samtliga PlcPgm, XttGraph och WebHandler objekt i volymen. Om volymen är ändrad sedan laddatafiler senast skapades för volymen, skapas nya laddatafiler. Även nya korsreferensfiler skapas om detta är angett i Options.

Klassvolym

Om klassvolymen är ändrad sedan laddatafiler senast skapades för volymen, skapas nya laddatafiler, och nya structfiler för volymen.

2.2.3. Byggmetod för noder

Nodens byggmetod anropar byggmetoden för öppnad volym och skapar därefter ny bootfil för noden.

Notera

Observera att endast volymer som är öppnade byggs om. Om noden innehåller flera volymer måste andra volymer byggas separat, innan noden byggs.

2.3. PSS9000 Remote rack

Ett proview system kan nu hämta data från en PSS9000 rack via ethernet. Racken konfigureras med ett Ssab_RemoteRack objekt i nodhierarkin, och därunder konfigureras korten i vanlig ordning.

2.4. ld_node_xxx.dat

ld_node filen innehåller de noder en nod tar kontakt med via QCOM vid uppstart av proview. Filen genereras utifrån data i NodeConfig och FriendNodeConfig objekt i projektvolymen.

Tidigare har ld_node filen varit gemensam för alla noder i ett projekt på samma QCOM bus, men nu har varje nod fått en separat ld_node fil. Detta gör det möjligt att styra vilka externa noder en node tar kontakt med individuellt.

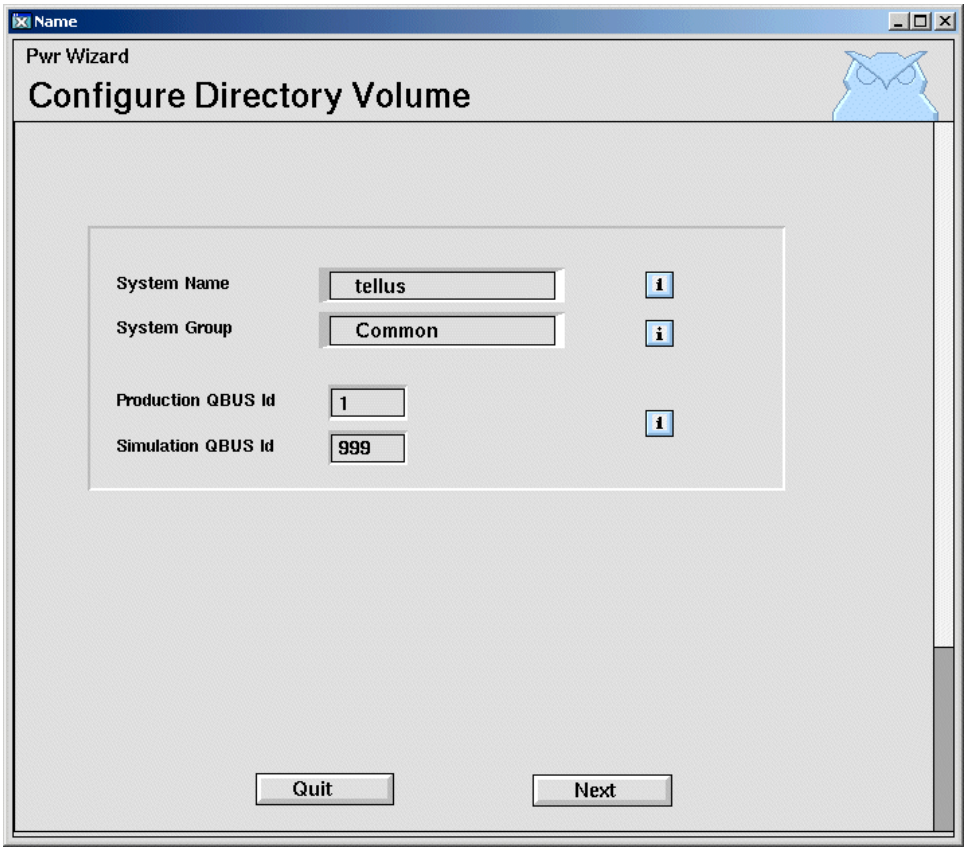
Liksom tidigare styrs detta genom att FriendNodeConfig objekt läggs in i projektvolymen. Dessa har tidigare lagts som syskon till NodeConfig objekten i en QCOM bus, och medför då att samtliga noder tar kontakt med denna externa nod. Nu kan FriendNodeConfig även läggas som barn till ett NodeConfig objekt, vilket medför att endast denna nod tar kontakt med den externa noden.

2.5. Buffring av prenumerationer borttagen

Buffringen av prenumerationer, vilken kunde ge upphov till ikappkörnings fenomen vid dålig kommunikation, är nu borttagen.

2.6. Wizard för konfigurering av projektvolymen

Konfigureringen av projektvolymen görs nu enkelt mha av en wizard som startar automatisk om man öppnar en tom projektvolym. Wizarden hämtar upp de volymer som är konfigurerade för projektet i den globala volymslistan, och skapar volym- och nod-konfigurerings objekt för dessa.



2.7. Uppdatering av klasser

Om man gjort en ändring i en klassvolym var man tidigare tvungen att dumpa ut databasen på en textfil och återladda denna för att uppdatera instanser av ändrade klasser. Nu finns det en funktion som uppdaterar instanser utan dumpning/laddning.

Varje databas lagrar alla laddatafiler för klassvolymer lokalt i filkatalogen för databasen. Det är alltså inte dbs-filerna på $pwr_load eller $pwrp_load som används när man öppnar arbetsbänken, utan de lokala dbs-filerna. Det här gör att man är oberoende av hur de globala dbs-filerna förändras. Vid uppstart av arbetsbänken jämförs versionerna på lokala dbs-filer med globala dbs-filer, och om någon global dbs-fil finns i en ny version får man ett varnings-meddelande om detta. Man kan då med kommandot 'check classes' se vilka klasser ändringen omfattar och om man har några instanser av dessa klasser. Man bör sedan aktivera Functions->Update Classes i menyn för att uppdatera klasserna och de lokala dbs-filerna.

Vid ändring av in- och utgångar i funktionsobjektsklasser har man tidigare varit tvungen att dra om kopplingarna till alla instanser. Det här har förbättrats något. Lägger man till nya ingångar eller utgångar anpassa funktionsobjektet automatisk till detta. Tar man bort in eller utgångar bör man se till att de in och utgångar som tas bort inte är synliga in någon instans, annars bör man dra om alla kopplingar. Flyttar man en in eller utgång bör man dra om kopplingarna.

2.7.1. Objekt för tidshantering

Ett antal nya objekt för att hantera tider har tillkommit i V4.2.0. Här finns objekt för att lagra, addera, subtrahera tider mm.

Signaler

Signal objekten ATv (AboluteTimeValue) och DTv (DeltaTimeValue) lagrar tidsvärden i form av absoluttid (av typen pwr_tTime, dvs ett datum), resp en deltattid (av typen pwr_tDeltaTime, dvs ett tidsintervall).

Objekten återfinns under signal-mappen i paletten. Någon IO-kopiering av objekten sker ej.

Plc objekt

Addition och subtraktion av tider sker i plc-programmet mha objekten **AtAdd, DtAdd, AtSub, DtSub** och **AtDtSub**.

För att hämta upp en ATv eller DTv används objekten **GetATv** resp **GetDTv**. För att hämta ett attribut av typen pwr_tTime och pwr_tDeltaTime i ett objekt, används objekten **GetATp** resp **GetDTp**

För att lagra ett tidsvärde i en **ATv** eller **DTv** används **StoATv** resp **StoDTv**, eller **CStoATv** resp **CStoDTv** för villkorlig lagring. För att lagra ett tidsvärde i ett attribut av typen pwr_tTime eller pwr_tDeltaTime används objekten **StoATp** resp **StoDTp**, eller **CStoATp** resp **CStoDTp** för villkorlig lagring.

För att konvertera en deltatid till flyttal används DtToA, och vice versa AToDt.

Samtliga objekten återfinns under mappen Signals->Time i plceditorns palett.

2.7.2. Updatering av klasser

Om man gjorde en ändring i en klass var man tidigare tvungen att dumpa databasen till en textfil och sedan ladda in den igen. Nu finns en funktion som konverterar objekten i en databas till den nya klassbeskrivningen utan omladdning. När man startar arbetsbänken undersöks om det finns någon ny version av klassvolymernas dbs-filer. Om så är fallet får man en varning i meddelandefönstret. Man kan då välja att fortsätta med den gamla klassbeskrivningen, eller att uppdatera objekten. Uppdateringen görs med Function->Update Classes i menyn. Vill man först se vilka objekt som kommer att påverkas, kan man exekvera kommandot `wtt> check classes` som ger en list på de klasser som är ändrade och antalet instanser som finns av respektive klass.

Man bör naturligtvis se till att den finns en backup av databasen innan man utför en klass uppdatering.

2.7.3. Ge

Object graph in Window and Folder

It is now possible to display an object graph in a window or folder object. The instance object of the object graph is inserted in the properties Window.Object and Folderx.Object.

Utvalsfärg i Table

En egenskap för att ändra färgen på utvalda celler tabeller har adderats till Table objektet. Sätt den önskade färgen i Table.SelectColor.

Bit typ i Invisible

Eftersom behörighet ofta beskrivs av bitar i en bitmask, kan man nu påverka känslighet och synlighet hos objekt med bitar. Typen för attributet markeras med ##Bit#32[7] vilket innebär en 32-bitars bitmask, bit nummer 7 (första biten är bit 0).

2.7.4. Ändrade typer

pwrb:DataRepEnum

Värdena Int32 och UInt24 har adderats.

2.7.5. Ändrade klasser

Profibus:Pb_Module

Attributet ModuleName har adderats.

BaseComponent:CompLimit

Attributet DisableAlarm har adderats, vilket gör det möjligt att använda gränsvärdesövervakningen i BaseSensor utan larm.

BaseComponent:CompModeDM, CompModeDMFo

Funktionalitet för körning i lokal mode har adderats.

BaseComponent:BaseMValveFo

Funktionalitet för körning i lokal mode har adderats.

RootVolume, SubVolume, SharedVolume

Attributet Modified har adderats. Här lagras tiden när man senast sparade volymen i utvecklingsmiljön.

2.7.6. Nya klasser

CompPID, CompPID_Fo

Pid regulatorn uppdelad i ett huvudobjekt och ett funktionsobjekt. Regulatorn kan då läggas som en komponent i ett annat objekt.

CompModePID, CompModePID_Fo

Mode objektet till ovanstående Pid regulator.

GetDatap

Plc objekt för att hämta upp referensen till ett data objekt, t ex en data utgång i DataArithm. Främst avsedd för att kunna skapa dataingångar i funktionsobjekt med template plckod.

pwrb:ATv

Absolute Time Value, lagring av en absoluttid, pwr_tTime.

pwrb:DTv

Delta Time Value, lagring av en deltatid, pwr_tDeltaTime.

pwrb:AtAdd

Addition av en absoluttid och en deltatid.

pwrb:DtAdd

Addition av två deltatider.

pwrb:AtSub

Subtraktion av en absoluttid från en absoluttid.

pwrb:DtSub

Subtraktion av en deltatid från en deltatid.

pwrb:AtDtSub

Subtraktion av en deltatid från en absoluttid.

pwrb:AtEqual, pwrb:AtGreaterThan, pwrb:AtLessThan

Jämförelse av två absoluttider.

pwrb:DtEqual, pwrb:DtGreaterThan, pwrb:DtLessThan

Jämförelse av två deltatider.

pwrb:CurrentTime

Hämtar systemtiden.

pwrb:DtToA, pwrb:AToDt

Konverterar från detatid till flyttal och vice versa.

pwrb:GetATv

Hämta värdet på en ATv.

pwrb:GetDTv

Hämta värdet på en DTv.

pwrb:StoATv

Lagra ett värde i en ATv.

pwrb:CStoATv

Villkorlig lagring av ett värde i en ATv.

pwrb:StoDTv

Lagra ett värde i en DTv.

pwrb:CStoDTv

Villkorlig lagring av ett värde i en DTv.

pwrb:StoATp

Lagra ett värde i ett absoluttids attribut.

pwrb:CStoATp

Villkorlig lagring av ett värde i ett absoluttids attribut.

pwrb:StoDTp

Lagra ett värde i ett deltatids attribut.

pwrb:CStoDTv

Villkorlig lagring av ett värde i ett deltatids attribut.

ssabox:Ssab_RemoteRack

Konfigurering av PSS9000 remote rack.

ABB_ACC800, ABB_ACC800Fo, ABB_ACC800Sim

Styrning av motoraggregat som använder kran macrot i ASC800.

ABB_ACC_PPO5

Profibus modul till ABB_ACC800.

ABB_ACS_Pb_Slave

Profibus slav till ABB_ACS800.

ABB_Sensor_Pb_PA, ABB_Sensor_Pb_PA_Fo

Basklass för ABB Profibus PA givare.

ABB_TempSensor_TF12, ABB_DiffPressure_265G, ABB_FlowSensor_FXE4000

Diverse ABB Profibus PA givare.

Siemens_ET200S_IM151, Siemens_ET200M_IM153

Profibus slav objekt för ET200S IM151 resp ET200M IM153

Siemens_ET200M_Di32, Siemens_ET200M_Di16, Siemens_ET200S_Di8

Profibus modulobjekt för ET200M digitala ingångsmoduler

Siemens_ET200M_Do32, Siemens_ET200M_Do16, Siemens_ET200S_Do8

Profibus modulobjekt för ET200M digitala utgångsmoduler

Siemens_ET200M_Ai8, Siemens_ET200M_Ai4, Siemens_ET200S_Ai2

Profibus modulobjekt för ET200M analoga ingångsmoduler

Siemens_ET200M_Ao8, Siemens_ET200M_Ao4, Siemens_ET200S_Ao2

Profibus modulobjekt för ET200M analoga utgångsmoduler

Siemens_ET200S_Di4, Siemens_ET200S_Di2

Profibus modulobjekt för ET200S digitala ingångsmoduler

Siemens_ET200S_Do4, Siemens_ET200S_Do2

Profibus modulobjekt för ET200S digitala utgångsmoduler

Siemens_ET200S_Ai2

Profibus modulobjekt för ET200S analoga ingångsmoduler

Siemens_ET200S_Ao2

Profibus modulobjekt för ET200S analoga utgångsmoduler

Kapitel 3. Uppgraderings procedur

3.1. Procedur för uppgradering

Uppgraderingen måste göras från V4.1.3. Om projektet ligger på en tidigare version måste uppgraderingen ske stegvis enligt följande schema **V2.1 -> V2.7b -> V3.0 -> V3.3 -> V3.4b -> V4.0.0 -> V4.1.3 -> V4.2.0**

Uppgraderingen görs i två steg:

- Ta en kopia av projektet
- Exekvera upgrade.sh

3.1.1. Ta en kopia av projektet

Gör `sdf` till projektet och starta administratören.
`> pwra`

Nu öppnas projektlistan. Gå in i editmode, logga in som adminstratör om du saknar behörighet. Leta upp aktuellt projekt, välj `Copy Project` från ProjectReg objektets popupmeny. Öppna kopian och ange ett lämpligt projektnamn och path. Ändra versionen till V4.2.0. Spara därefter och gå ur administratören.

Gör sdf till projectet.

3.1.2. upgrade.sh

upgrade.sh är ett skript som är uppdelat på ett antal olika pass. Efter varje pass får man ange att man vill fortsätta med nästa pass.

Starta scriptet med
`> upgrade.sh`

och kör igenom de olika passen.

dumpdb

Går igenom alla databaser och dumpar varje volym i en egen dumpfil. Dumpfilen för respektive volyme läggs i `$pwrp_db/'volumename'.wb_dmp`

classvolumes

Skapar laddatafiler och structfiler för klassvolymerna.

renamedb

Sparar undan de gamla databaserna med namnet `$pwrp_db/'volumename'.db.1`.

dirvolume

Skapar en directory databas och laddar in dumpfilen för projektvolymen i denna.

loaddb

Skapar databaser och laddar in dumpfilerna i dem.

compile

Kompilerar om samtliga plcprogram.

createload

Skapar laddatafiler för rotvolymer.

createboot

Skapar bootfiler för alla noder i projektet.

Nu återstår att bygga eventuella applikationer. Detta får man göra för hand.

Rensa bort filer från uppgraderingen:

`$pwrp_db/*.wb_dmp.*`

`$pwrp_db/*.db.1` (V4.1 databaser, filkataloger vars innehåll även tas bort)