



## **Release Notes V5.2**

2014 12 02

Copyright SSAB AB 2014

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts.

## Table of Contents

Upgrading to Proview V5.2.0.....	5
New functions.....	5
Inc3p MaxWidup.....	5
PID controller.....	5
Bumblers Manual/Auto switching for P and PD controllers .....	5
Limited widup for parts of output.....	5
MinWidup and MaxWidup.....	5
Event selection on type.....	5
Core plc program.....	7
Profibus FDL.....	7
Default OpPlace attribute.....	7
Start of Xtt without operator window and navigator.....	7
Display alarms in Ge graphs.....	7
Ge dynamics DigTextColor and TimeoutColor.....	8
Graph borders added to Xtt command 'open graph'.....	8
Plc trace, not updated objects are dimmed.....	9
Build configuration.....	9
BuildDirectory.....	9
BuildCopy.....	9
BuildMake.....	9
BuildExec.....	9
Export.....	9
Import.....	9
Build and distribution of configuration files.....	10
Data reference type.....	11
Data reference value.....	11
Plc help texts.....	11
Fault tolerant time functions.....	13
New time plc objects.....	14
New data reference plc objects.....	14
New string plc objects.....	15
New Classes.....	15
AlarmTable.....	15
DataRefv.....	15
GetDataRefv.....	15
StoDataRefv.....	15
CStoDataRefv.....	15
GetDataRefp.....	15
StoDataRefp.....	15
CStoDataRefp.....	15
BuildConfig.....	16
Import.....	16
ApplImport.....	16
Export .....	16
ApplExport.....	16
BuildDirectory.....	16
BuildCopy.....	16
BuildMake.....	16
BuildExecute.....	16
HelpText.....	16

HelpTextL.....	17
AtSel .....	17
AtMux .....	17
AtDemux .....	17
AtMin .....	17
AtMax .....	17
AtLimit .....	17
DtSel .....	17
DtMux .....	17
DtDemux .....	17
DtMin .....	17
DtMax .....	17
DtLimit .....	17
DataSel .....	18
DataMux .....	18
DataEqual .....	18
DataNotEqual .....	18
StrSel .....	18
StrMux .....	18
StrEqual .....	18
StrNotEqual .....	18
StrAdd .....	18
StrTrim .....	18
StrParse .....	18
Pb_FDL_SAP.....	18
Pb_FSL_DataTransfer.....	18
Modified Classes.....	19
Upgrade procedure .....	19

# Upgrading to Proview V5.2.0

This document describes new functions in Proview V5.2.0, and how to upgrade a project from V5.1.0 to V5.2.0.

## New functions

### ***Inc3p MaxWindup***

Inc3p has a new attribute, MaxWindup, to avoid infinite accumulation at unsuitable parameters. If  $\text{MaxWindup} > 0$  the time to action (Acc) is limited to  $\pm \text{MaxWindup}$ .

### ***PID controller***

Functions for bumpless switching and limited windup is added to the PID controller in the PID and CompPID objects.

#### ***Note!***

To get the same PID function as in previous versions, set

```
PDabsFlag = 1
WindupMask = BPID
MaxWindup = same value as MaxOut
MinWindup = same value as MinOut
```

### ***Bumpless Manual/Auto switching for P and PD controllers***

The new attribute PDabsFlag makes bumpless switching from manual to auto possible also for P and PD controllers. Setting PDabsFlag to zero will give the new function where and offset is calculated in manual mode that is added to the output when the controller is switched to auto.

Setting PDabsFlag to 1 will give the old function.

### ***Limited windup for parts of output***

The new attribute WindupMask specifies which parts of the output has limited windup. WindupMask can be set to I, BI, PBI or BPID. BPID will give the old function.

With I or BI, windup for P disturbances are eliminated. This was previously achieved by setting MaxOut to 110%.

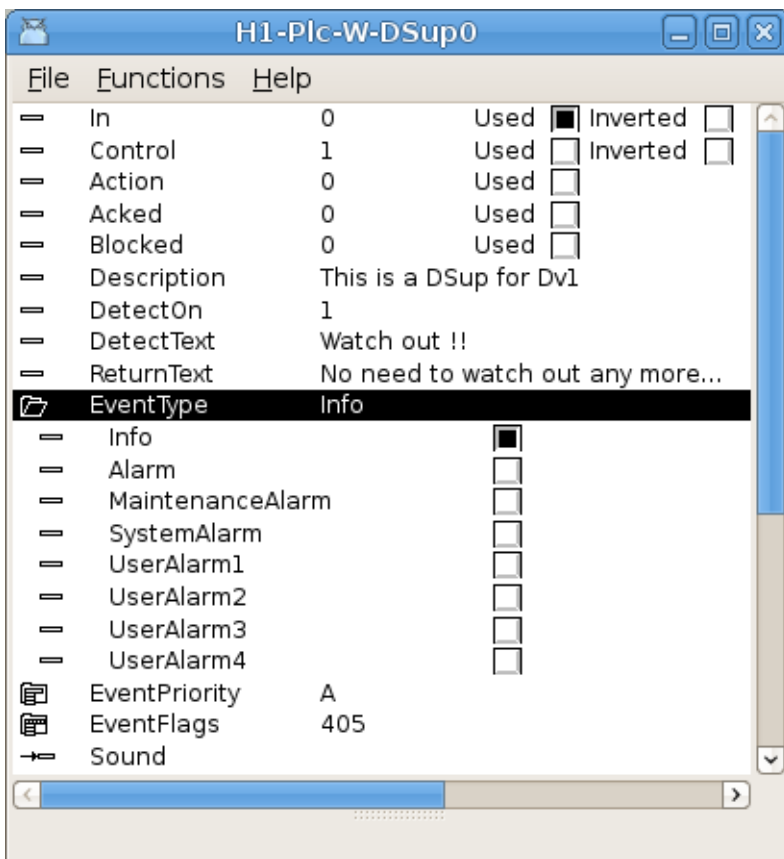
### ***MinWindup and MaxWindup***

The new attributes MinWindup and MaxWindup can be used for example with servo valve control with leakage compensation and limited I part.

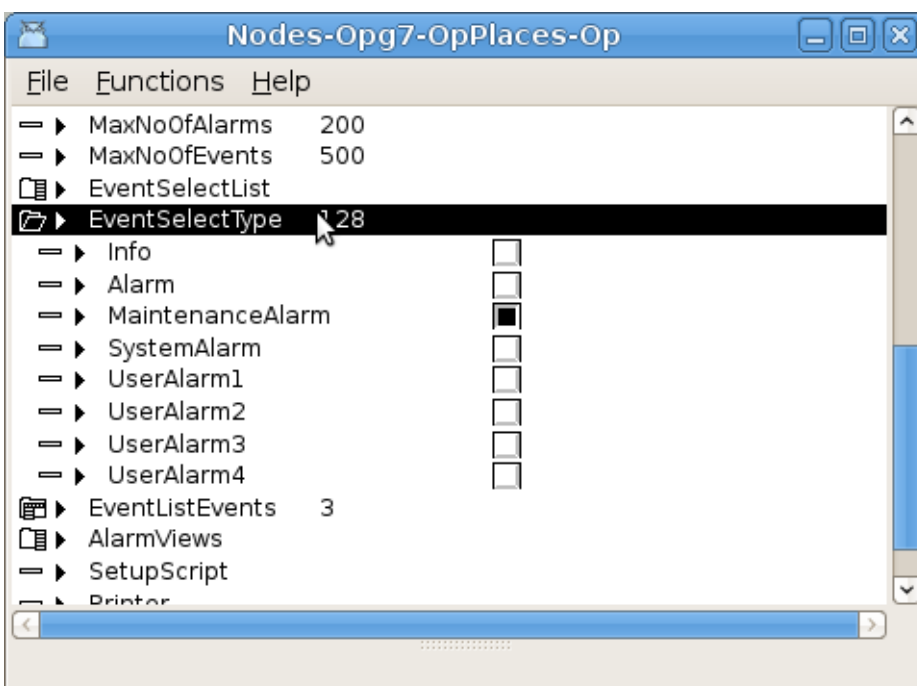
### ***Event selection on type***

Previously the event selection in operator places has been done only on position in the object tree. Now the EventType has been extended with six new types: MaintenanceAlarm, SystemAlarm,

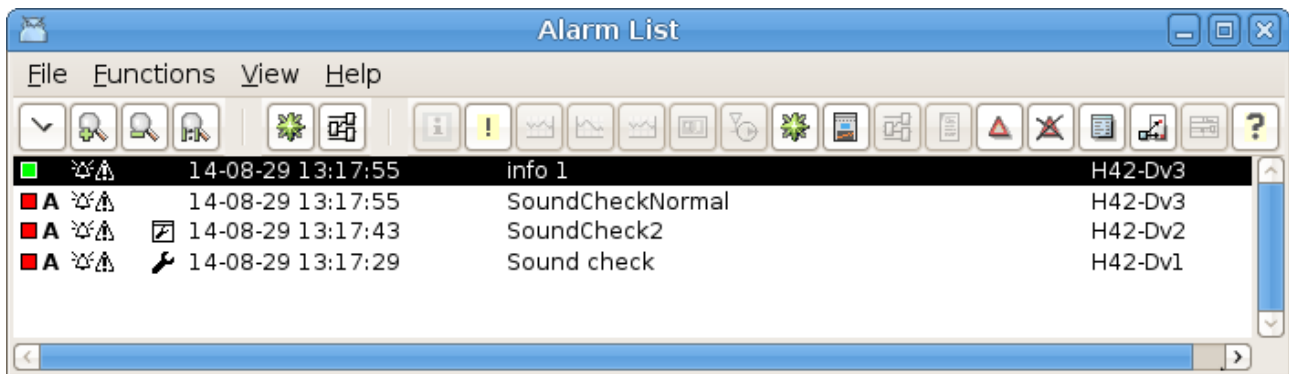
UserAlarm1, UserAlarm2, UserAlarm3 and UserAlarm4, and the event selection can also be made on these types. With the new attribute EventSelectType the types of event that should be displayed is specified. The EventSelectType is present in the OpPlace, AlarmCategory and WebHandler objects.



**Fig Available event types in a DSup object**



**Fig Displayed event types in an OpPlace object**



**Fig System and maintenance types are marked with a wrench**

## ***Core plc program***

Operator stations often doesn't need a plc program, but to get all functionality it has been advised to create a dummy plc. This is no longer necessary as the core plc program will be started if there is no PlcPgm's configured. The core plc program should be configured with a PlcProcess and a PlcThread object.

One advantage with the core plc program is that operator stations can easier be configured for other platforms than the development station. The dummy plc program doesn't need to be built one the development station any more.

## ***Profibus FDL***

The Profibus FDL is a lower level in the Profibus protocol than the previously implemented DP. It can be used to communicate with Siemens S5 and S7 systems. The FDL implementation is an interface to the Sofing Profiboard card. The Softing card is configured, as for DP, with a Pb\_Profiboard object. Below this the FDL service access point and data transfer is configured with Pb\_FDL\_SAP and Pb\_FDL\_DataTransfer objects. See the documentation for these classes for more information.

## ***Default OpPlace attribute***

Previously the default OpPlace object, ie the OpPlace used when no other is specified, was indicated with the name 'OpDefault'. Now an attribute, IsDefaultOp, should be set in the OpPlace object instead, and the OpDefault can be given an arbitrary name.

Note! When upgrading the IsDefaultOp attribute should be set in all OpDefault objects.

## ***Start of Xtt without operator window and navigator***

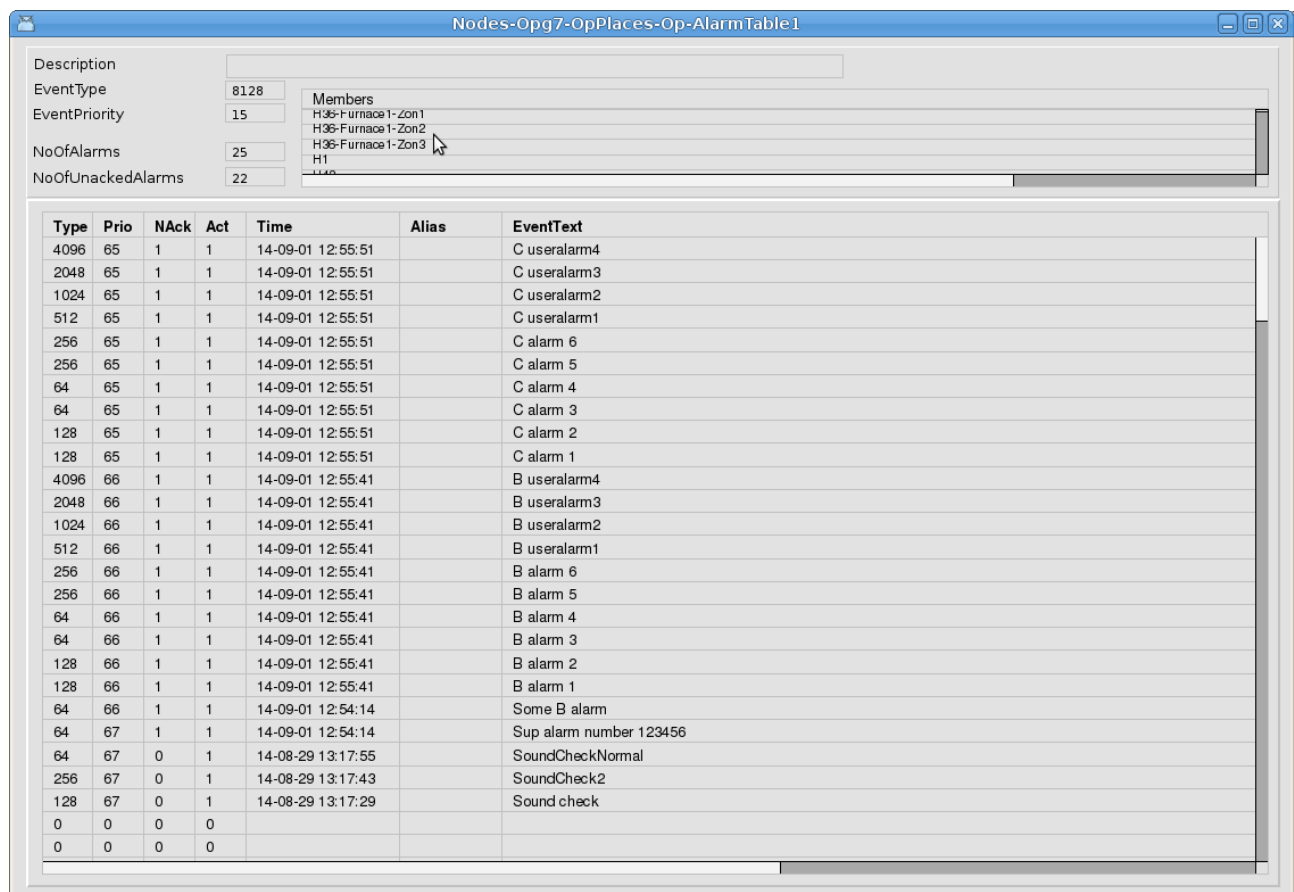
Previously either the operator window or navigator was opened when rt\_xtt was started, and one of them functioned as the main window. Now it's possible to hide both, and the first AutoStart graph will function as main window instead. This makes it possible to design the operator window as a graph in the Ge editor. With the AlarmTable described below, also alarms can be viewed in the graph.

## ***Display alarms in Ge graphs***

With the AlarmTable object it's possible to display alarms in a Ge graph.

The AlarmTable object is placed under an OpPlace object and will gain it's alarm from this OpPlace, ie the event selections for the opplace will also be applied on the AlarmTables. It is possible restrict the alarm selection even further with the Member, EventType and EventPriority

attributes. The alarm information is stored in array attributes which can be displayed in a Ge table.



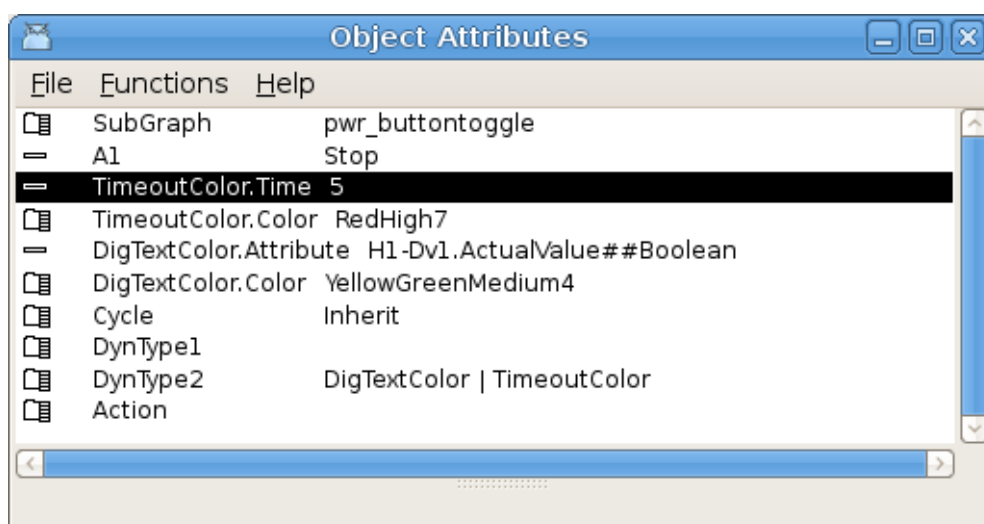
The screenshot shows a window titled "Nodes-OpG7-OpPlaces-Op-AlarmTable1". It contains a form with fields for "Description", "EventType" (8128), "EventPriority" (15), "NoOfAlarms" (25), and "NoOfUnackedAlarms" (22). Below the form is a table with columns: Type, Prio, NAck, Act, Time, Alias, and EventText. The table lists various alarm events with their respective IDs, priorities, acknowledgment status, action status, timestamps, aliases, and descriptions.

Type	Prio	NAck	Act	Time	Alias	EventText
4096	65	1	1	14-09-01 12:55:51		C useralarm4
2048	65	1	1	14-09-01 12:55:51		C useralarm3
1024	65	1	1	14-09-01 12:55:51		C useralarm2
512	65	1	1	14-09-01 12:55:51		C useralarm1
256	65	1	1	14-09-01 12:55:51		C alarm 6
256	65	1	1	14-09-01 12:55:51		C alarm 5
64	65	1	1	14-09-01 12:55:51		C alarm 4
64	65	1	1	14-09-01 12:55:51		C alarm 3
128	65	1	1	14-09-01 12:55:51		C alarm 2
128	65	1	1	14-09-01 12:55:51		C alarm 1
4096	66	1	1	14-09-01 12:55:41		B useralarm4
2048	66	1	1	14-09-01 12:55:41		B useralarm3
1024	66	1	1	14-09-01 12:55:41		B useralarm2
512	66	1	1	14-09-01 12:55:41		B useralarm1
256	66	1	1	14-09-01 12:55:41		B alarm 6
256	66	1	1	14-09-01 12:55:41		B alarm 5
64	66	1	1	14-09-01 12:55:41		B alarm 4
64	66	1	1	14-09-01 12:55:41		B alarm 3
128	66	1	1	14-09-01 12:55:41		B alarm 2
128	66	1	1	14-09-01 12:55:41		B alarm 1
64	66	1	1	14-09-01 12:54:14		Some B alarm
64	67	1	1	14-09-01 12:54:14		Sup alarm number 123456
64	67	0	1	14-08-29 13:17:55		SoundCheckNormal
256	67	0	1	14-08-29 13:17:43		SoundCheck2
128	67	0	1	14-08-29 13:17:29		Sound check
0	0	0	0			
0	0	0	0			

Fig AlarmTable object graph

## Ge dynamics DigTextColor and TimeoutColor

Two new type of dynamics are added in 5.2. DigTextColor changes the color of a text, and TimeoutColor changes the color of an object if the subscription is not updated within a specified time.



The screenshot shows a window titled "Object Attributes" with a menu bar (File, Functions, Help) and a list of attributes. The attribute "TimeoutColor.Time" is highlighted with a value of 5.

Attribute	Value
SubGraph	pwr_buttontoggle
A1	Stop
TimeoutColor.Time	5
TimeoutColor.Color	RedHigh7
DigTextColor.Attribute	H1-Dv1.ActualValue##Boolean
DigTextColor.Color	YellowGreenMedium4
Cycle	Inherit
DynType1	
DynType2	DigTextColor   TimeoutColor
Action	

Fig Dynamic DigTextColor and TimeoutColor

## Graph borders added to Xtt command 'open graph'

It now possible to specify the borders of the graph with the /x0 /y0 /x1 /y1 qualifiers in the comands



'open graph' and 'set subwindow'. The qualifier values will overwrite the values given in Graph attributes for the graph, thus making it possible to display different parts of the graph for different operators or on different nodes. For the 'set subwindow' command this function is implemented for graphs in a multiview cell, not for Ge window objects.

## ***Plc trace, not updated objects are dimmed***

Plc trace are now checking the SubscriptionOldness property and function objects with old subscriptions are dimmed.

## ***Build configuration***

The build configuration makes it possible to handle

- building of applications, with execution of makefiles and copying of include files.
- copying of graph files.
- copying of configuration files from \$pwrp\_cnf.
- importing files from other projects.
- exporting files to other projects.

The build is configured in the directory volume under a BuildConfig object, See Fig 1.

Other objects in the build configuration are

### ***BuildDirectory***

Configures how a directory, eg \$pwrp\_appl or \$pwrp\_pop is built. The actions executed when building the directory is specified with BuildCopy, BuildMake and BuildExecute object.

### ***BuildCopy***

Copies a file, or a number of files specified with wildcard from the source directory to the build tree.

### ***BuildMake***

Executes a make file.

### ***BuildExec***

Executes a shell command.

### ***Export***

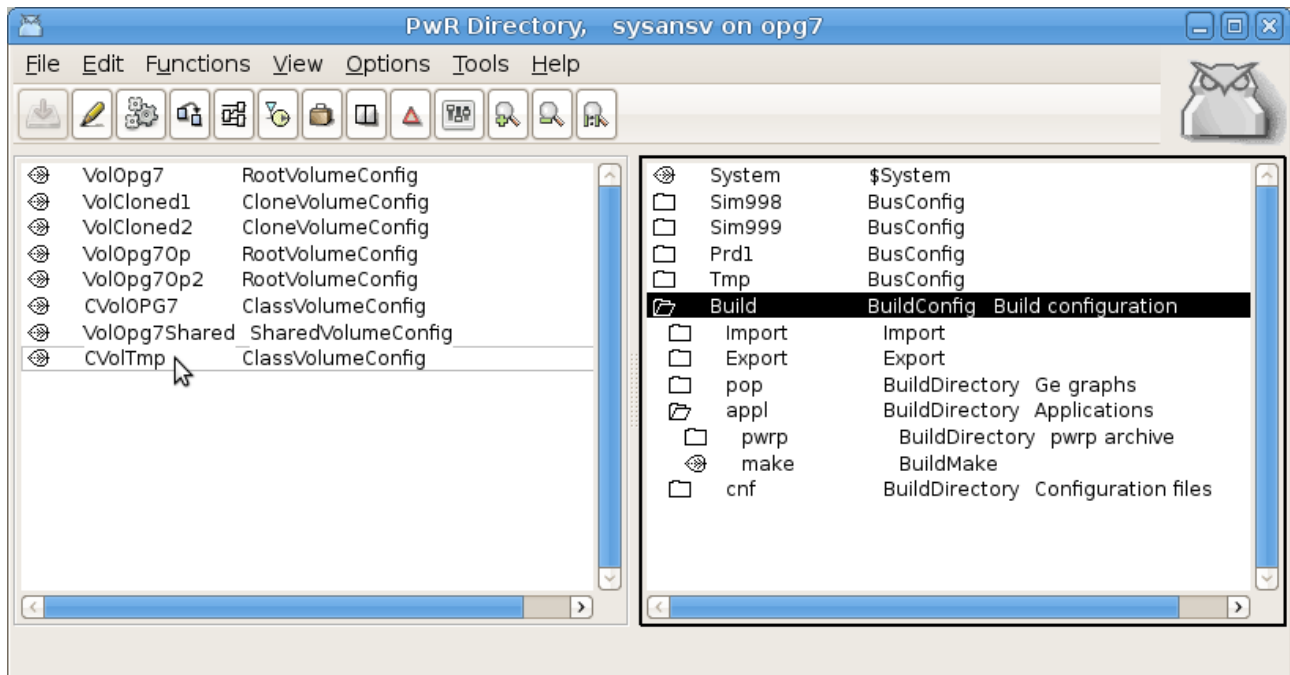
Configures files that should be exported to other projects. Normally files are exported to a common directory from where they are imported by other projects. Specific files to export is configured with ApplExport objects.

### ***Import***

Configures files that is imported from other projects. Specific files are imported with ApplImport objects.

The build, export and import can be performed automatically when building a node, by configuring BuildBeforeNode or BuildAfterNode in the Option attribute of the BuildDirectory, Import and Export objects.

To perform an selective build, export or import, the Build Directories, Import and Export window is opened from the Functions menu in the configurator.



**Fig 1 The build configuration**

## ***Build and distribution of configuration files***

The build and distribution of configuration files is more strict regarding the separation in source and build tree. Now all configuration files reside in \$pwrp\_cnf, and in \$pwrp\_cnf there can be one subdirectory per node. Under each node there can be one subdirectory for each linux user on the node. No files should be distributed directly from \$pwrp\_cnf, they should first be copied to \$pwrp\_load in the build tree. When the node is built, a corresponding node and user structure is created under \$pwrp\_load, and files are copied from \$pwrp\_cnf to \$pwrp\_load. The files handled by the build methods are

\$pwrp_cnf/xtt_help.dat	→ \$pwrp_load/
\$pwrp_cnf/'node'/xtt_help.dat	→ \$pwrp_load/'node'/
\$pwrp_cnf/'node'/'user'/xtt_help.dat	→ \$pwrp_load/'node'/'user'/
\$pwrp_cnf/xtt_setup.rtt_com	→ \$pwrp_load/
\$pwrp_cnf/'node'/xtt_setup.rtt_com	→ \$pwrp_load/'node'/
\$pwrp_cnf/'node'/'user'/xtt_setup.rtt_com	→ \$pwrp_load/'node'/'user'/
\$pwrp_cnf/Rt_xtt	→ \$pwrp_load/
\$pwrp_cnf/'node'/Rt_xtt	→ \$pwrp_load/'node'/
\$pwrp_cnf/'node'/'user'/Rt_xtt	→ \$pwrp_load/'node'/'user'/
\$pwrp_cnf/pwrp_stop.sh	→ \$pwrp_load/
\$pwrp_cnf/'node'/pwrp_stop.sh	→ \$pwrp_load/'node'/
\$pwrp_cnf/pwrp_alias.dat	→ \$pwrp_load/

The copying is made when the node is built, but can also be made separately with the configurator command 'build cnf /node='. All copying can also be disabled from the configurator options dialog, in case one prefers to this with make files or scripts.

## ***Data reference type***

The type `pwr:Type-$DataRef` is used for data references in all objects handling data references, eg `NmpsCell`, `DataArithm`, `DataCollect` etc. The type contains a pointer and an attribute reference and the c declaration in `pwr.h` is

```
typedef struct {  
    pwr_tVoid      *Ptr pwr_dAlignLW;    //!< Private plc pointer to data object.  
    pwr_tAttrRef   Aref pwr_dAlignLW;    //!< Attribute reference to data object.  
} pwr_tDataRef;
```

In `NmpsCell` objects, the previous pointer and object id, for example `Data1P`, and `Data1ObjId`, is replaced by a `DataRef` named `Data1`. If these are used in any c-code, they should be replaced by the corresponding `DataRef` elements, eg

```
Data1P      → Data1P.Ptr  
Data1_Objid → Data1P.Aref.Objid
```

Plc objects to fetch and store an attribute of type `DataRef` are `GetDataRefp`, `StoDataRefp` and `CstoDataRefp`. Also a signal object that stores a data reference is added, see below.

## ***Data reference value***

A new signal object to store a data reference is added, `DataRefv`. The data reference can be fetched for example with a `GetData` or from a data output of a `NmpsCell`.

Plc objects to fetch and store a `DataRefv` are `GetDataRefv`, `StoDataRefv` and `CStoDataRefv`.

If the `DataRefv` references a dynamic object, the pointer is invalid if the object is deleted. The user is responsible to assure that the pointer is not used if the object is deleted.

## ***Plc help texts***

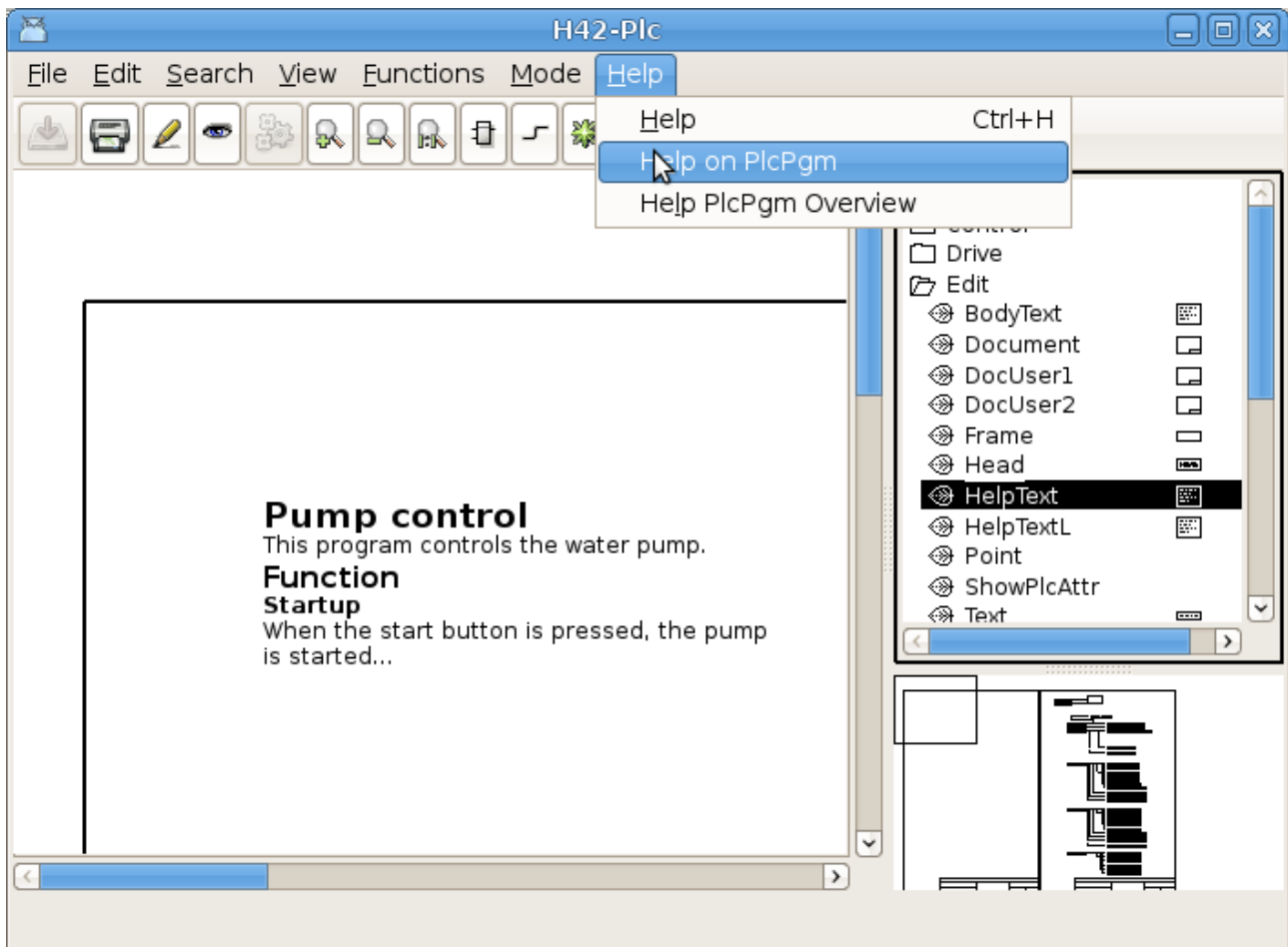
The plc objects `HelpText` and `HelpTextL` are added to view and generate help texts from plc documents. The help texts inserted in these objects are viewed in the plc document, and can also be viewed in the help browser.

The text can contain tags to format the text and to insert images and links. Images will not be displayed in the plc document, they will only be shown in the help browser. Neither will links work in the plc document. The supported tags are

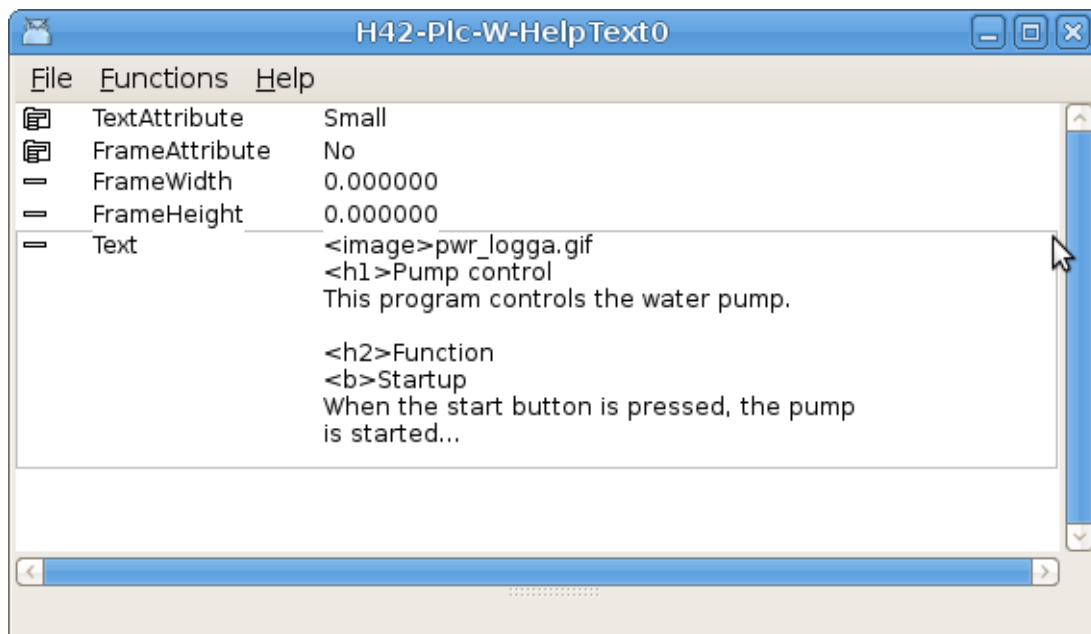
```
<h1>      Large header  
<h2>      Header  
<image>   Display an image.  
<link>    Link to another help text topic
```

The maximum text size is for the `HelpText` object 1023 characters, and for `HelpTextL` 8191 characters.

The help texts will be concatenated to one help text file for the volume, when the volume is built. The file will contain one topic for each plc window, that contains the text from all `HelpText` objects in the window. The texts are displayed in the order the `HelpText` objects are found in the database.

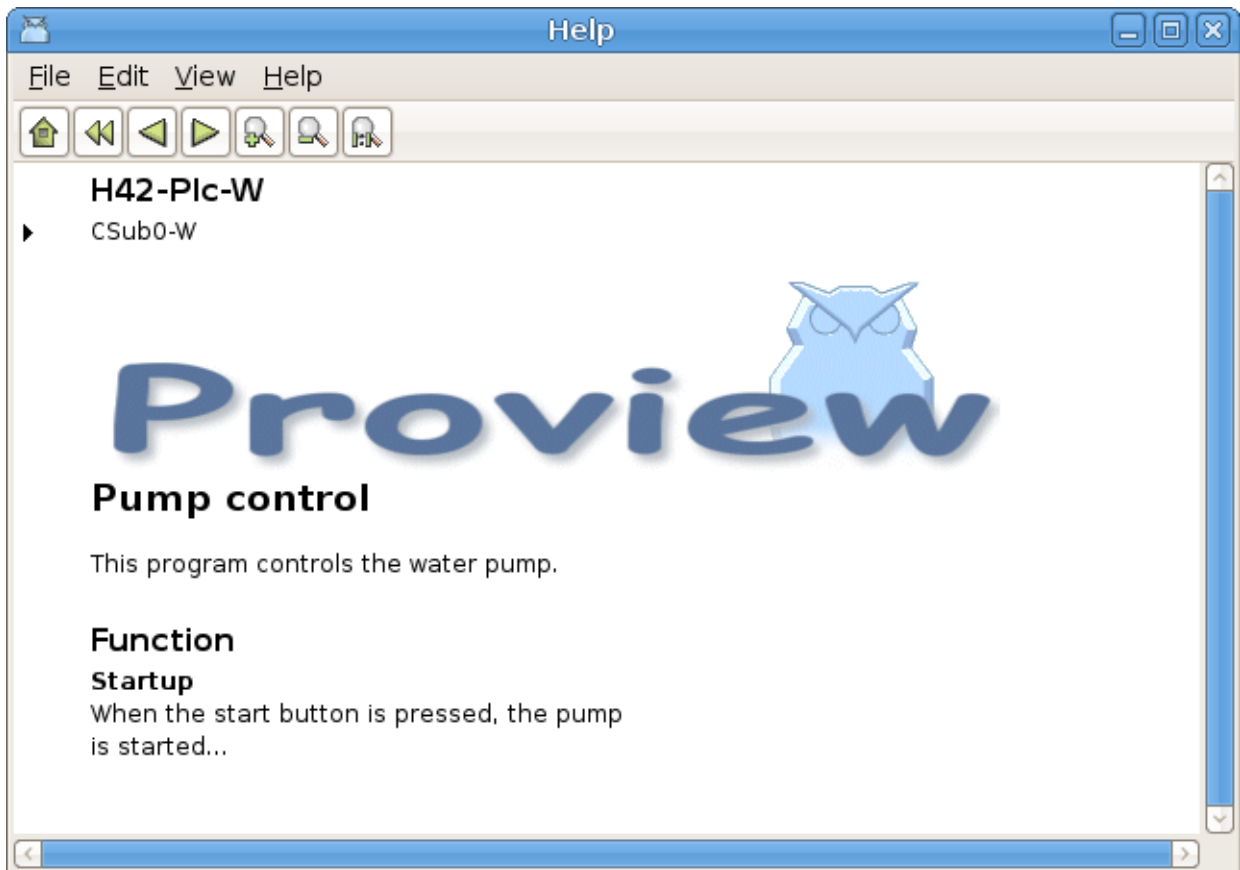


**Fig Help text example**



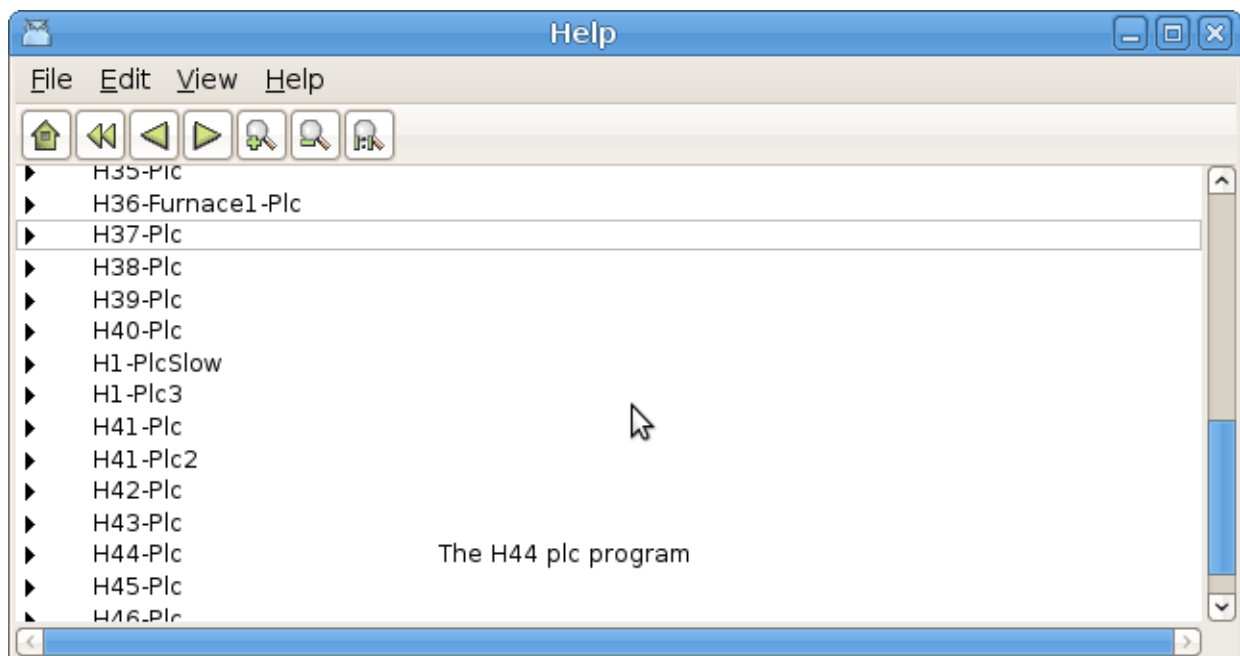
**Fig Edited text**

The help texts are viewed in the help browser, by activating help in the popup menu for the PlcPgm object, or from 'Help on PlcPgm' in the plc editor menu. Also in runtime the text can be displayed in the same way. The text in the help browser first shows a list of all subwindows with links to the help texts of the subwindows.



**Fig Text in help browser**

From 'Help PlcPgm Overview' in the plc editor menu, a list of all the PlcPgm's in the volume is displayed with description and links to the help texts of each PlcPgm.



**Fig Help PlcPgm overview**

## ***Fault tolerant time functions***

The previous time functions in Proview terminates the program if an invalid time is detected. A new set of functions that continues the execution is added to the time archive. Instead of terminating, the constants pwr\_cNotATime or pwr\_cNotADeltaTime is returned when an erroneous time is detected.

The new functions are

```
time_Aadd_NE()  
time_Acomp_NE()  
time_Adiff_NE()  
time_Asub_NE()  
time_Dabs_NE()  
time_Dadd_NE()  
time_Dneg_NE()  
time_Dsub_NE()  
time_Dcomp_NE()
```

The function is equivalent to the previous functions without the \_NE suffix, except for the handling of invalid times.

Also time\_DTtoFloat() and time\_DTtoFloat64 are modified and returns NAN for invalid deltetime input, and time\_FloatToD() and time\_Float64ToD() returns pwr\_cNotADeltaTime for invalid float input.

The new fault tolerant functions are inserted in all PLC objects handling absolute and delta times. The constants pwr\_cNotATime and pwr\_cNotADeltaTime are displayed in xtt as 'NotATime' and 'NotADeltaTime'.

## ***New time plc objects***

A number of new time plc objects are added.

AtSel	Select one of two absolute times.
AtMux	Absolute time multiplexer.
AtDemux	Absolute time demultiplexer.
AtMin	Absolute time minimum function.
AtMax	Absolute time maximum function.
AtLimit	Absolute time limiter.
DtSel	Select one of two delta times.
DtMux	Delta time multiplexer.
DtDemux	Delta time demultiplexer.
DtMin	Delta time minimum function.
DtMax	Delta time maximum function.
DtLimit	Delta time delimiter.

## ***New data reference plc objects***

A number of new data reference plc objects are added.

DataSel	Select one of two data references.
DataMux	Data reference multiplexer.
DataEqual	Check if two data references are equal.
DataNotEqual	Check if two data references are not equal.

## ***New string plc objects***

A number of new string plc objects are added.

StrSel	Select one of two strings.
StrMux	String multiplexer.
StrEqual	Check if two strings are equal.
StrNotEqual	Check if two strings are not equal.
StrAdd	Add a number of strings.
StrTrim	Remove leading and trailing spaces in a string.
StrParse	Parse a string.

## **New Classes**

### **AlarmTable**

Object placed under an OpPlace object that contains a table with alarms or events

### **DataRefv**

Signal object containing a data referens value of type DataRef. It contains a pointer to the data object, and the attribute referens of the data object.

### **GetDataRefv**

Plc object to fetch a DataRefv.

### **StoDataRefv**

Plc object to store a DataRefv.

### **CStoDataRefv**

Plc object for conditional storage of a DataRefv.

### **GetDataRefp**

Plc object to fetch the value of a DataRef attribute.

### **StoDataRefp**

Plc object to store a data reference to a DataRef attribute.

### **CstoDataRefp**

Conditional storage of a data referens to a DataRef attribute.

## **BuildConfig**

The BuildConfig is the root object for the build configuration. This object is placed in the directory volume as toplevel object in the node view.

The build configuration makes it possible to handle

- building of applications, with execution of makefiles and copying of include files.
- copying of graph files.
- copying of configuration files from \$pwrp\_cnf.
- importing files from other projects.
- exporting files to other projects.

## **Import**

Configuration of imported files from other projects or external modules.

## **ApplImport**

Configuration of specific files to import.

## **Export**

Configuration of files to export to other projects.

## **ApplExport**

Configuration of specific files to export.

## **BuildDirectory**

Configures how a directory, eg \$pwrp\_appl or \$pwrp\_pop is built. The actions executed when building the directory is specified with BuildCopy, BuildMake and BuildExecute objects.

## **BuildCopy**

Copies a file, or a number of files specified with wildcard from the source directory to the build tree.

## **BuildMake**

Executes a make file.

## **BuildExecute**

Executes a shell command.

## **HelpText**

Plc object for documentation in plc sheet and help text browser.



## **HelpTextL**

Same as HelpText with larger text buffer (8191 characters).

## **AtSel**

Plc object. Select one of two absolute times.

## **AtMux**

Plc object. Absolute time multiplexer.

## **AtDemux**

Plc object. Absolute time demultiplexer.

## **AtMin**

Plc object. Absolute time minimum function.

## **AtMax**

Plc object. Absolute time maximum function.

## **AtLimit**

Plc object. Absolute time limiter.

## **DtSel**

Plc object. Select one of two delta times.

## **DtMux**

Plc object. Delta time multiplexer.

## **DtDemux**

Plc object. Delta time demultiplexer.

## **DtMin**

Plc object. Delta time minimum function.

## **DtMax**

Plc object. Delta time maximum function.

## **DtLimit**

Plc object. Delta time delimiter.

## **DataSel**

Plc object. Select one of two data references.

## **DataMux**

Plc object. Data reference multiplexer.

## **DataEqual**

Plc object. Check if two data references are equal.

## **DataNotEqual**

Plc object. Check if two data references are not equal.

## **StrSel**

Plc object. Select one of two strings.

## **StrMux**

Plc object. String multiplexer.

## **StrEqual**

Plc object. Check if two strings are equal.

## **StrNotEqual**

Plc object. Check if two strings are not equal.

## **StrAdd**

Plc object. Add a number of strings.

## **StrTrim**

Plc object. Remove leading and trailing spaces in a string.

## **StrParse**

Plc object. Parse a string.

## **Pb\_FDL\_SAP**

IO Rack object configuring a FDL service access point for Softing Profiboard card.

## **Pb\_FSL\_DataTransfer**

IO Card object configuring a FDL data transfer.

# Modified Classes

## Upgrade procedure

The upgrading has to be done from any version in the interval V4.7.0. If the project has a lower version, the upgrade has to be performed stepwise following the schema

V2.1 -> V2.7b -> V3.3 -> V3.4b -> V4.0.0 -> V4.1.3 -> V4.2.0 -> V4.5.0 -> V4.6.0 -> V4.7.0 -> V4.8.6 -> (V5.0.0) -> V5.1.0 -> V5.2

Enter the administrator and change the version of the project to V5.2.0. Save and close the administrator.

Enter the directory volume and save.

If you have any class volumes, enter the class editor and build the volume.

Enter the configurator for each root volume and activate 'Function/Update Classes' and build.

Note ! The update procedure upgrade.sh doesn't have to be executed.

Modify OpPlace, PID and CompPID objects

- set the IsDefaultOp attribute in the OpDefault OpPlace object to 1.
- set PDABsFlag = 1, WindupMask = BPID, MaxWindup = same value as MaxOut and MinWindup = same value as MinOut in PID and CompPID objects.