Release Notes V5.2

2014 12 02

Table of Contents

Upgrading to Proview V5.2.0

This document describes new functions i 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 unsutible parameters. If MaxWindup > 0 the time to action (Acc) is limited to +- 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 = save value as MinOut
```

Bumbless 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 disturbancies 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 shoud be displayed is specified. The EvenSelectType is present the the OpPlace, AlarmCategory and WebHandler objects.

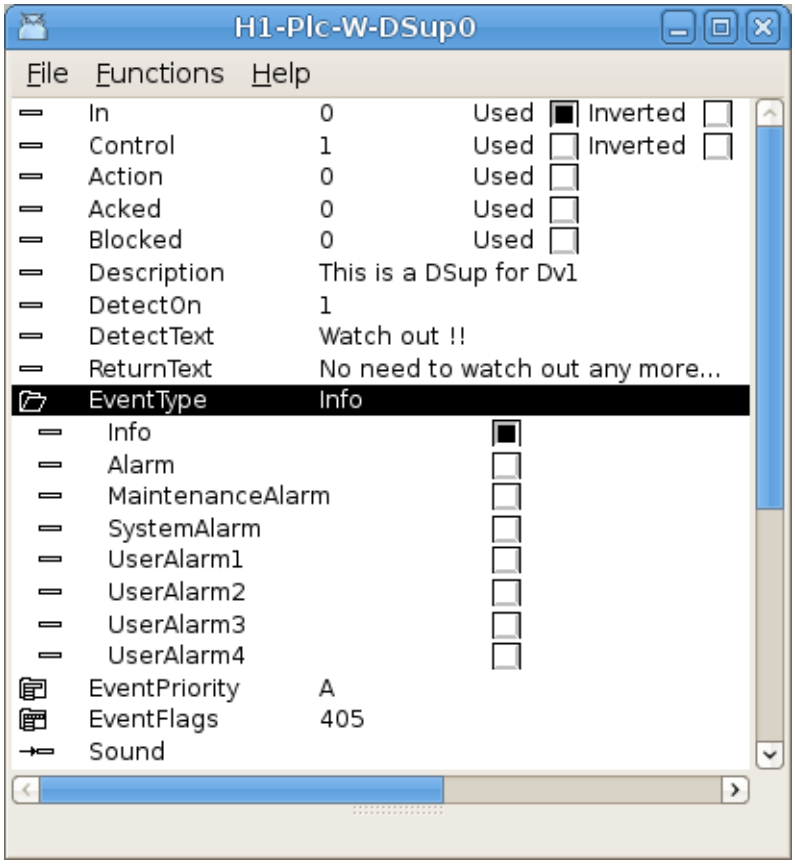


Fig Available event types in a DSup object

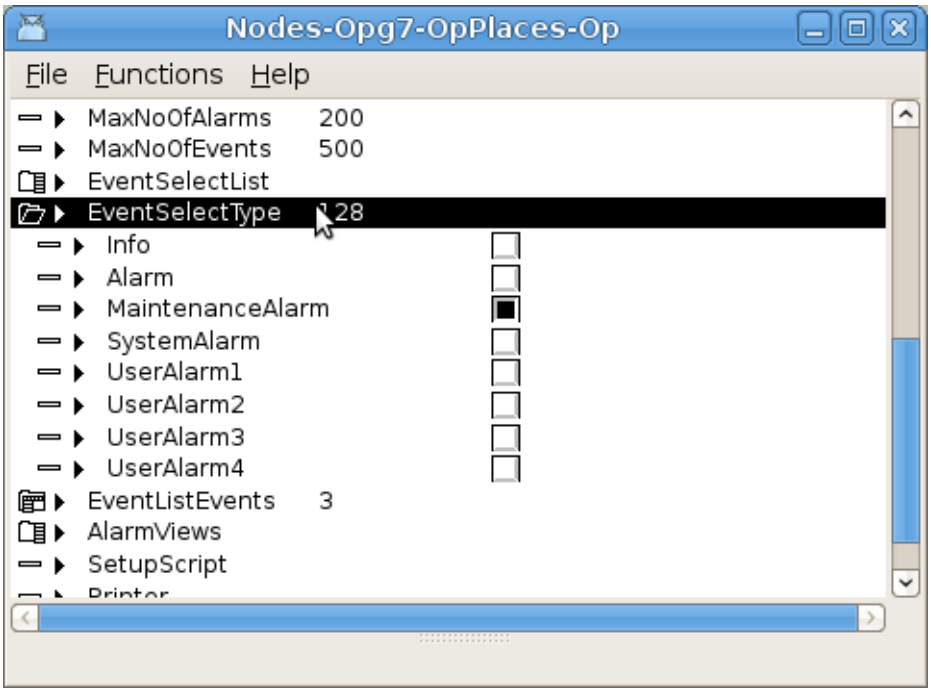


Fig Displayed event types in an OpPlace object

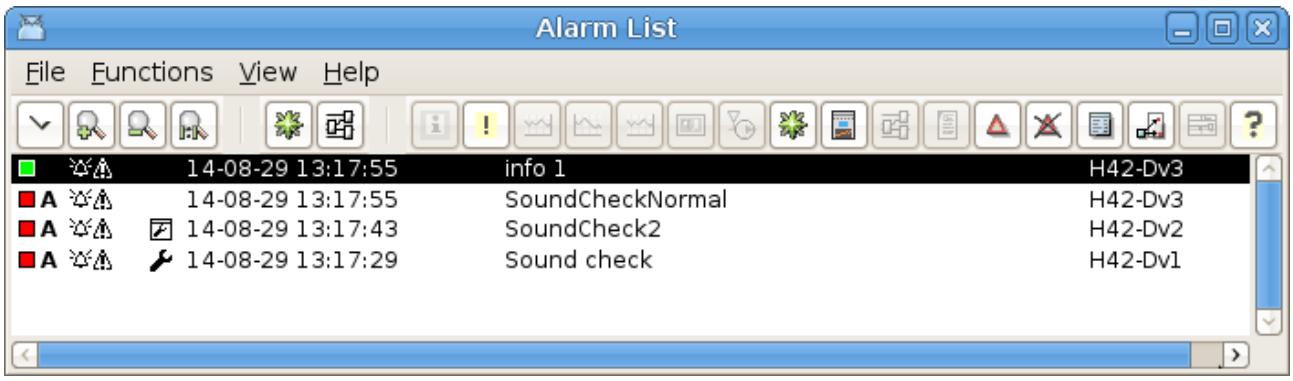


Fig System and maintenance types are marked with a wrench

Core plc program

Operator stations often doesn't need a plc program, but to get all functionallity it has been adviced to create a dummy plc. This is no longer necessary as the core plc program wil 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 progam 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

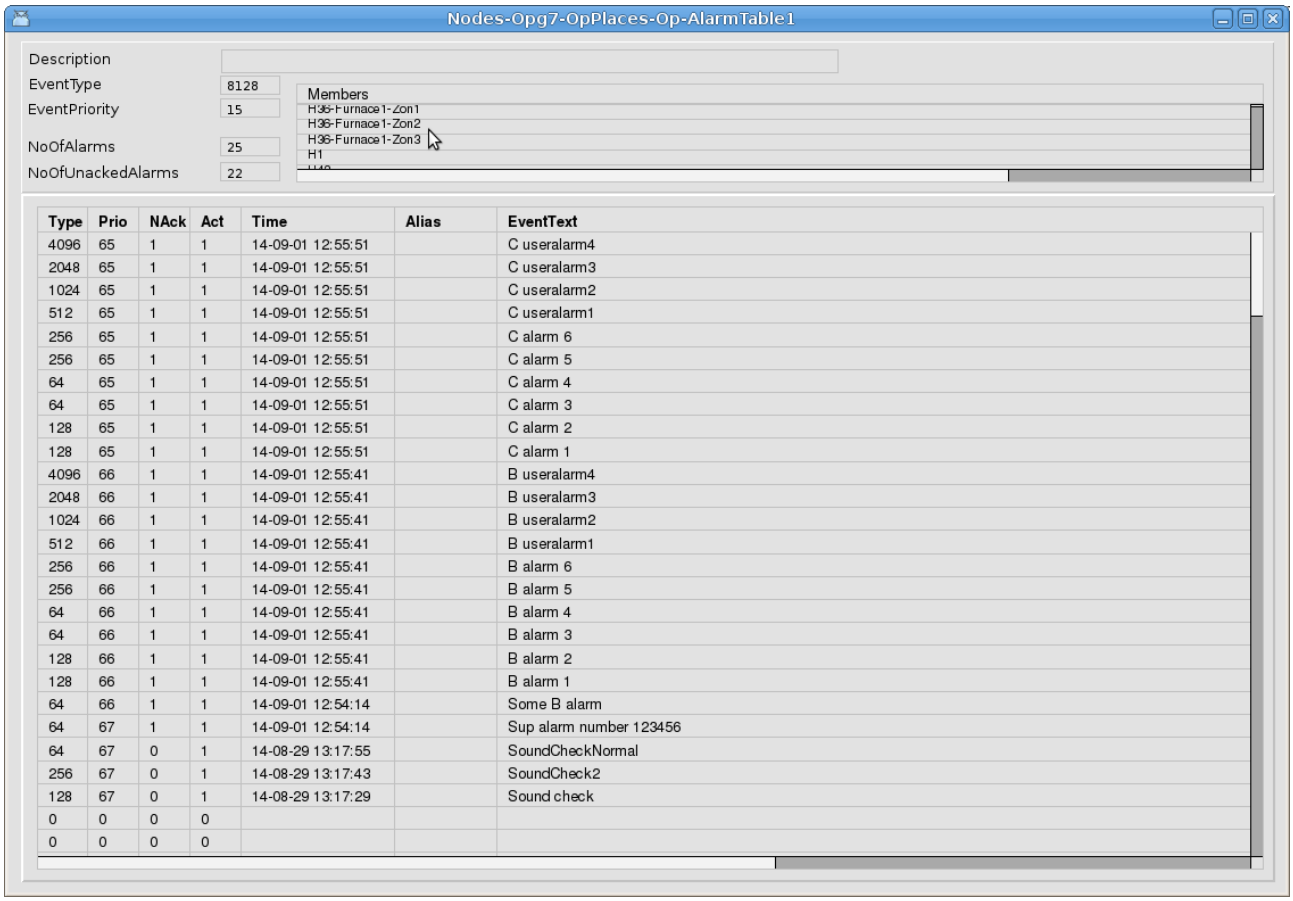
Previously either the operator window or navigator was opened when rt_xtt was started, and one of them functioned as the main window. No it's possible to hide both, and the first AutoStart graph will function as main window instead. This makes it possible to design the operator window as a graph in the 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.



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.

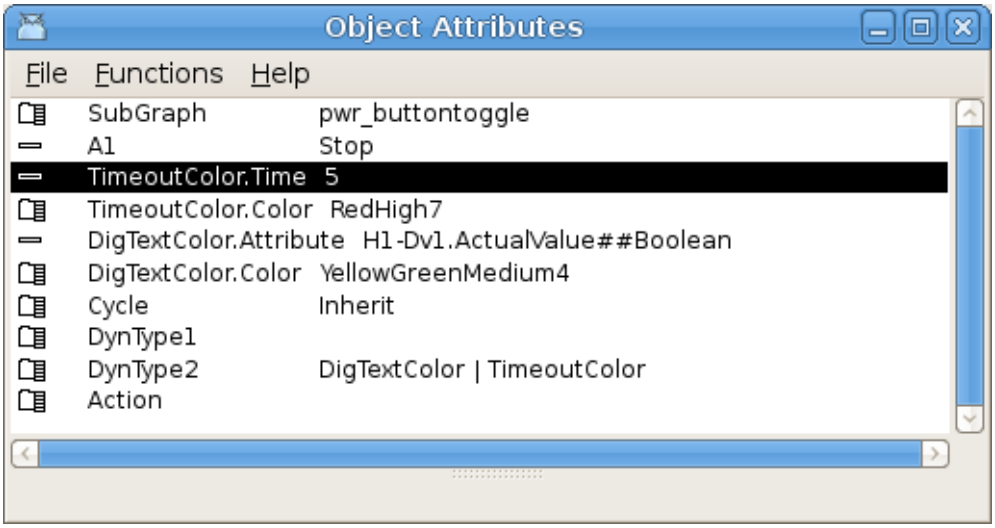


Fig Dynamic DigTextColor and TimeoutColor

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.

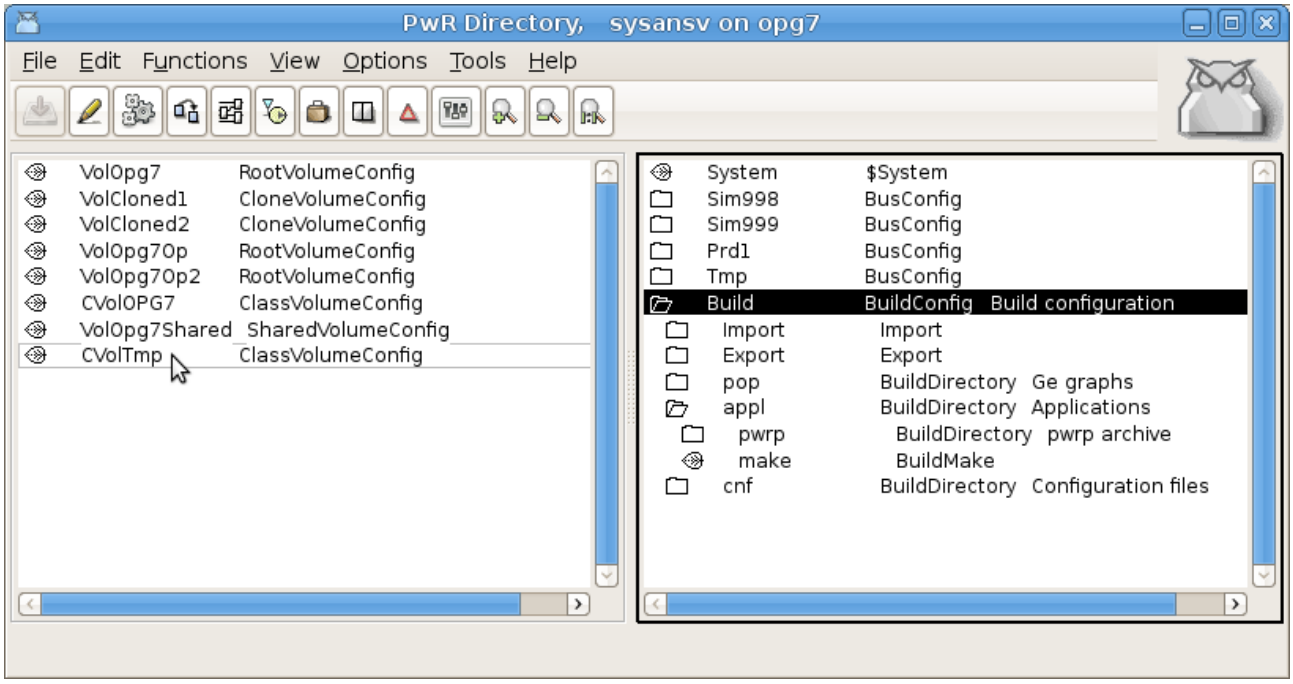


Fig 1 The build configuration

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 separateley with the configurator command 'build cnf /node='. All copying can also be disabled from the configurator options dialog, in case one preferes to this with make files or scripts.

Data reference type

The type pwrs: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 Data1P. 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 DataRev 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.

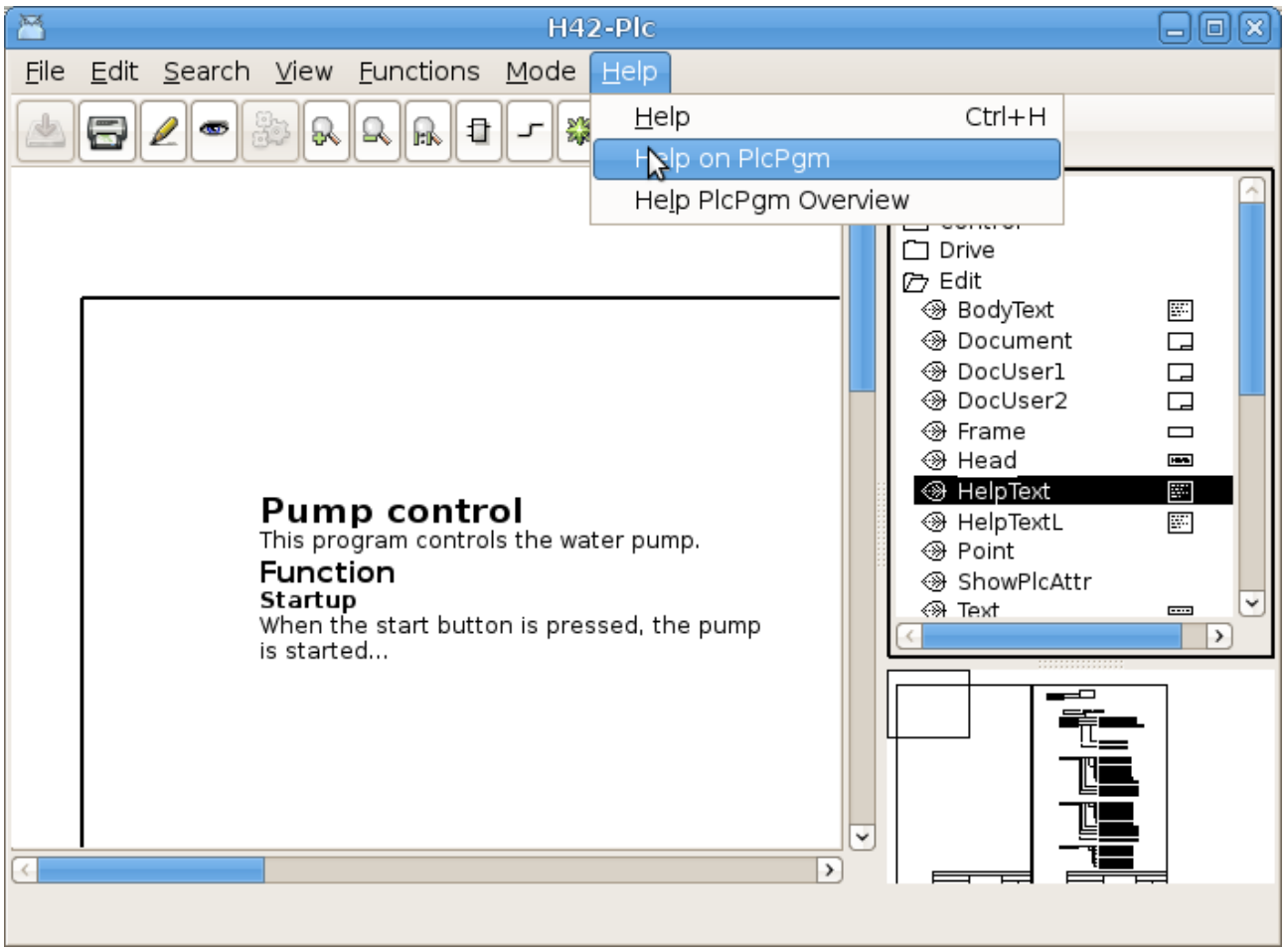


Fig Help text example

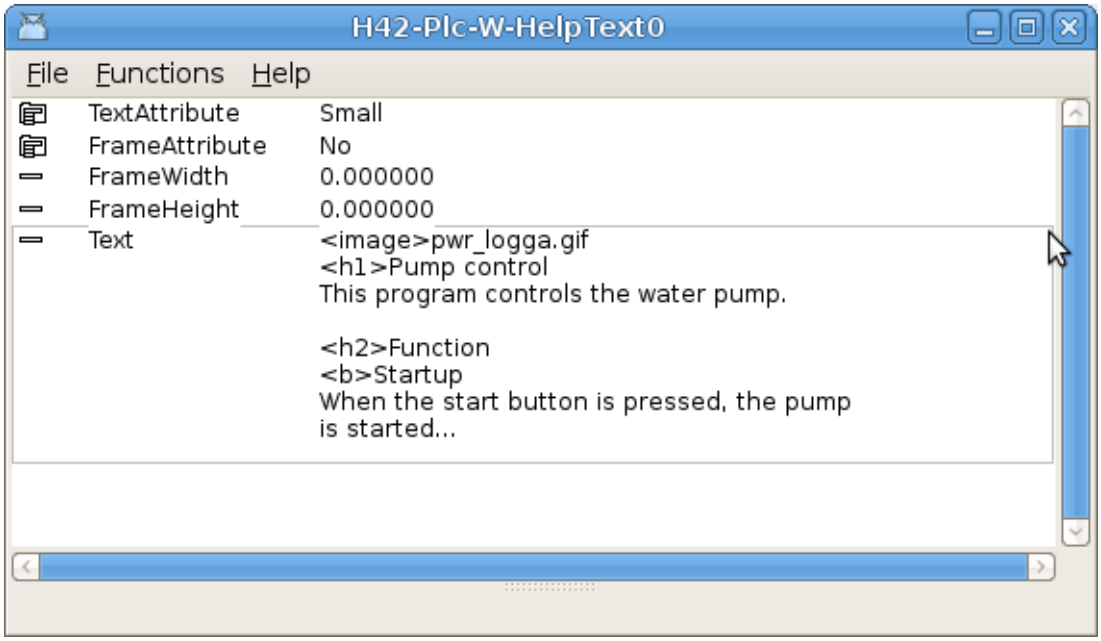


Fig Edited text

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.

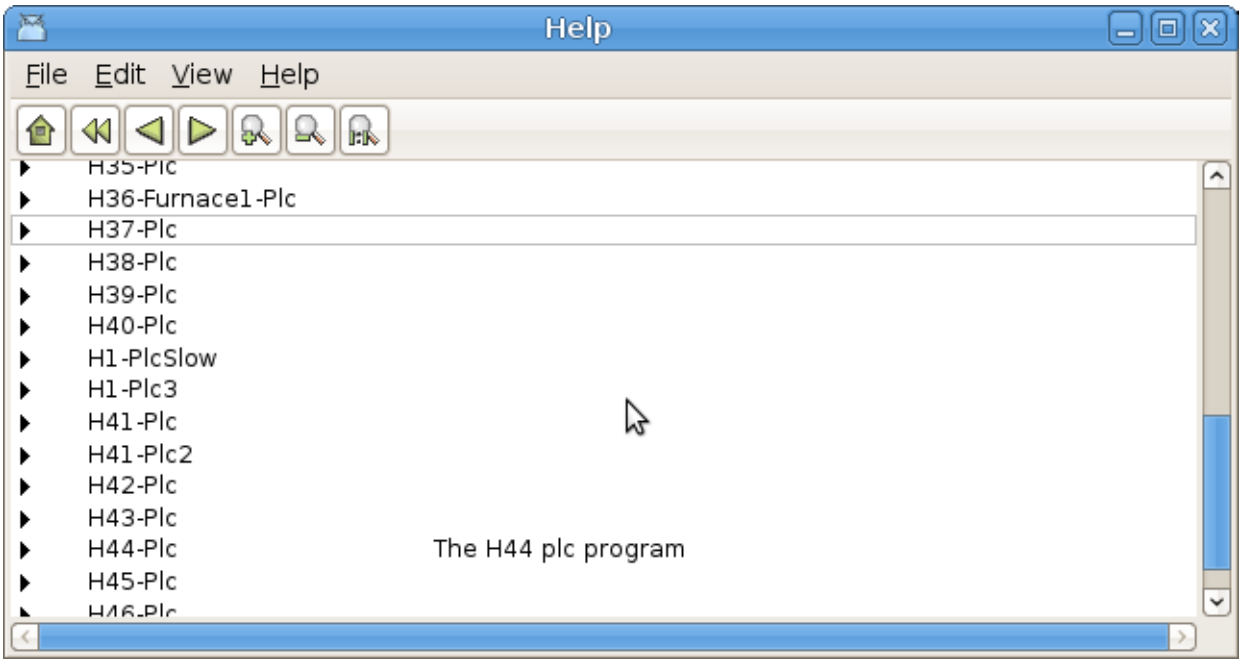


Fig Help PlcPgm overview

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_DToFloat() and time_DToFloat64 are modified and returns NAN for invalid deltatime 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 constans 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 of 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 contaning 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 of two data references are equal.

DataNotEqual

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

S trSel

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 obeject 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 a V5.1 version. 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

Projects not containg NmpsCell objects

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.

I 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 for projects without NmpsCells.

Modify OpPlace, PID and CompPID objects

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

Projects contaning NMpsCell objects

If the previous version should be kept, first make a copy of the project.

Make a copy of the project

Do `sdf` to the project and start the administrator

```
> pwra
```

Now the Projectlist is opened. Enter edit mode, login as administrator if you lack access. Find the current project and select Copy Project from the popup menu of the ProjectReg object. Open the copy and assign a suitable project name and path. Save and close the administrator.

Dump the databases

Execute the first pass, *dumpdb*, in the script *`reload.sh`*.

```
> reload.sh
```

```
  reload.sh   Dump and reload of database.

  Arguments   Database or databases to reload.
              I no arguments is supplied, all databases will be
             reloaded.

  Pass

    dumpdb       Dump database to textfile $pwrp_db/'volume'.wb_dmp
    classvolumes Create structfiles and loadfiles for classvolumes
    renamedb     Rename the old database
    loaddb       Load the dump into the new database
    compile      Compile all plcprograms in the database
    createload   Create new loadfiles.
    createboot   Create bootfiles for all nodes in the project.


-- Reloading volume  directory volopg2

Pass: dumpdb classvolumes renamedbloaddb compile createload createboot

Enter start pass [dumpdb] >

------------------------------------------------------------------------
 Pass dump database
------------------------------------------------------------------------
Do you want to continue ? [y/n/go] y
ls: cannot access /data0/pwrp/opg2/common/db/*.wb_dmp: No such file or
directory
Dumping volume directory in /data0/pwrp/opg2/common/db/directory.wb_dmp
...
I Database opened /data0/pwrp/opg2/common/db/volopg2.db
ls: cannot access /data0/pwrp/opg2/common/db/*.wb_load: No such file or
directory



------------------------------------------------------------------------
 Pass create structfiles and loadfiles for classvolumes
------------------------------------------------------------------------
Do you want to continue ? [y/n/go] n
setdb is obsolete
>
```

Check that the one dumpfile is created for every rootvolume

```
> cd $pwrp_db
> ls -l *.wb_dmp
-rw-rw-r-- 1 cs pwrp 7467 2010-03-26 16:32 volopg2.wb_dmp
```

Linux release upgrade

If you are using an older Ubuntu version to upgrade the linux release and install the pwr52 package.

Change version

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

upgrade.sh

Do `sdf` to the project.

upgrade.sh is a script that is divided into a number of passes. After each pass you you have to answere whether to continue with the next pass or not.
Start the script with

```
> upgrade.sh
```

Start from the classvolumes pass.

```
Enter start pass [classvolumes] >
```

classvolumes

Create loadfiles and structfiles for the class volumes.

renamedb

Store the old databases under the name `$pwrp_db/'volumename'.db.1.`

loaddb

Create databases and load the dumpfiles into them.

compile

Compile all the plc programs.

createload

Create loadfiles for the root volumes.

createboot

Create bootfiles for all nodes in the project.

If the project contains any application programs, these has to be built manually.

Delete files from the upgrading procedure:

`$pwrp_db/*.wb_dmp.*`
`$pwrp_db/*.db.1` (old databases, directories which content also should be removed)

Other modifications

Modify OpPlace, PID and CompPID objects

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

Replace GetDatap objects with GetDataRefp. Concatenate the separately specified object and attribute in the GetDatap to one signle attribute specification in the GetDataRefp.

```
Replace pointers to NmpsCell objects in DataArithm code:

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

List example

```
>
> sdf opg2
Setting base /data0/x5-3-1/rls
>
> upgrade.sh

  upgrade.sh  Upgrade from V5.1.0 to V5.2.0


  Pass

    classvolumes   Create loadfiles for classvolumes.
    renamedb       Rename old databases.
    loaddb         Load dumpfiles.
    compile        Compile all plcprograms in the database
    createload     Create new loadfiles.
    createboot     Create bootfiles for all nodes in the project.


-- Upgrade opg2

Enter start pass [classvolumes] >

------------------------------------------------------------------------
Pass create structfiles and loadfiles for classvolumes
------------------------------------------------------------------------
Do you want to continue ? [y/n/go] y
ls: cannot access /data0/pwrp/opg2/src/db/*.wb_load: No such file or
directory

------------------------------------------------------------------------
Pass rename old databases
------------------------------------------------------------------------
Do you want to continue ? [y/n/go] y
-- Saving file /data0/pwrp/opg2/src/db/volopg.db ->
/data0/pwrp/opg2/src/db/volopg.db.1

------------------------------------------------------------------------
Pass load database
------------------------------------------------------------------------
Do you want to continue ? [y/n/go] y
-- Loading volume volopg
...
-- Processing line: 57
-- Building volume directory
I Volume directory loaded
I Database opened /data0/pwrp/opg2/src/db/directory.wb_load
```

```
-- Processing line: 200
-- Building volume VolOpg
I Volume VolOpg loaded
Berkeley DB 4.6.21: (September 27, 2007)
info put: 0
Berkeley DB 4.6.21: (September 27, 2007)
info get: 0
int rc =  m_txn->abort(): 0

------------------------------------------------------------------------
Pass compile plcprograms
------------------------------------------------------------------------
Do you want to continue ? [y/n/go] y
...
Berkeley DB 4.6.21: (September 27, 2007)
info get: 0
I Database opened /data0/pwrp/opg2/src/db/volopg.db
-- Plc window generated            F1-Z1-Plc-W
-- Plc window compiled for x86_linux optimized -O3 F1-Z1-Plc-W
-- Plc plcpgm compiled for x86_linux optimized -O3 F1-Z1-Plc
-- Plc window generated            F1-Z2-Plc-W
-- Plc window compiled for x86_linux optimized -O3 F1-Z2-Plc-W
-- Plc plcpgm compiled for x86_linux optimized -O3 F1-Z2-Plc

------------------------------------------------------------------------
Pass create loadfiles
------------------------------------------------------------------------
Do you want to continue ? [y/n/go] y
-- Removing old loadfiles
rm: cannot remove `/data0/pwrp/opg2/bld/common/load/ld_vol*.dat': No
such file or directory
...
Berkeley DB 4.6.21: (September 27, 2007)
info get: 0
I Database opened /data0/pwrp/opg2/src/db/volopg.db
-- Building archive for volume: 000_001_001_012
-- Archive built for volume: 000_001_001_012

-- Working with load file volume 'VolOpg'...
-- Open file...
-- Successfully created load file for volume 'VolOpg'
-- 26 objects with a total body size of 21976 bytes were written to new
file.

Before this pass you should compile the modules included by ra_plc_user.


------------------------------------------------------------------------
Pass create bootfiles
------------------------------------------------------------------------
Do you want to continue ? [y/n/go] y
-- Creating bootfiles for all nodes

Proview is free software; covered by the GNU General Public License.
You can redistribute it and/or modify it under the terms of this
license.
```

```
Proview is distributed in the hope that it will be useful
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
GNU General Public License for more details.

-- Creating bootfile for node opg
      plc_opg_0507_00011
-- Plc thread generated priority 0, scantime   0.10000 s, 2 plcpgm's
-- Plc process compiled for x86_linux optimized -O3 Dummy
-- Plc program linked for x86_linux  node plc_opg_0507
-- Creating bootfile for node aristotle
      plc_aristotle_0517_00011
-- Plc thread generated priority 0, scantime   0.10000 s, 2 plcpgm's
-- Plc process compiled for x86_linux optimized -O3 Dummy
-- Plc program linked for x86_linux  node plc_aristotle_0517

-- The upgrade procedure is now accomplished.

setdb is obsolete
>
>
```