DESCRIPTION OF NEW FUNCTIONS IN AUTOMATION VER. 22

This release document describes the new functions in Automation version 22.

PCSCHEMATIC Automation has its own manual, which is included in the program.

The Panelrouter, the Component Wizard and PCSCHEMATIC Automation Service (the Mounting Assistant) have their own dedicated manuals describing their functions, and those manuals are also included in the Automation program.

Last edit: Julyl 2020

CONTENTS

1 NEW LOGOS

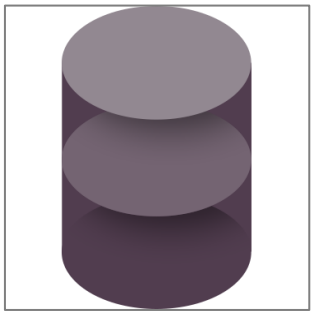
Hard to hide: our programs have new logos:

Automation

Automation Service

Automation Viewer

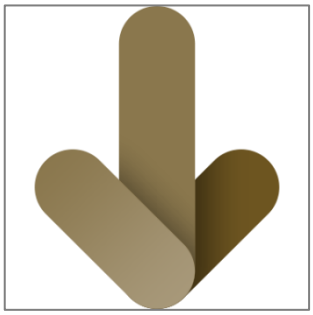
Database



License Manager

License Manager Config

Installer



We are implementing the new logos in our programs, on our website, in advertissements etc.. Our other programs also have new logos.
Our company logo and website also changes. Look and read much more on www.pcschematic.com.

2 PREPARING FOR A NEW COMPONENT DATABASE

At the user meeting held in October and November 2019, we announced that we are making a new structure for our component database. And in that also means that we will 'tidy up' in some functions.

Generally, you are going to work more component oriented, when you use the database. And this will lead to changes in symbols and projects as described below.

The changes that we make in this version, and that you can read more about in the chapters below, have the following headlines:

- We make limitations to what can be changed on a component on the project
- Symboltype 2 is discontinued
    - We have made a major restructuring of functions and types of connection points

i

When you load an existing project in to ver. 22 and/or fetch a component from the database, we convert the connection points in the project according to the rules described in this chapter.

The easiest way of seeing the changes is by making a new installation, instead of an update. You can make the update later, after the inspection.

The Component Portal is not yet ready for release, but when it happens, we introduce more changes, and they will be described at that time.

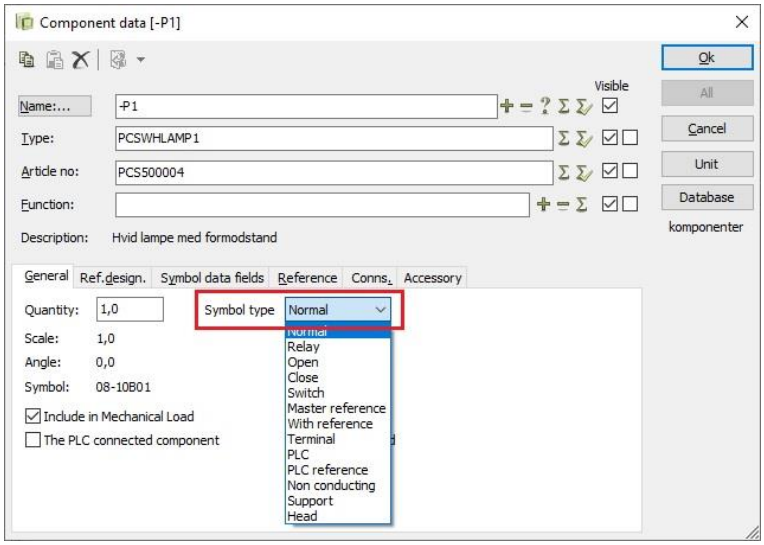
3 CHANGES IN SYMBOLS AND CON. POINTS

3.1 The symbol type cannot be changed in the project

When you place a symbol, it has been possible to change the symbol type.

This is not possible from ver. 22.

It was useful earlier, if you needed to have the component's conncetion ofn the terminal list: Earlier you would change symbol type 1 to terminal and then selected the proper symbol type 2 and then changed the connection points. According to some not-so-obvious rules.



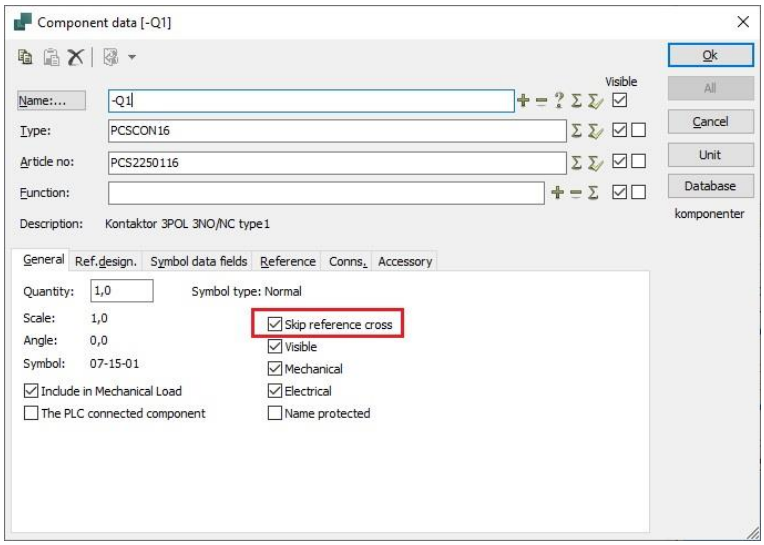
This is changed now!

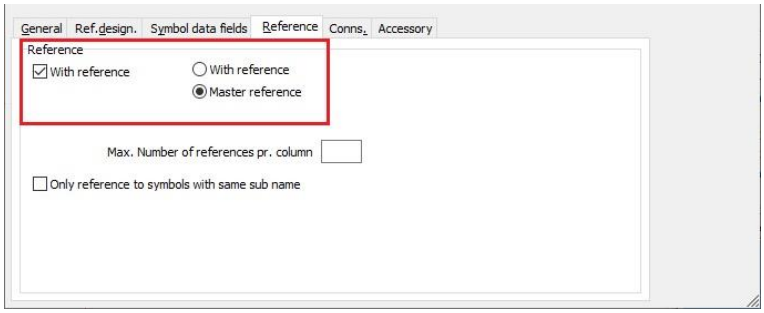
In the following, you can read what this means for all symbol types and connection point types.

3.1.1 Reference cross can be deselected for Relay symbols

Because it is not possible to change symbol types in the projects, you can now show the relay symbol without reference cross.

On the Reference tab you choose how to treat the references; you probably want to use the coil as Main reference. [ii]





3.2 Symbol states on component symbols cannot be changed

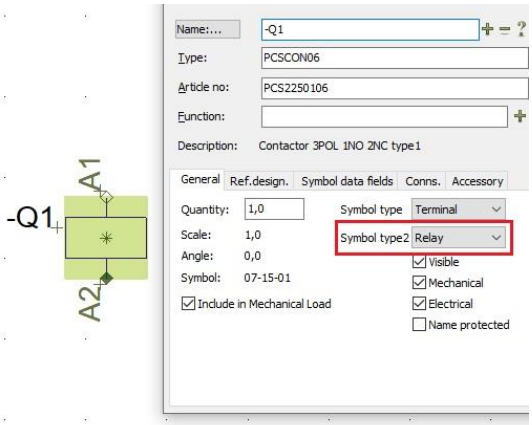
When a component is created in the database, the creator has chosen the right symbol(s) for it. That also means, that if the symbol has various states, it is no longer possible to change the synmbol state in the project.

If the symbol is in state 0, it is stil possible to change it.

3.3 Symbol type 2 is discontinued

In connection with optimizing symbol types, we discontinue Symbol type 2.[iii]

Symbol type 2 has only existed to make it possible to show the component's connections on the terminal list. This is now solved in a different way, which you can read more about later in the next chapter.



3.4 A simpler definition of terminals

Until now, we have had many different definitions of terminals, and we have used symbol type 2 for this. The purpose has been making it possible to show connections on the terminal list, so that the list could show all connections that should be mounted.

3.5 Terminals are ALWAYS through terminals

In the future, the following is valid for terminals:

- Components with the symbol type Terminals, which we hereafter call Terminals, are ALWAYS through terminals, meaning that you have the same potential on both sides of the component!
- All connections have the same name.
- There is minimum one connection on side 1 and side 2, respectively, of the component.
- All connections are by default on the terminal list.

The rule, that all connections on a terminal have the same name means, that a terminal block with prenamed connections will be treated as a NORMAL component and not a TERMINAL component!

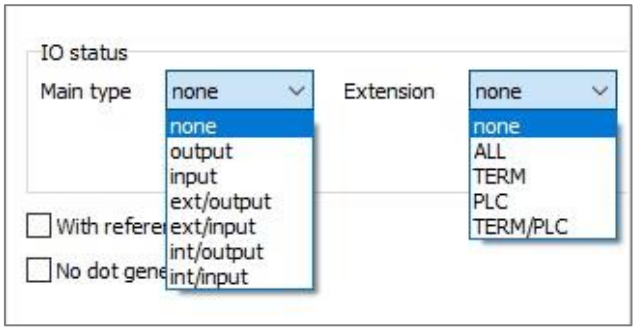
It also means that the old types, like plc-terminal, relay-terminal etc are discontinued. Read more about this later, and see how to get their connections on the terminallist (it will be easier 😊).

3.5.1 Conversion of connection points

As a consequence of discontinuing symbol type 2, many combinations of connections' main type and sub type are irrelecant.

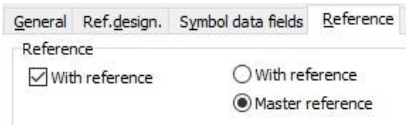
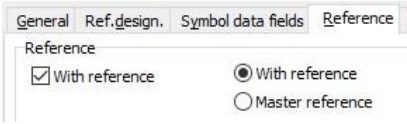
Below you can see, how the combinations are converted. [iv v]

The old options – a lot of options, where many weren't used or simply not understood.



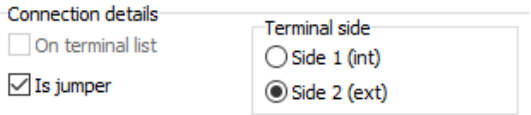
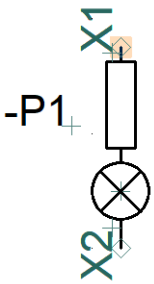
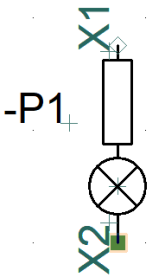
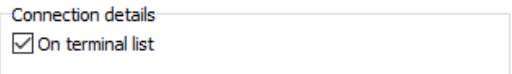
3.5.2 Change of syntaxes

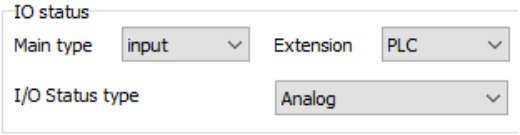
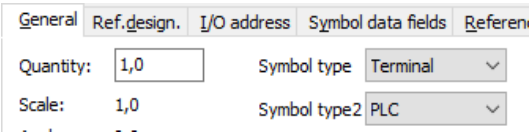
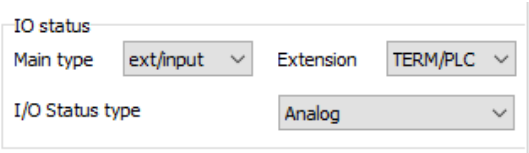
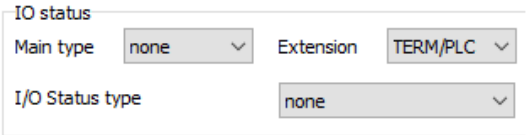
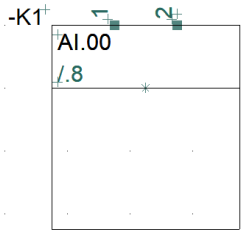
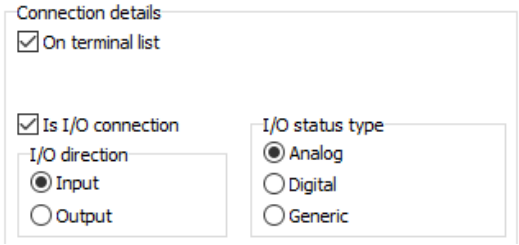
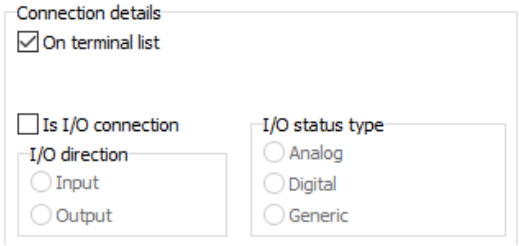
Symbol types can still be controlled by the database. The following changes to the syntax have been made:

Old syntax	New syntax
Syntax=T,O (Terminal, Open)	Symboltype1 = Open
Syntax=T,C (Terminal, Closed)	Symboltype1 = Closed
Syntax=T,S (Terminal, Switch)	Symboltype1 = Switch
Syntax=T,P (Terminal, PLC)	Symboltype1 = PLC
Syntax=T,N (Terminal, Normal)	Symboltype1 = Normal
Syntax=T,R (Terminal, Relay)	Symboltype2 = Relay
Syntax *M (Symboltype = Master Reference)	Syntax is not changed, but it will be read as Symboltype1 = Normal, with Reference as Master reference 
Syntax *W (Symboltype = WithReference)[vi]	Syntax is not changed, but it will be read as Symboltype1 = Normal, with Reference as Master reference 
Not found	Syntax=/I:IJ – connection point for jumper only. Side 1, that is the internal side.
Nout found	Syntax=/I:EJ – connection point for jumper only. Side 2, that is the external side.

3.6 Overview of changes to connection points

Earlier (until and including ver. 21)	Now (from ver. 22)
IO status Main type: input Extension: TERM In the terminal list, there is a data field for connection points, input side.	The internal side of the terminal is now called Side 1, and its connections are on the terminal list, when the check mark is set in the 'On terminal list'.  The terminal side is indicated as earlier. On terminal list is shown as a square. Connection details ☑ On terminal list ☐ Is jumper Terminal side ◉ Side 1 (int) ○ Side 2 (ext)
IO status Main type: output Extension: TERM In the terminal list, there is a data field for connection points, output side.	The external side of the terminal is now called Side 2, and its connections are on the terminal list, when the check mark is set in the 'On terminal list'.  The terminal side is indicated as earlier. On terminal list is shown as a square. Connection details ☑ On terminal list ☐ Is jumper Terminal side ○ Side 1 (int) ◉ Side 2 (ext)
If you didn't want the connection on the terminal list, you needed to change the symbol type to Normal and change the connection point to None/None. Nothing was indicated in the diagram.	If you DON'T wish to show the connection on the terminal list:  Side 1 is not on the terminal list, and this is indicated with the normal diamond. The side of the terminal is kept. Connection details ☐ On terminal list ☐ Is jumper Terminal side ◉ Side 1 (int) ○ Side 2 (ext)
If you didn't want the connection on the terminal list, you needed to change the symbol type to Normal and change the connection point to None/None. Nothing was indicated in the diagram.	If you DON'T wish to show the connection on the terminal list:  Side 2 is not on the terminal list, and this is indicated with the normal diamond. The side of the terminal is kept. Connection details ☐ On terminal list ☐ Is jumper Terminal side ○ Side 1 (int) ◉ Side 2 (ext)

Earlier (until and including ver. 21)	Now (from ver. 22)
Not possible.	When you have a component, you might have a connection point only for jumpers. This is indicated with a small triangle. The side of the terminal is kept. Only jumpers are allowed to connect to this type of connection point.  
Not possible.	When you have a component, you might have a connection point only for jumpers. This is indicated with a small triangle. The side of the terminal is kept. Only jumpers are allowed to connect to this type of connection point.  
Symbol type 1: Terminal Symbol type 2: selected Individual setting on the connection point.	For symbols of types Normal, Relay, Open, Close, Switch, it is now possible to select per symbol to have its connections on the terminal list. Non-selected connections are indicated as previously.  
Symbol type 1: Terminal Symbol type 2: selected Individual setting on the connection point.	For symbols of types Normal, Relay, Open, Close, Switch, it is now possible to select per symbol to have its connections on the terminal list. Selected connections are indicated with a filled-out square, as they are considered to be on side 2 of the terminal row.  

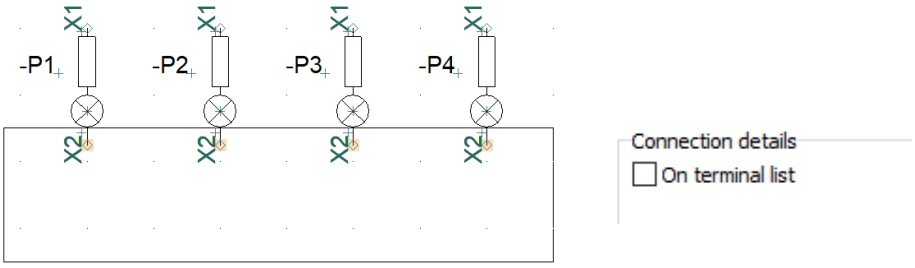
Earlier (until and including ver. 21)	Now (from ver. 22)
Same symbol's connection points in ver. 21: Connection point 1:  If you want to see the connections to this address on the terminal list, you need to do the following: Change the symbol type:  Make settings to connection point 1:  and to connection point 2:  	Here you see a PLC input symbol.  If you want to have the connection points on the terminal list, you simply tick the box 'On terminal list'. You can also see, whether you have an IO connection or not. It is also easy to select status type. Connection point1:  vii Connection point 2: 

Tips:

If you want to select more connection points in one operation:

Drag the mouse across the select, hold down the Ctrl-button when you finish the selection.

And use the button Object data in the Toolbar to select the desired connection details.



3.7 Indication for parked lines

We have made a lot of changes of how we show connection point statuses.

The 'old' way of showing a parked line (not-connected, conducting line) looks very much like a terminal, side 1.

Therefore, we have made a new marking of parked lines. This marking becomes smaller, the more you zoom in the page

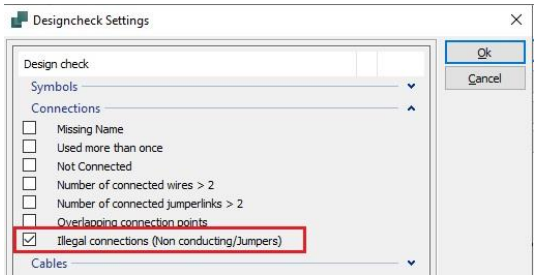
3.8 Symbol editor will check for inconsistencies

From ver. 22, the symbol editor will make sure, that symbols aren't created with incorrect functionalities on connection points. [viii]

The old syntaxes are kept on your existing components, but they are converted as explained in the table above. We have added two new syntaxes, and they are also explained in the table above.

3.9 More functionality in Designcheck

As a consequence of the new jumper connection option, Design check will check for illegal connections to such a connection point.



4 TRIM YOUR DATABASE 😊

Remember, the truth about components is found in the database!

That also means, that many functions in the program depend on the quality of your database; is its setup ok, does it have the database fields that are necessary for various modules like the Component Wizard?

4.1.1 Do it this way – create your own database

The database, we deliver with the program, contains all necessary fields.

In the database program (not Automation), you can create your own database:

- Go to Files|New
- Make a new database as a copy of our database. Give it a good name.
- Check that the menu file is from the new database!
- If you need to add extra database fields, go to Files|Properties, and add/insert the desired data fields.
- Import your existing components into your new database.
- Go to Automation.
- Go to settings|Database and select it as your new database.

Ready to go 😊

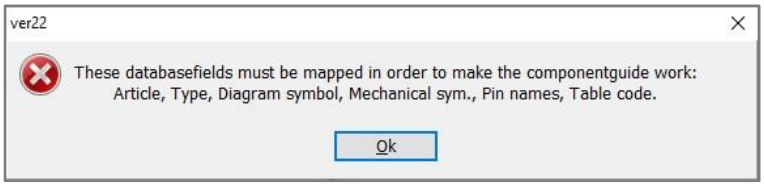
4.2 Component Wizard

We are improving the Component Wizard continiously and we recommend that you use it when you create new components in the database. Below

4.2.1 Warning for non-mapped fields in database setup

The Component Wizard requires that some database fields are mapped correctly. If one of them is mapped, you can't open the wizard, and you get a warning during startup, telling you about the missing fields.

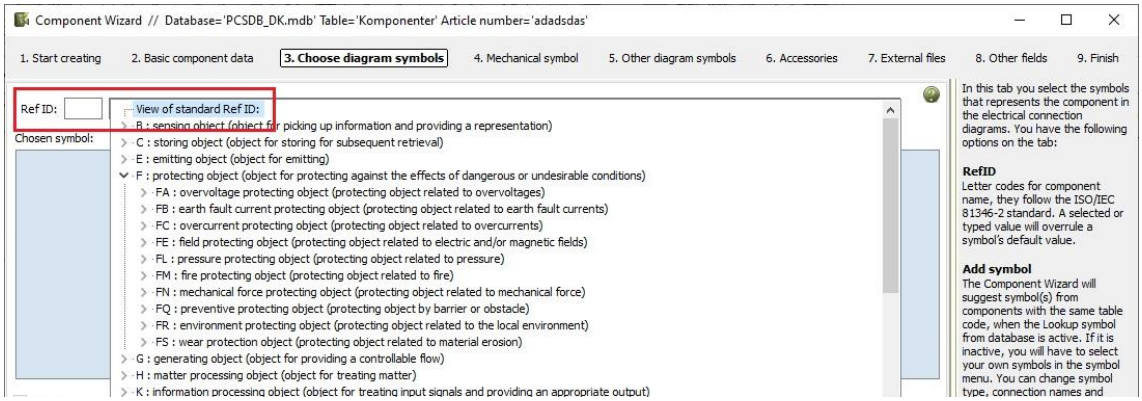
Apart from those fields, the database contain the CWCODE data fields.



4.2.2 Letter codes from 81346-2 are also in the Component Wizard

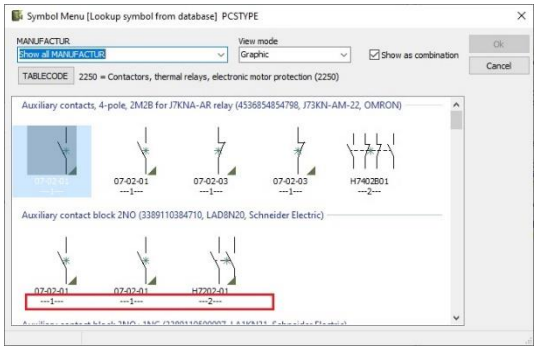
When you create new compponents by using the Component Wizard, we have updated the list of code letters. The standard was updated in 2019.

Now you can choose to use 1, 2 or 3-letter codes for your components.



4.2.3 Selecting diagram symbols shows alternatives

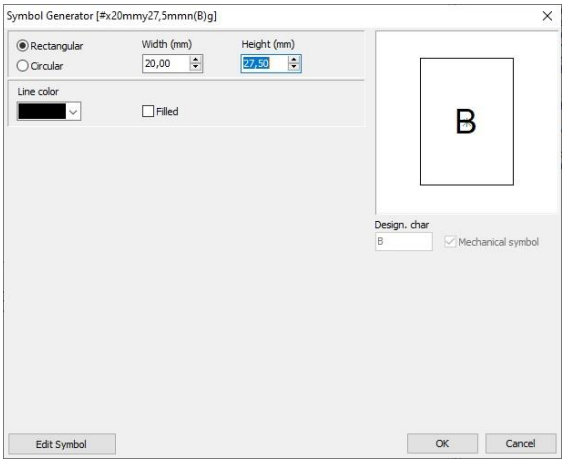
When looking for the correct diagram symbols, the window will now show alternatives.



4.2.4 Changes regarding connection points

All describes changes regarding connection points that are described in this document, are also valid in the Component Wizard.

4.2.5 Mechanical symbols in the Component Wizard

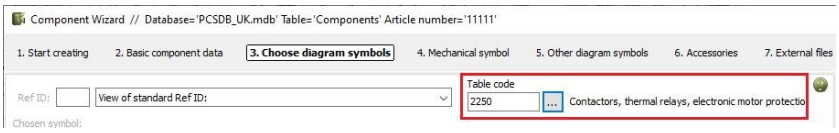


When you want to create a mechanical symbol for your component, we have removed Automec in the standard setup. That means, that when you use our standard database, it is gone.

Instead we have adjusted the Symbol Generator in the following way:

- you can't select connection points: earlier you could select them, but we removed them when saving the symbol ...
- component name shows your selection from RefID
- the 'Mechanical symbol' is already selected.

4.2.6 You can change table codes on components in the database



If you have created components with the wrong table code (often blank), you can now move them by using the Component Wizard:[x]

- If you edit one component, simply change the table code on the first tab
- If you change more components, you change the table code in the Diagram symbol tab – next to RefID.

4.2.7 Adjustable Help pane

The width of the Help section can now be adjusted – simply drag the splitter to the desired position.

4.2.8 Automec and prices are now longer mapped in new installations

We are going to replace Automec with something else in the coming up database. It is not our impression that the Automec function has been used widely? But if you did, we didnt delete it, we just hid it.

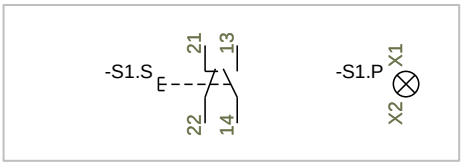
The same goes for prices. We don't maintain any price information in the system. If you want to do that, you can still do so.

Did you know.... The data fields can be used for summing other units, ie kW eller kg. If you do so, however, you must be consequent about units when creating your components.

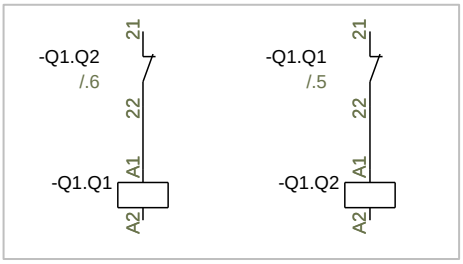
5 USAGE OF SUBNAMES - CLARIFICATION

Subnames is a concept in PCSCHEMATIC Automation, and it has been there in many program versions.

The function was made to be able to show sub-divisions of a component, is a switchbutton with a lamp indicator, which consists of a function for the switch button itself and a function for the lamp. The component is called S1!



Another usage of the function could be a disconnector with two coils, in which the internal connections and activations are clearly indicated. The component is called Q1!



What the two examples have in common, is that the subname is part of the component, that means a part of a certain article, and that the subname is fixed for this article and that it is controlled by the database.

Another common thing is, that the component is not divided into parts, and in the diagram you only show and name the components main function, which are S and Q respectively.

That also means, that in the Component menu, we only show the *Component name* and not the *symbol name*.

That also means that when you take a relay from the database, that was created *without* subnames, and give it a subname in the project – which is not according to the rules according to above – then you only see the main name of the relay. And that is exactly the meaning.

Dividing components into part and subnames as such are practical tools when you want to show the internal functions – in the contactor – or if you need to replace one component with several components – as in the switchbutton with lamp indicator. However, you might also use the reference designations, that is the product aspect.

You can find the switch in the pickmenu, and if you want to create your own, simply use the Component Wizard.

5.1 Naming convention of cards in a plc

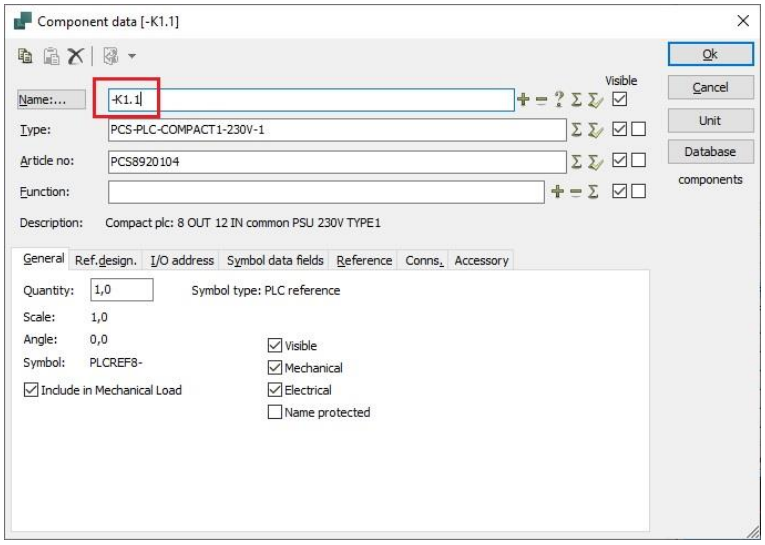
We have – unfortunately – told everyone, that you can use subnames to indicate slot numbers on plcs. That should not be done. Anymore.

Instead, you should name the cards in the diagram in this way:

Card for Slot 1: -K1.1

Card for Slot 2: -K1.2

Visuallly, in the diagram, there is no difference at all. But naming the card in this way means, that the program works correctly: The Component Menu shows the right IOs, and import of IO-data works correctly.



If you do it the ‘old’ way everything except the Component menu works.

Unfortunately, we also need to edit our Tutorials and YouTube-videos.

6 HOW TO WORK WITH ACCESSORIES?

How to work with accessories changed from ver. 20 and we have also come up with a few extra changes in versions 21 and 22, so below you get a 'guided tour' through the accessory function in the program.

6.1 Purpose of changing work process and method

1. As a user, I wish to have overview of selected accossories for a given component.
2. As a user, I wish that it is easy to select the correc accesory for a component.
3. As a user, I wish to make a copy of a component including its accessories.

6.1.1 Better overview

When you open Component Data on a master component, you can see all fixed and optional and maybe selected accessories. If the accessories have been created with a picture in our database, you can see the picture, when you select the article on this tab.

When you have seleced accessories for at component, the Component menu will show a tree structure – the branches show the accessories. Fixed accessories will not show if you don't place it.

You will get the most out of the accessory function if all optional accessories are created in the database.

6.1.2 Electrical accessories

Are selected in the Component menu on DIA-pages. Electrical accessories have their own diagram symbols.

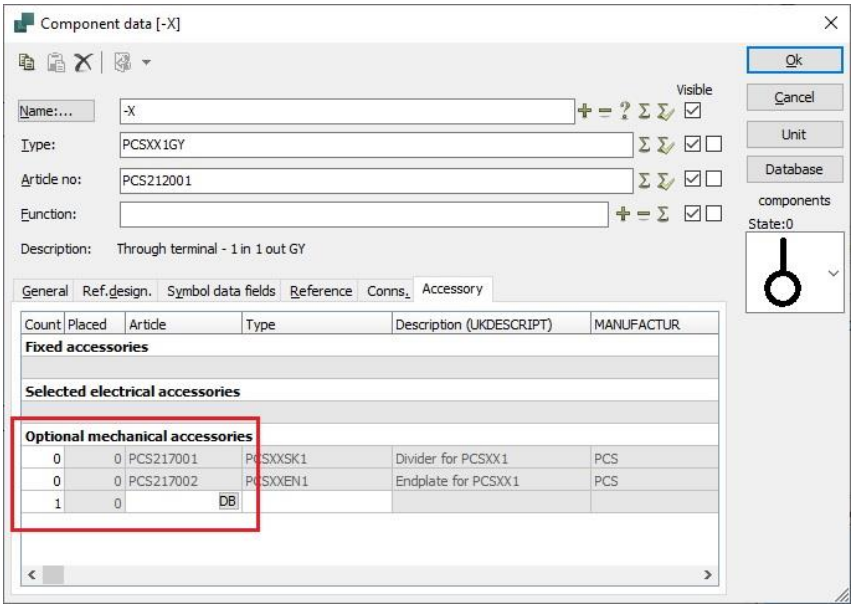
If you don't select the accessory through the Component menu, you won't get the tree structure and it won't be regarded as accessory by the program.

6.1.3 Mechanical accessories

The easiest way to select mechanical accessories is to select it on the Accessory tab.

In that way, all selected accessories are included in the parts and components lists and if it has a mechanical symbol, you can – optionallly – place it on the GRP-page, by using the Component menu's available symbols.

You can also place mechanical accessories directly on the GRP-page, as long as it has mechanical symbols.



6.1.4 Fixed accessories

Fixed accessories are defined in the database and can only be changed there.

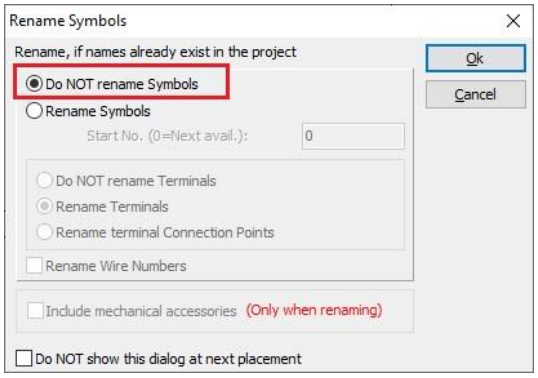
6.1.5 Copy a component with mechanical accessories

Accessories are connected to the main component.

When the program can't see, which component the accessory is connected to, it will not be copied.

This is relevant, when you copy and you don't rename the components.

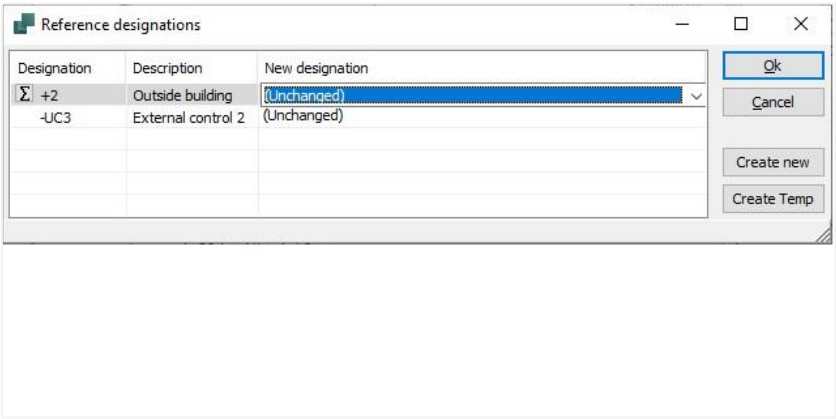
Be aware, that this is also done, when you work with naming according to page and current path, no matter whether or not your terminals follow the naming convention.



6.1.6 Copy and reference designations

When you work with ref. designations, the program will help you select a new designation for the copied parts.

If you don't know which designation to apply, use a temporary one, which is easy to change at a later stage.



Remember

'Like' doesn't mean 'the same'!

Same article number doesnt mean same component name.

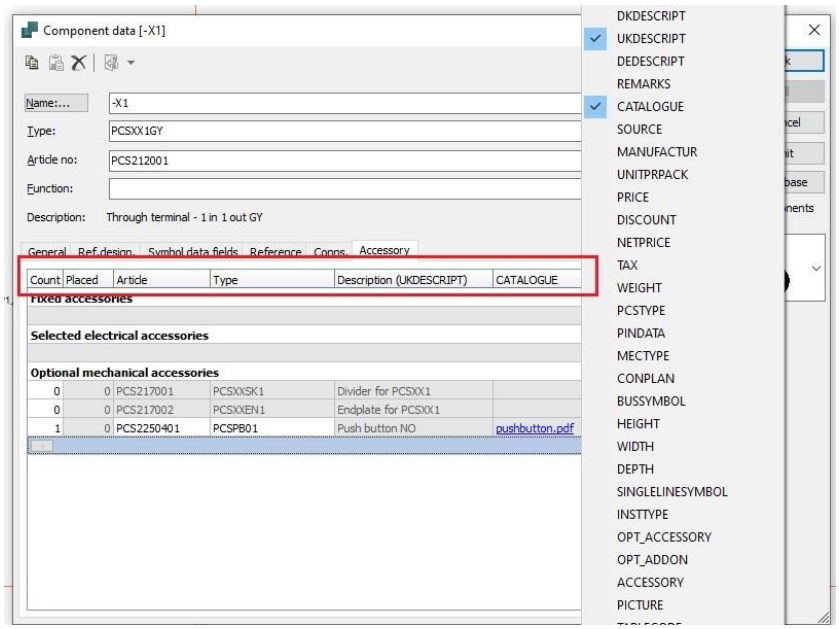
Same component name doesn't means same reference designaiton.

6.2 The accessory tab is configurable

You can adjust the width of columns on the accessory tab.

If you rightclick in the top row, you can select which database fields you want to see for the accessory. The setting is for all component data dialogs.

Remember, that the data you select to show in this way, is also seen in the small popup-info, so don't select too many.



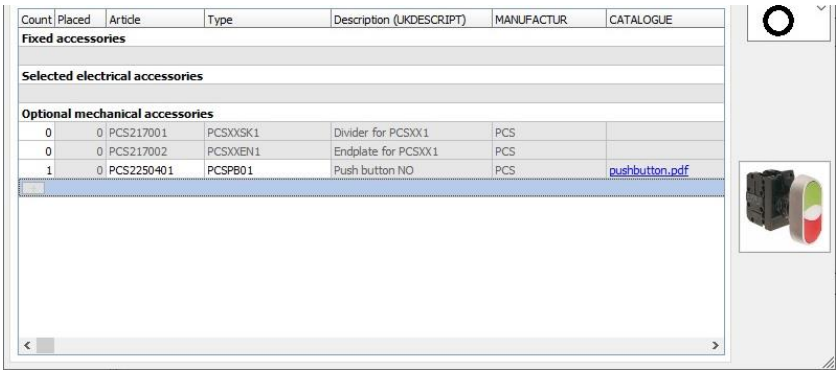
Some data fields can contain links, if so, you can open the link directly from here.

And remember, links to the main component are found through the Folder icon in the top of the Component data dialog.

6.3 Any pictures can be viewed on the tab

If your accessory – optional or selected – has a pictoure connected in the database, you can view it, when the mouse comes over the row.[xi]

Your component data dialog must be tall enough to allow space for the picture!



6.4 Support symbols and accessory

Support symbols are the ’old’ way of working with accessories. What happened with them?

6.4.1 Support symbols in existing projects

Support symbols in existing projects work as always. That means that if you have an existing project where optional mechanical accessory is added by using support symbols, they are also part of the project in ver. 21 and 22.

However, you should know, that it is not recognized as accessory, meaning that it will not be shown as *accessory* on the tab or as a branch in the Component menu.

It also means, that if you copy the main component, the accessory is not automatically includes, because the program doesn’t know that it is accessory.

It still included in the parts and component lists.

6.4.2 Support symbols are ignored in the Component menu from ver. 21

If a component’s only diagram symbol is a support symbol, this symbol will be ignoret in the Component menu from ver. 21.

That means, that you won’t find any ‘available’ support symbols in this way, thus rminimizing the risk of selecting too much accessory.

Still, if the support symbol is only one of more diagram symbols, it is still shown as a reminder of possible accessories for the component.

7 TERMINALS IN COMP.GROUP 0 AND ACCESSORIES *

We have changed the rules about accessories slightly, in general and particularly regaring terminals.

That means, that it is not possible to have accessories on terminals in component group 0.

If you haven't thought deeply about component gropus earlier, then simply skip the rest of this section. 😉

7.1.1 A little history

In older versions, symbols – and among them terminals – were placed in the project in component group 0. That simply meant that in the project you hadn't decided which components to use. Yet.

When – later – you selected your article numbers, all components – except multi-layer terminals – could keep this component group and your parts list would count correctly. Almost always.

For normal, one-layer through terminal, it meant that every symbol counted as one component, meaning that the parts list was ok, but when you placed the mechanical symbols for the components, the program couldn't keep track of which symbols belonged to which components, meaning that the right-click function Go to symbol didn't work.

For several versions, the program hasn't assigned component group 0, but there are still many project 'out there' that use component group 0, simply because you reuse (parts of) your old projects.

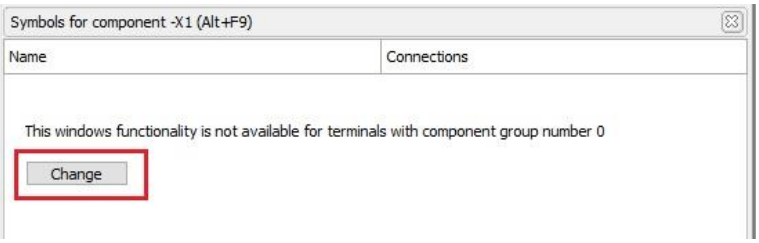
What the program has done in the later versions is, that components get consequent component group numbers, which means that it is possible to track the individual terminal on all pages in the project; this simply means that all components have a unique ID. And then you can use the rightclick function to go to the component's symbols on various pages.

Still, there are more challenges to 'guessing right' and most of the challenges are related to accessories. And that is the reason why we are phasing out component group 0.

7.1.2 Terminals with accessories and they are not placed mechanically

Here are no problems. The parts and the components lists are correct, so continue the good work.

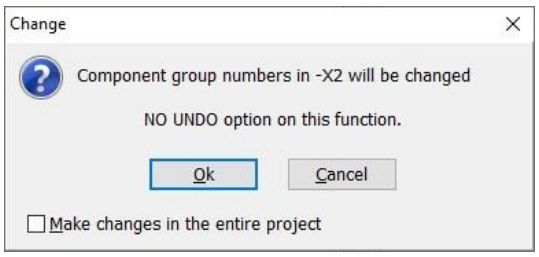
But, if you want to continue working with the terminal row, either by clicking on the component on the diagram page to open the Component data dialog or if you want to place the components on the mechanical page, you must use the Change function, simply click the button in the Component menu.



You can't edit the terminal row when its symbols are in component group 0.

When you click the button, all previously selected accessories to the terminal row will be added to one of terminals in the terminal row, and their components groups will become 1, 2, 3, etc.. The parts list will be correct, and you can place the components freely on the mechanial page.

In ver. 22 you can change all terminal rows in the project at once.



7.1.3 Terminals with accessories and they are placed mechanically

When you have terminals with accessories and they are placed mechanically, you might experience incorrect parts lists. It depends on how you selected the accessories and how you placed the mechanical symbols.

7.1.3.1 Accessories selected with support symbols

This is the oldest way of selecting accessories. Still, sometimes, you see an incorrect parts list.

7.1.3.2 Accessories selected on the (Mechanical) Accessory tab

Since ver. 18 it has been possible to select accessories on the Accessory tab. That works perfectly, unless you have set the terminal row back to component group 0!

7.1.4 The easiest way to correct my project's terminal rows when my parts list is incorrect due to component group 0

The easiest and simplest way may sound drastic! But it works.

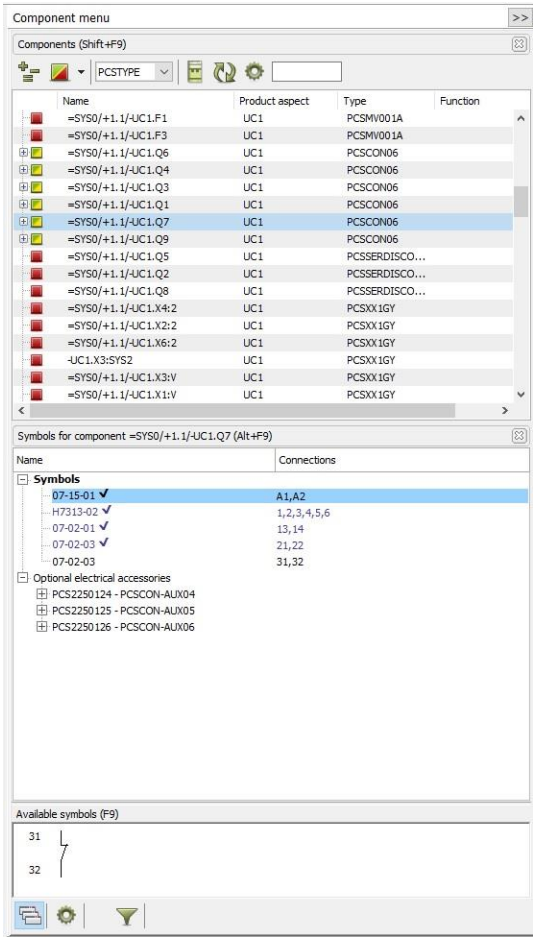
1. Delete the terminal row on the mechanical page!
2. Go to the diagram page and use the Change button.
3. Go back to the mechanical page and place the terminal row again
    a. Rightclick on one of terminals, select Component name, and in the symbol window you can select all symbols in the terminal row, all in the right order Finally, place the selected accessories.

And now, the lists are correct, and you can go from diagram to mechanical and back again on a selected component's symbols.

8 SMALL CHANGES TO THE COMPONENT MENU

The Component Menu now remembers size etc.:

- The menu's width
- Each window's size
- Symbol size
- Stacked symbols – or not-stacked



8.1.1 Branches are closed

When you have components with alternative symbols, the alternative's braches are closed as soon as they become irrelevant.

8.1.2 The menu stays closed

If you choose to close the Component menu, it stays closed in most cases:

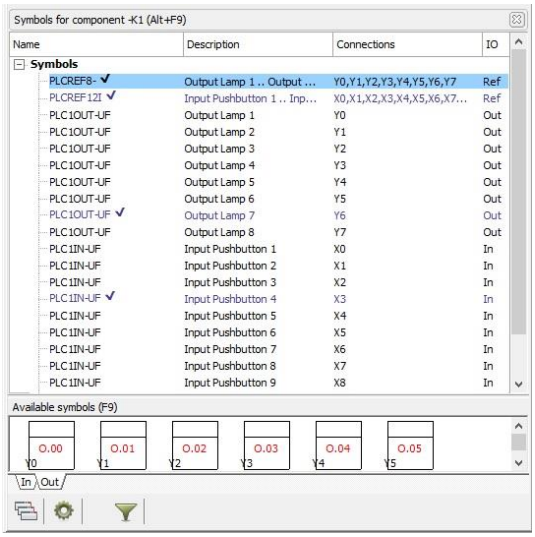
If you scroll through your project or if you add a component with only one symbol.

The menu opens automatically, if you add a component with more than one symbol – as this is where you select the symbols for the diagram.[xii]

8.1.3 Popup on optional accessories

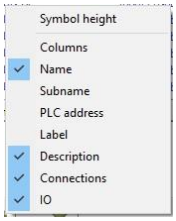
Optional accessories has a popup that shows article no., type and description.

On the Accessory tab you can also see more info, among it a picture, if one was added, link to data sheet etc



8.1.4 Select columns in the symbol window

When you press the settings icon at the buttom (the cogwheel) you can select which columns to show. Some columns are only relevant for plcs, and will be dimmed for other components.[xiii]



8.2 Placing mechanical symbols

On the GRP-page. when you double-click on a component name, where the mechanical symbol isn't placed yet, you will get the symbol in the crossh hair.

In this way you will save clicks and make it easy to place the symbol.

8.3 Changed shortcut

The new function above means, that the 'old' shortcut 'Double-click to select component name' is discontinued.

Double-click on a component in the Component menu is changed in the following way:

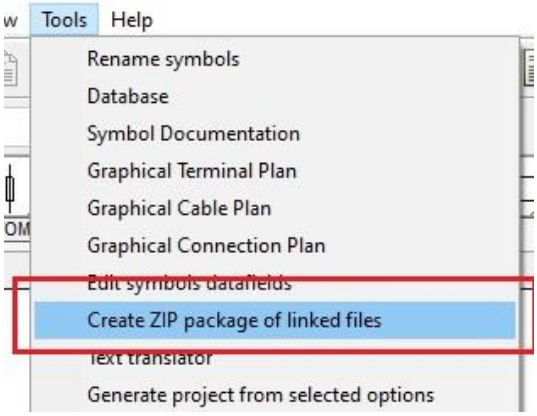
- On DIA-pages: nothing happens
- On GRP-pages: if the mechanical symbol isn't placed, you get it in the cross-hair
- Select component name is still there – in the right-click menu as earlier. The function is very practical, particularly when you place terminal rows.

9 LINKED DATA SHEETS CAN BE ZIPPED

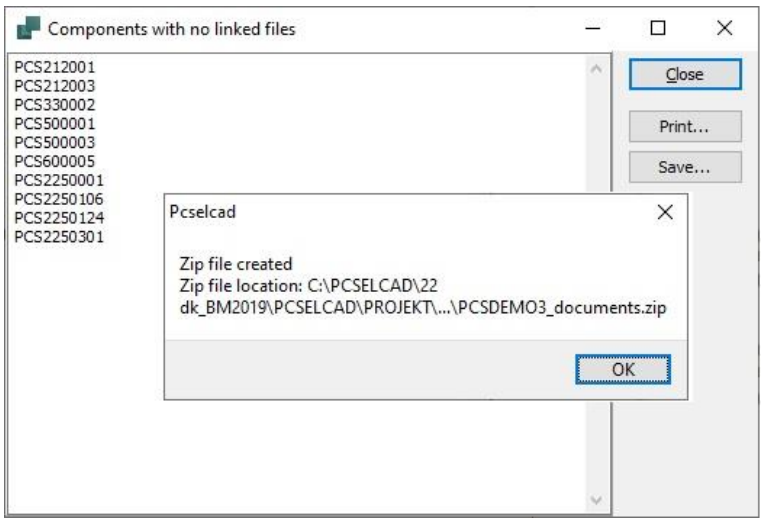
From version 22, you can create a zip-file with linked data sheets.

The function is a tool which can be found in the Tools menu.

The function selects the files that are linked by the 'Preferred link field' and packs the files into a zip-file. It also creates a list of components without linked files.[xiv]

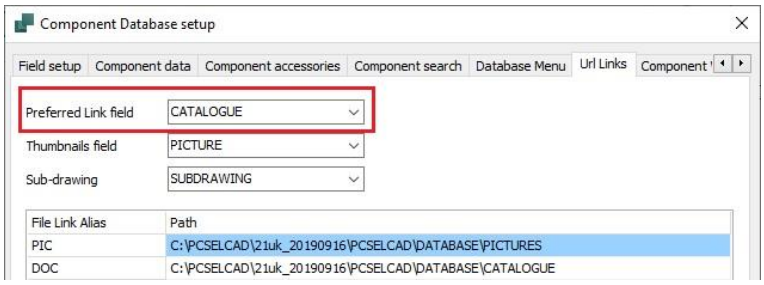


The zip-file is saved in the same folder as the project itself, and it has the same name as the pro-file, ie PCSDEMO3_documents.zip. [xv]



In this setup, we link to the database field CATALOGUE.

The module will only select files that it finds in the Alias folder, and not files that are linked to a www-address. [xvi]



10 SOMETHING ABOUT LISTS – IMPORT AND EXPORT

Being able to import and export lists to and from projects is a key function.

That also means that there are many wishes to functionality in this area, and also the reason why we almost always have new functions here.

10.1 Load parts and component lists using a format file

When you load a parts or component list, the program will now ask for the format file, that can interpret it.

That means, that when you load a list, you now select a format file, that defines which data is in which column. Earlier, parts and components lists had a fixed format, now it is possible how to read the lists in different formats.

When you place a component from the list it works like before.

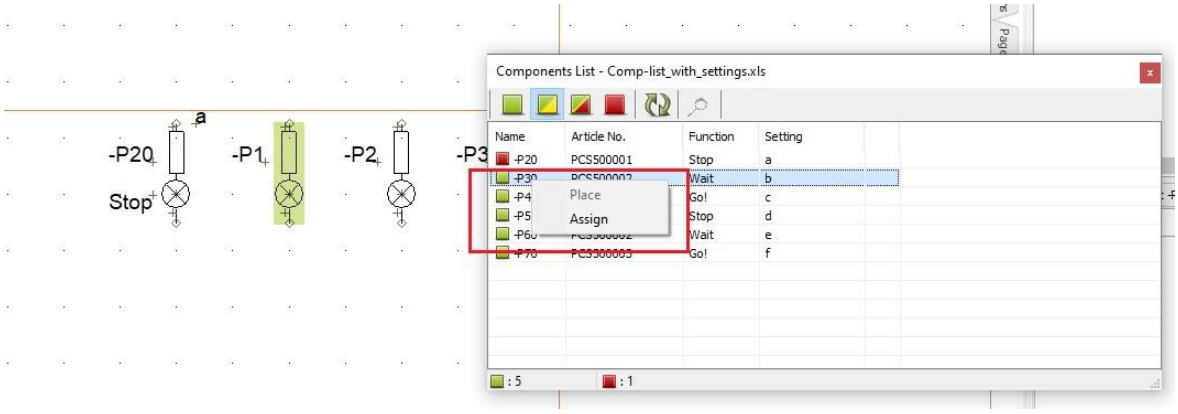
10.2 Assign data from an imported component list to the project

You can import a component list that contains component names and data and assign this data to already placed symbols in an existing project.

The component list can also contain data in symbol data fields.

It works the following way:

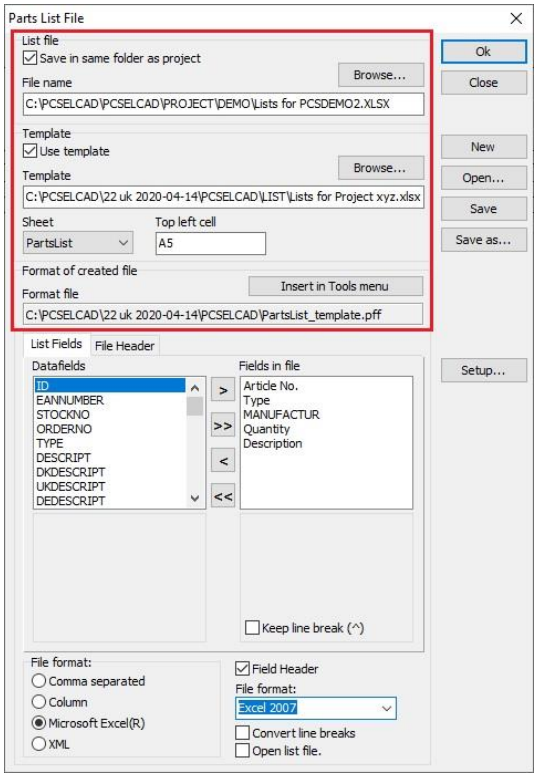
Start by creating a component list in Excel. It might look like the example below. The list's format follows a format file, in the same way as we define othe import and export formats in the program.



1. Select which list to import
2. Select the format file, that interprets it
3. Mark a symbol in the project, right-click on the row with the relevant data
4. Click Assign
5. Click OK in the Component data dialog. [xvii]

You can use the two new files 'Comp-list_with_settings', one is an Excel-file and one is a format file. They are both in the List folder.

10.3 List export to Excel can write to an existing template or file



When you export your lists to Excel, you often do it, because you want to reuse data for something else, ie for labelling.

List export is enhanced to that when you export to Excel you can select to export to an existing file/template, on a specific sheet and starting in a specific cell.

In this wat you can have several lists in the same Excel file without having to 'cut and paste' manually.

The function is only available when exporting to Microsoft Excel.[xviii]

You can use the function in this way:

- In the List folder, you find the file 'Lists for Project xyz'. Use this file as template for the first 'List-to-file' from the project.
- You can – as before – save the file with your own file name. Default file names indicate list type, but when you use this function, you will probably want to rename the file 'Lists for project abc'.
- For the next 'List-to-file' from this project, simply select 'Lists for project abc' as your template.
- The template contains a sheet for every list type. It is also possible to select a starting cell.[xix]
- All selections can – as earlier – be saved in a format file.
- All 'List-to-files' can be saved in the project folder.[xx]

The file Lists for Project xyz is included with ver 22.

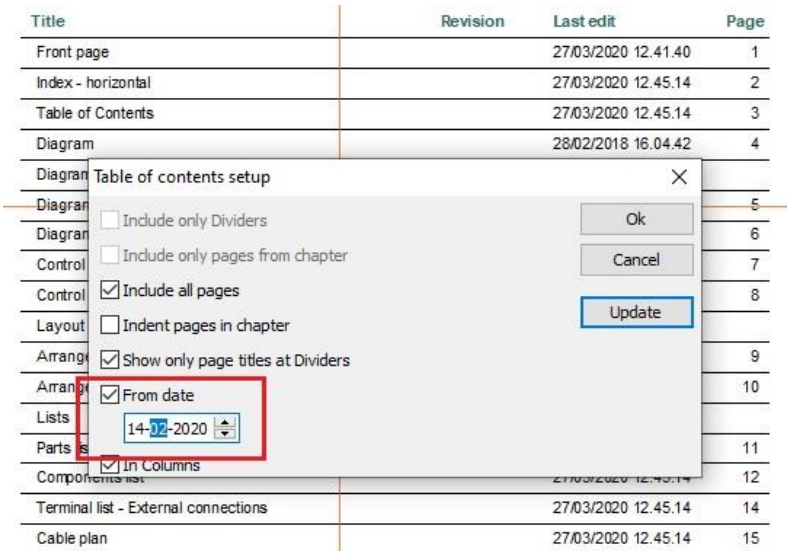
10.4 The Table of Contents can show pages changed after date

The Table of Contents setup has a new function:

It is now possible to show only the pages that have been changed after a selected date.

That gives you the option to quickly list changed pages.

Change date, month or year by selecting the section and use the arrows. [xxi]



10.5 Active hyperlinks to specific lines from a parts or components list

If you include lists in your parts and components lists, it is now possible to jump directly to a specific line, in the same way as you can with a component. [xxii]

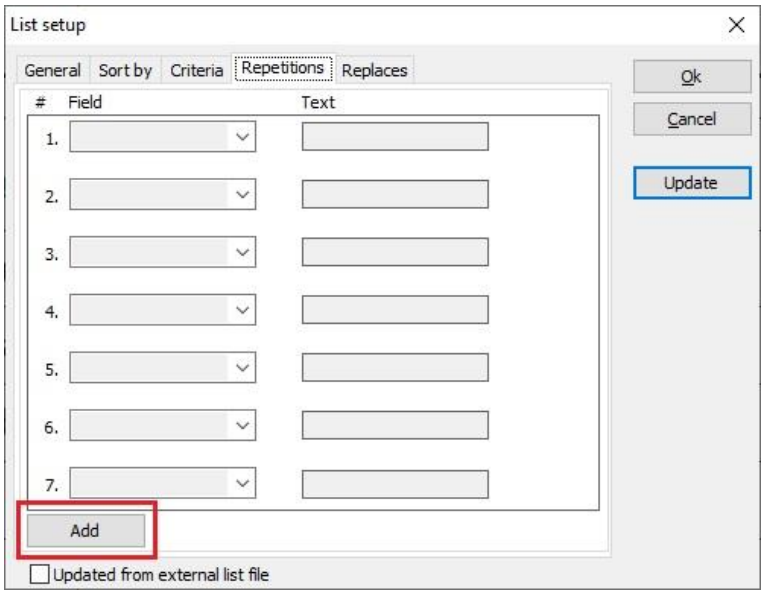
Component	Article no.	Type
LINE 1	asdf	Line
LINE 2	sdfg	Line
LINE 3	dfgh	Line

Note: If you load an old project, the list MUST be updated before this works!

10.6 Number of repetitions and replacements are extended

It is possible to decide how many repetitions and replacements you want to make in a list.

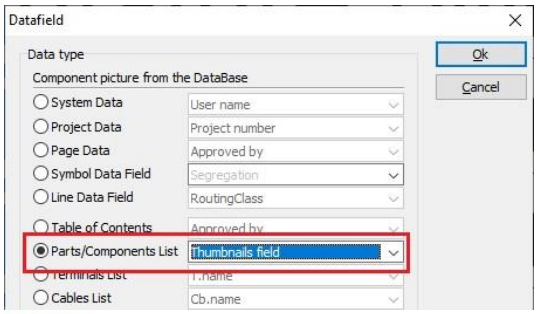
The 'Add' button has been added on both tabs. [xxiii]

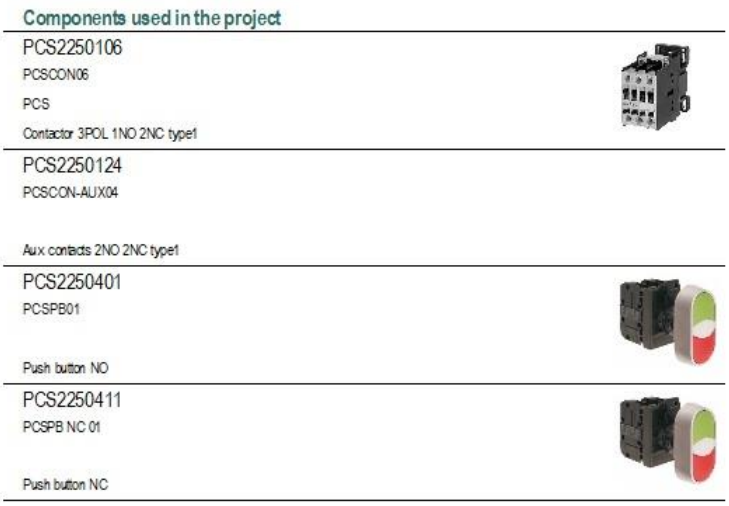


10.7 New data field in parts and component lists

The parts/components list section contains a new data field that can contain thumbnail pictures.

Below you can see an example of a parts list that contains article no., type, manufacturer, descriptiona and the new thumbnail picture.

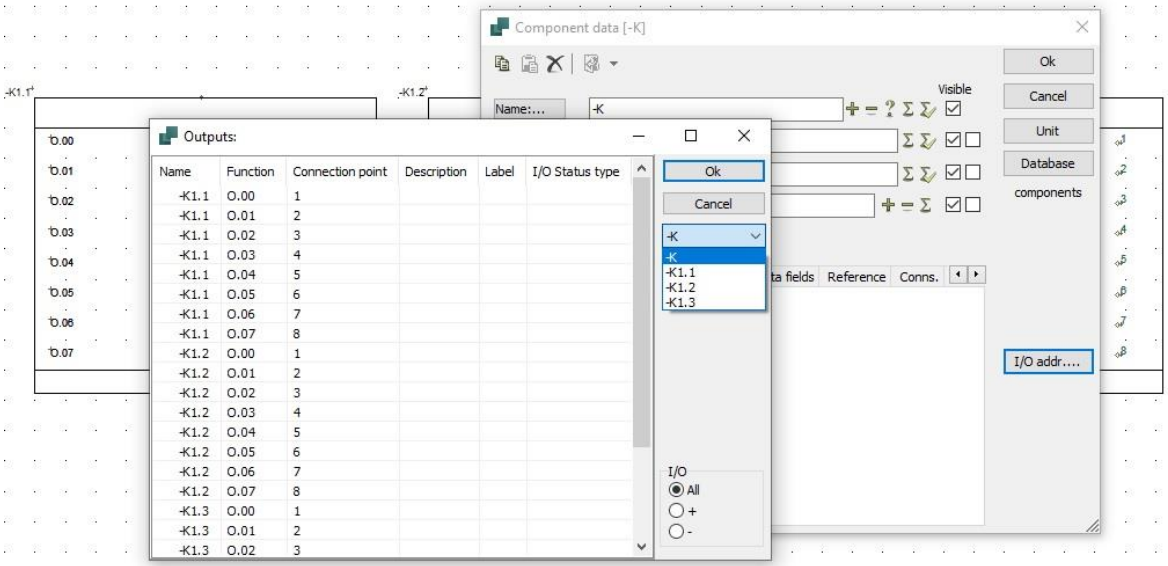




10.8 PLC list includes plc subname in the sorting *

The PLC-list now includes the plc subname in its sorting and other views.

This applies when you look for an address with the IO-button:



And it applies in the PLC-list.

Before you start using (or continue using) subnames, you should read about Subnames and plc on page 19.

Name	I/O	Description
-K1.1:1	O.00	Slot 1 - addr. O.00 - O.07
-K1.1:2	O.01	
-K1.1:3	O.02	
-K1.1:4	O.03	
-K1.1:5	O.04	
-K1.1:6	O.05	
-K1.1:7	O.06	
-K1.1:8	O.07	
-K1.2:1	O.00	Slot 2 - addr. O.00 - O.07
-K1.2:2	O.01	
-K1.2:3	O.02	
-K1.2:4	O.03	
-K1.2:5	O.04	
-K1.2:6	O.05	
-K1.2:7	O.06	
-K1.2:8	O.07	

10.9 Export to Cablemanager

This new button makes it possibel to export data directly from Automation to Cablemanager.

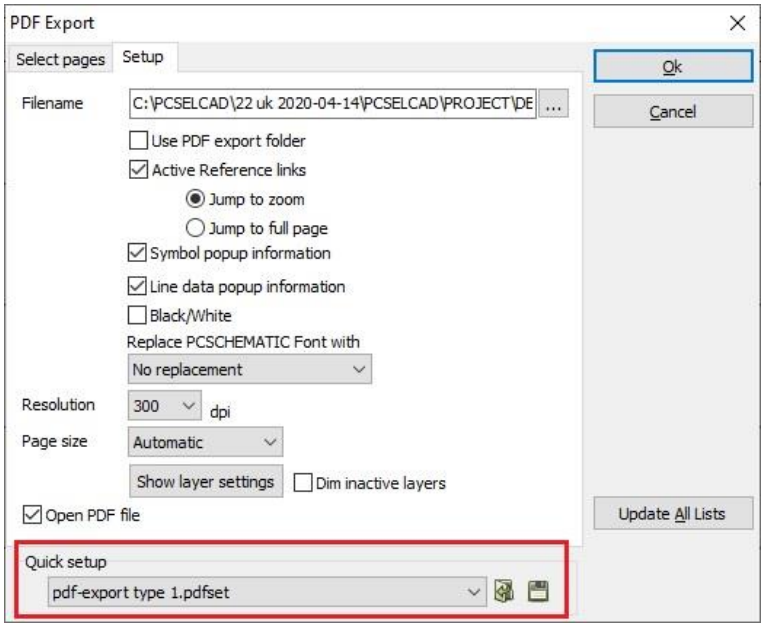
To make it work, you must have a Symbol data field called Segregation.

The field is created in a new installation, but you have to create it manually if you update your installation.

Cables that you want to export to Cablemaanger MUST have data in this field, otherwise they are not included in the export.

10.10PDF-export with command line

You can start a PDF-export with a command line, and you have been able to do that in many versions.



The new part is, that the command can use the settings in the Quick setup file.

The Quick setup example here simply contains the settings that you see above.I have made it by pressing Save and giving it a name. A practical function if you need different PDF-formats for different purposes.

The Quick setup default saved in xx\PCSELCAD.

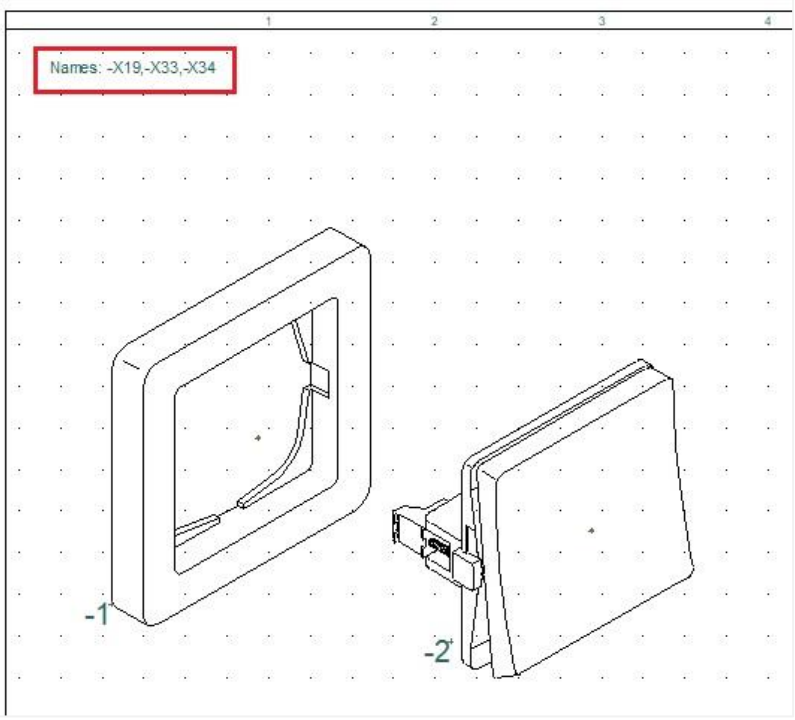
The command file format is:

FILESAVEAS.PDF pdf-export type 1.pdfset c:\pcselcad\Project\MyProject.pro

10.11 The namelist in unit drawings are with hyperlinks

When you click the namelist in unit drawings, they are with hyperlinks, which means that you will jump directly to the selected component in the project.

xxv



11 MOUNTING ASSISTANT

There are new functions to the Mounting Assistant:

11.1 Two or more users on the same project

If the project is saved on a common drive, then more users can open the same project and Order No.

There is always a Refresh button, and you get a message when the number of concurrent users change.

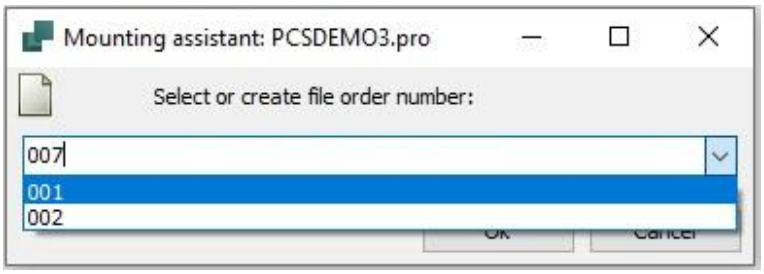
11.2 The connection can be selected in the list or in the diagram

The Mounting Assistant is based on the project's connection list, and for that reason all connections are presented in the same order as they are in the project.

As an alternative to finding the connection in the list, it is possible to select it directly in the diagram, and then the selected connection will be seen in the window.

11.3 All order numbers based in the project are in the dropdown list

All created order numbers on the current project are now seen in the dropdown list when you open the Mounting Assistant.

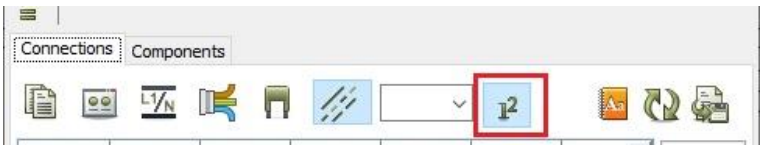


11.4 The Overview window has a big Mounted button

If you use the Mounting Assistant on a tablet or any type of touch screen, you can done touch the Mounted button at the far right of the Overview window.

11.5 Wire numbers in the wire list

Wire numbers have their own column in the list.



11.6 Export to Excel

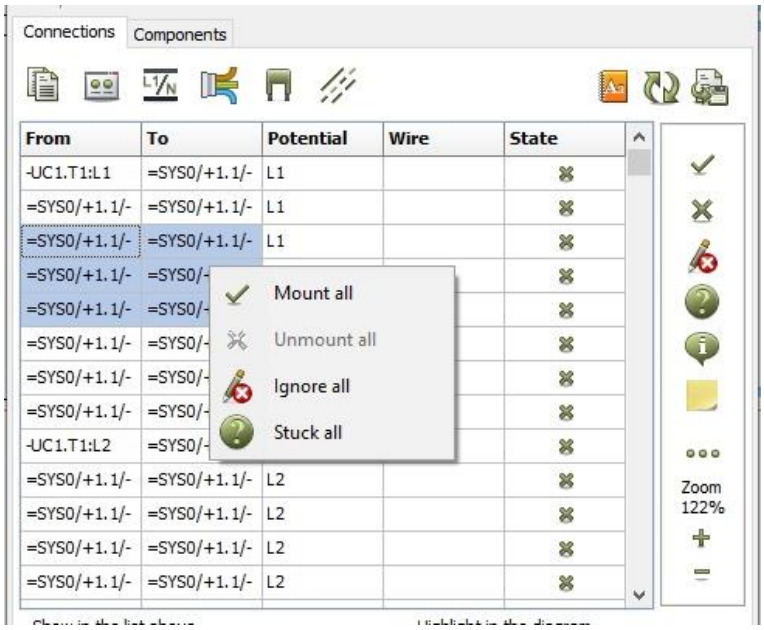
You can export directly to Excel from the Connections and the Component tabs.

You export the selected list – 'What you see is what you get'.[xxvi]

11.7 Possible to change status on more items in one operation

When you rigthclick you can change status on all connections on one operation.

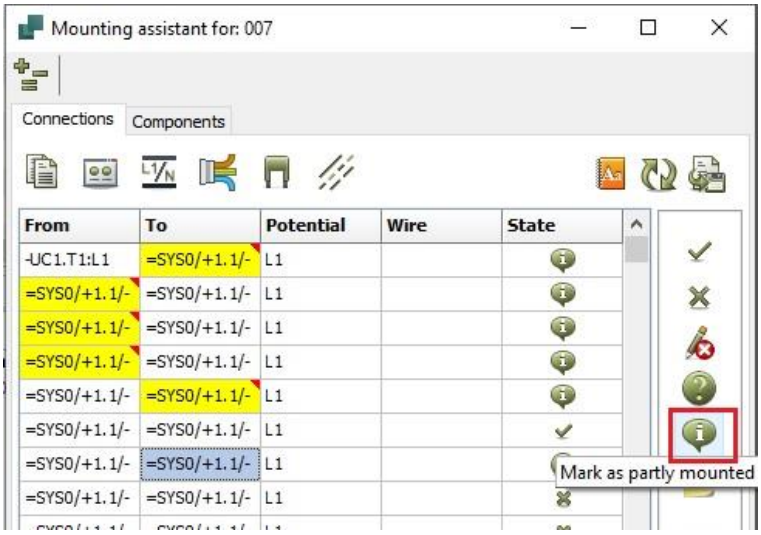
There is a similar function on the Components tab.



11.8 Connections can be partly mounted

When you work on a large project, you will sometimes need to only mount one end of the connection.

In the list of connections, you can select the From or To end, and mark it as Partly mounted.



12 MORE FUNCTION IN THE OBJECT LISTER (F7)

The Object Lister is a tool that can be used in many ways in the program. That also means, that some functionality is improved on request.

12.1 Object lister – new default setup *

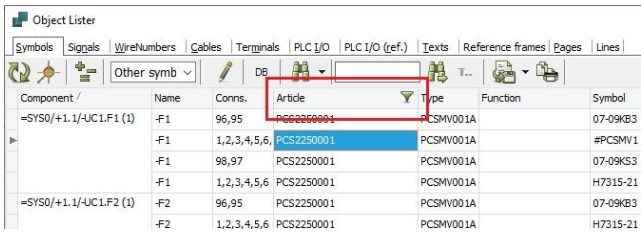
When you install Automation (not update), the Object lister has had a new default setup.

The setup includes a Component function – see the two pictures on this page.

Remember, you can always make your own setup: Rightclick in the headline and select the columns you want, and drag them to the position you want.

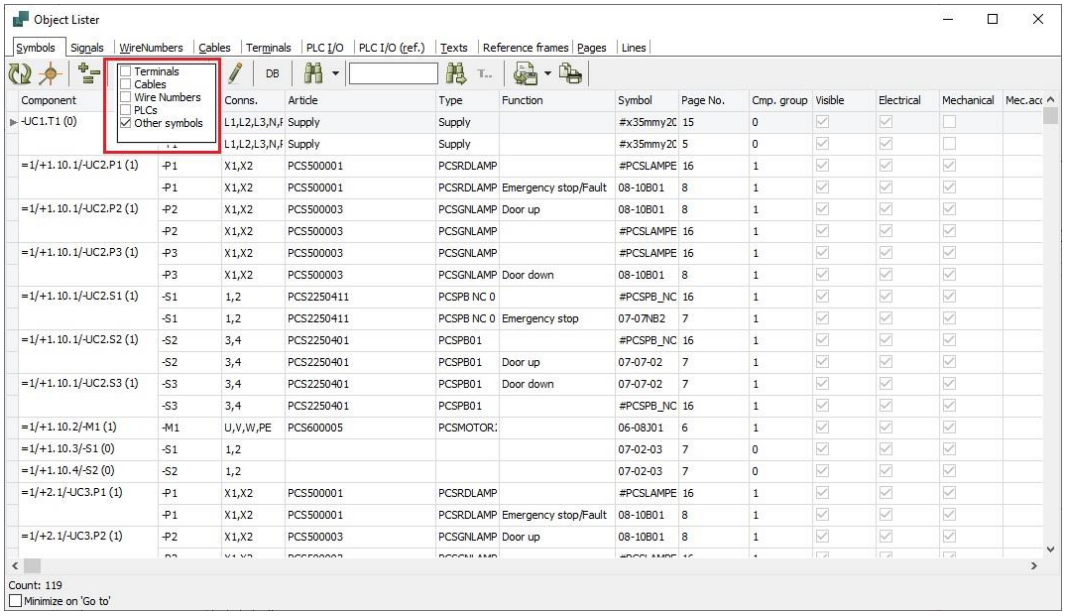
12.2 The Object lister has a filter in all columns

Youcan set a filter in all columns in the Object lister. It works in the same way as in eg. Excel, so type a part of the contents in a columnm and see only the objects that has this part.



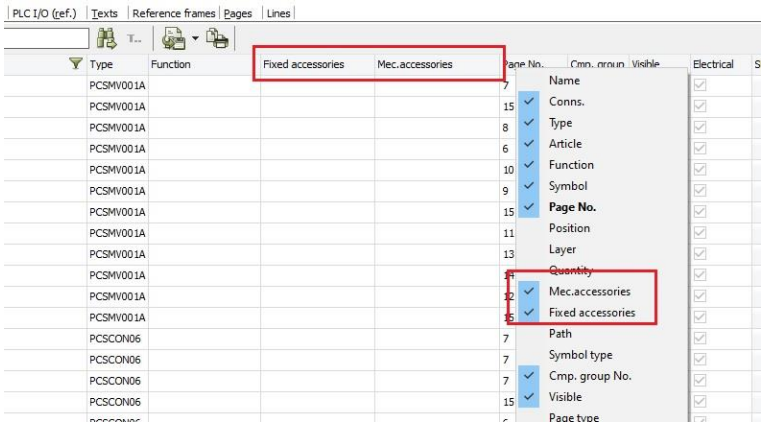
12.3 Object lister – all symbol types can be shown on the symbol tab *

You can choose to show all symbol types on the symbol tab, which means that info on eg terminals and cables can be shown together with info on 'ordinary' symbols.



12.4 The Object lister can show accessories

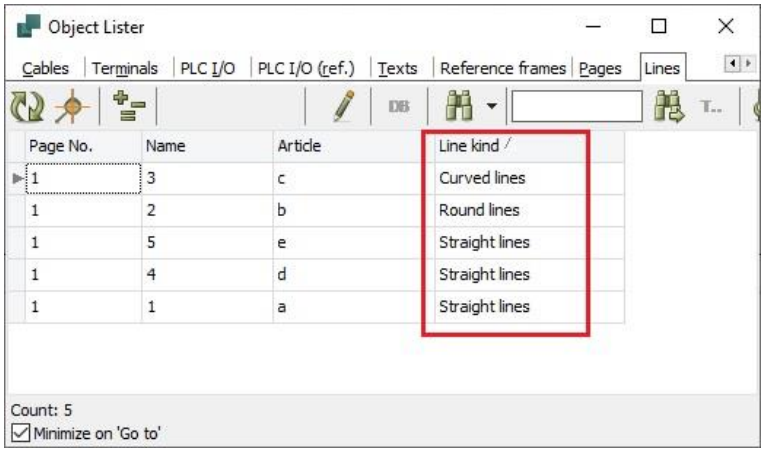
The Object lister can show all mechanical and fixed accessories.[xxvii]



12.5 Objektlister kan vise stregart

The Object lister can show the line kind, which will make it easy to find 'clouds'.

Clouds are round lines. [xxviii]



13 EXTENSION TO SYMBOL DATA FIELDS

13.1 New fixed symbol data field

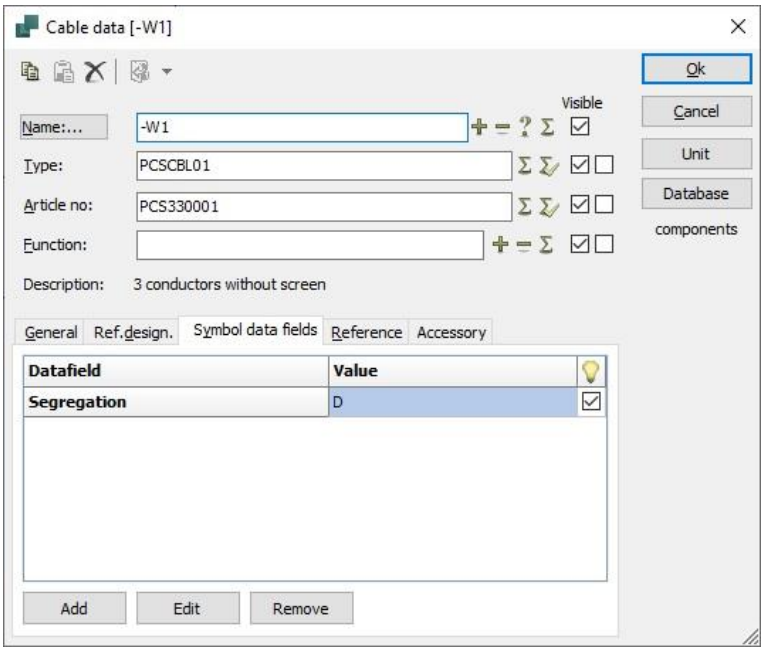
The data field Segregation is a fixed data field in all new installations. The data field is intented to be added to all cables, that is to be exported to Cablemanager.

A segregation is a name / code for the track in the cableway.

See also the section about export to the Cablemanager on page

34.

You assign the field to the desired cables.

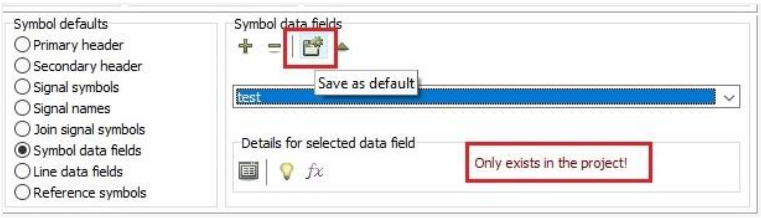


13.2 Symbol data fields are by default only in the current project

When you create a symbol data field from now on, it will only be present in the current project, and not in all projects, as it has been until now.

If you want to have the data field in all projects, simply save it as

Oprettelse af symboldatafelter er nu pr projekt og ikke som tidligere i hele programmet!



Hvis du ønsker at feltet skal være der altid, så skal du klikke på ikonet Gem som standard.

13.3 Symbol data fields with formula

Symbol data fields can have formulars in the same way as line data fields.

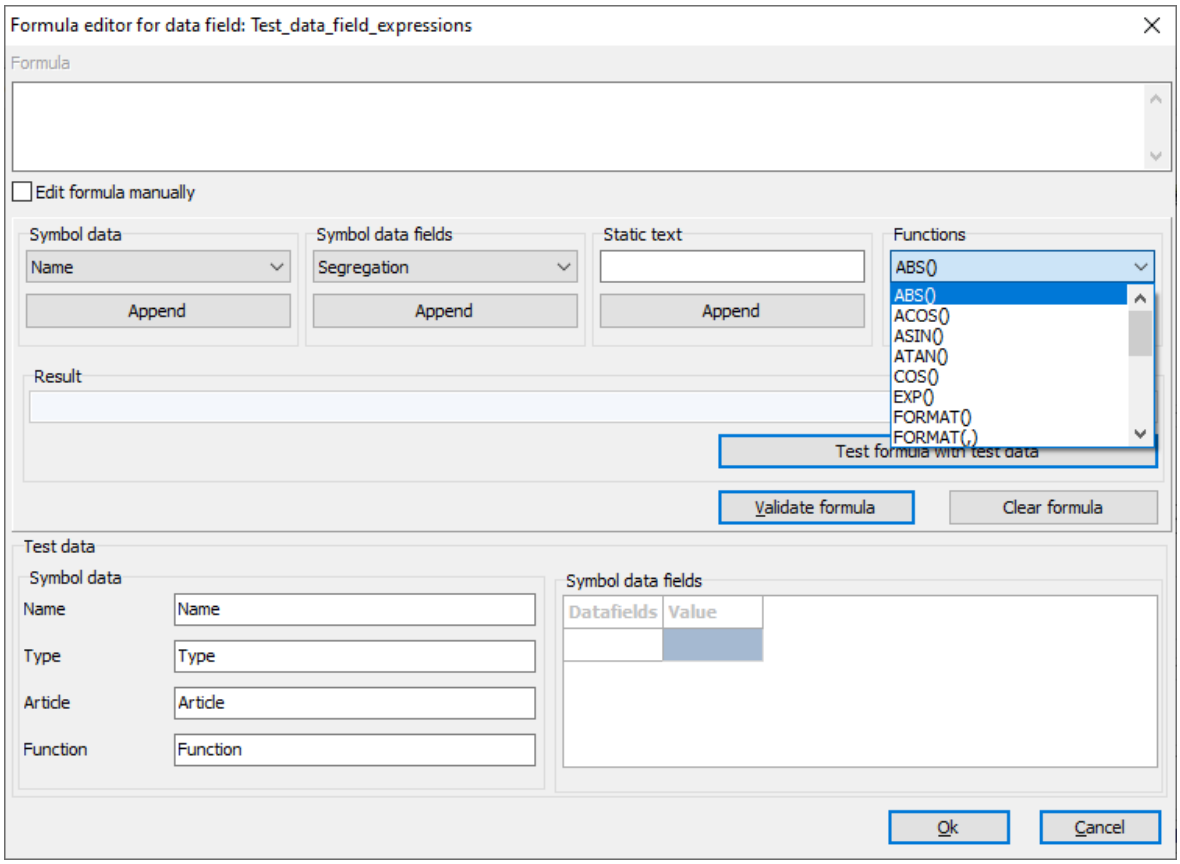
Formulars are made up of Symbol data (eg Name, Type) your own Symbol data fields and Static texts.

The contents of the data field is treated like all other texts, which means that you can control font, color, adjustment etc..

13.4 New formular editor for line and symbol data fields

In case you need to make logical, mathematical expression in your line and symbol data fields, we have now made a Formula editor for this.

In case you need the full documentation for the function, please refer to the last chapter in this document.



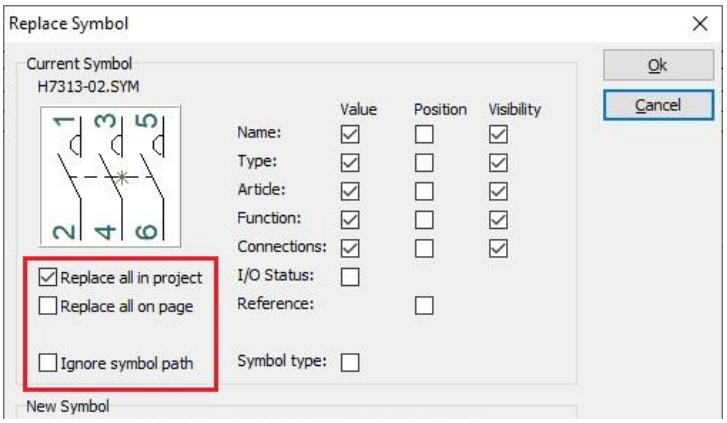
14 MISCELLANIOUS NEWS AND IMPROVEMENTS

Also this year, a collection of minor news and changes in the program.

14.1 Change Symbol med 'Ignore symbol path'

When you select All in project or All on page, you get the option to 'Ignore symbol path'.

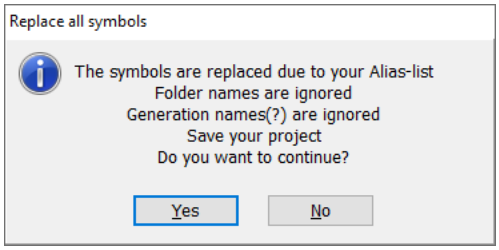
That means that the function replaces all occurences of the symbol, no matter where it originates from.



14.2 Replace all symbols in the project

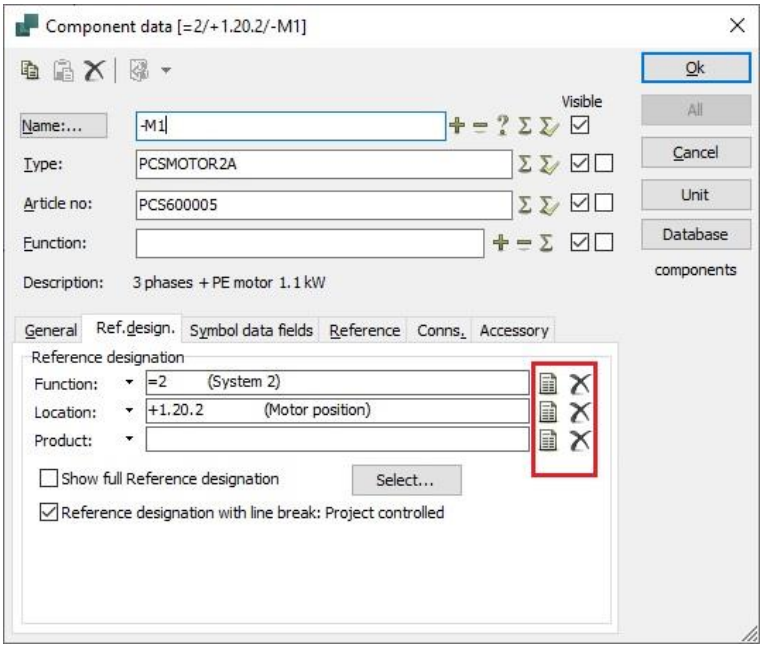
A small repetition of a function that was launched ver 21:

You can replace all symbols in a project in one operation. The function finds the symbols according to your ALIAS list.



14.3 Select ref.designations with As page and Delete button *

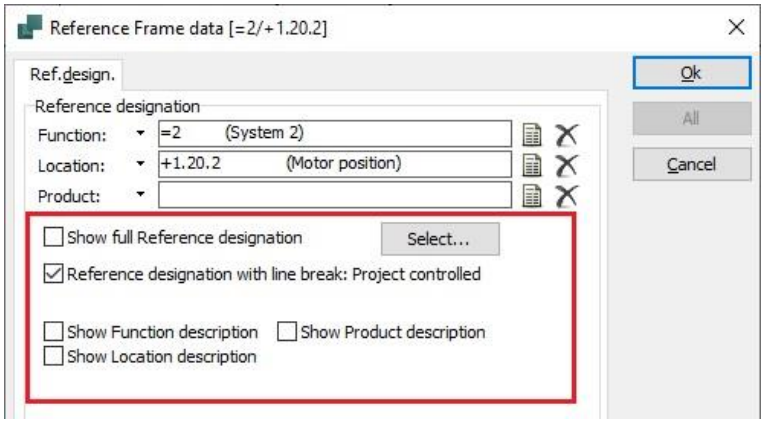
In order to make it easier to select reference designations on components, we have added an 'As page' button and a 'Delete' button on the Ref.designation tab. [xxix]



14.4 Settings for Insert ref.frame are saved *

Reference frames also have an 'As page' and a 'Delete' button.

As an improvement, the last selection of settings in the red frame are saved, and will be used for the next frame you insert. [xxx]



14.5 Export to DWG and DXF

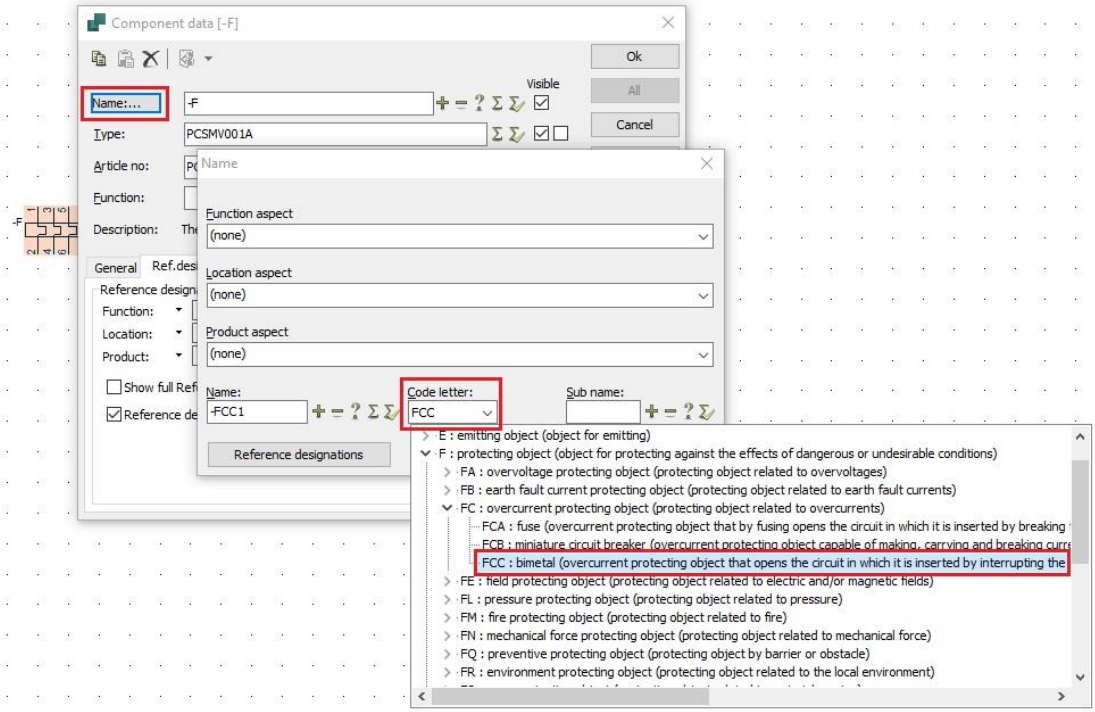
When you export to DWG or DXF you can now call your layer setting, so that you can change those before the export. [xxxi]

14.6 New letter codes from 81346-2 are in the program

On the Name tab, you can now find a list if all letter codes from IEC ISO 81346-2 which are used for component designations in the projects.

The new edition of the standard contains 1, 2 and 3 letter codes, and they are all on the list.

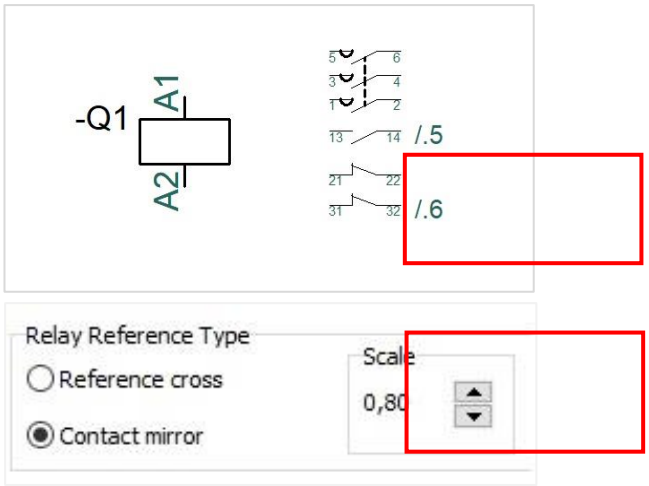
When you place a symbol, which designation is e.g. F, then the list will open from 'F'. If you select the code FCC the program will – as always – help you find the next available number.



14.7 Improved contact mirror

The contact mirror – another kind of reference cross – now uses the same font and color as other connections and cross references. Above, I have moved the reference symbol (right-click command).

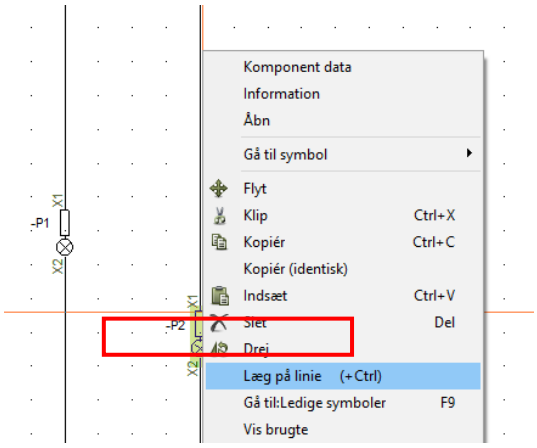
It is possible to control the size of the letters and numbers in Project data.



14.8 More functionality to the Align function *

The Align function can now align symbols on the same page and across pages:

1. Select the symbol with the right position, that makes it to the symbol that 'decides'.
2. Now select the other symbols that are to be aligned. The symbols don't need to be on the same page; the program remembers the position – the 'pointing line' occurs on the page.
3. The symbols will align wiht the first selected symbol.



Align can also be made in this way:

1. Select the symbol that is going to be aligned.
2. Now Ctrl+click on the symbol that has the right alignment – the one that 'decides'.
3. The first selected symbol now moves and aligneds with the last symbol.

Ctrl+click only works on the same page and not across pages. [xxxiii]

Remember: The Align function always finds the easiest way to alighn – vertical or horisontal.

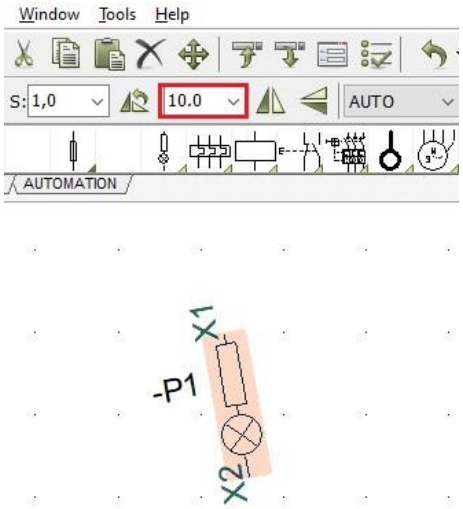
14.9 Design of symbol with the Symbol generator

When you type width and height for a new symbol, the preview window is updated dynamically.[xxxiv]

14.10 Rotate an object with 10° *

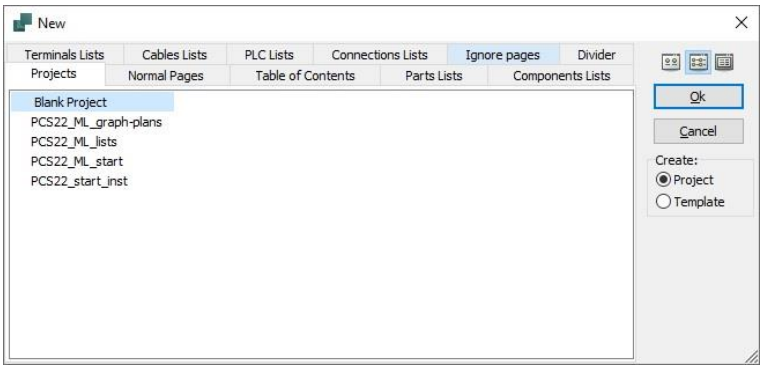
When you select an object and press Space, the object rotates 90 °.

If you hold down Ctrl and Space the object rotates with 10 °. [xxxv]



14.11 The icon New and Files|New has the same function *

When you press the icon you open the dialog with all templates. Same function as with Files|New. [xxxvi]



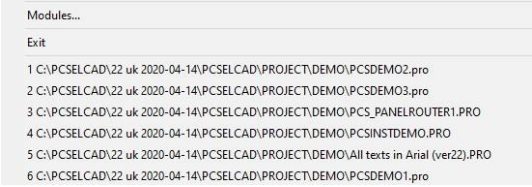
14.12 The length of the list of last opened files can be changed *

You can change the length of the list.

Go to the ini-file, and in the section [SystemData], you can write:

MaxPickFiles=20 to get a list of 20 files.

[xxxvii]



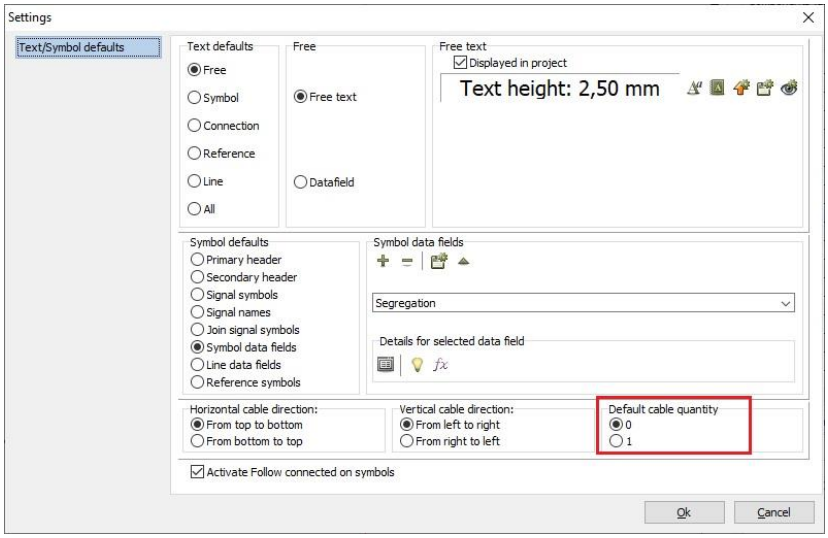
14.13 Default cable quantity can be changed *

It is possible to change the default cable quantity.

The setting is a system settings, which means that it follows the program not the project.

The selected setting is valid for all new cables that are added to the project.

Nothing is changed in existing projects. [xxxviii]

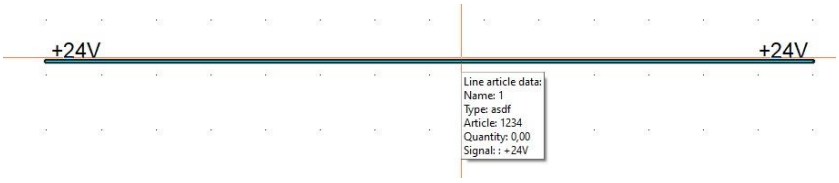


14.14 Lines with article data

You can draw a line with article data (if you pick a line with data from the pickmenu).

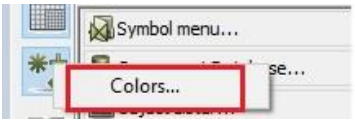
If you later makes this line longer by drawing another line at the end of it – a line with no article data – then the resulting line will inherit the line article data from the original line. In previous versions, the visuallly single line consisted of two segments, which created trouble for the Panelrouter.

The line type and color must be the same to make this function work, ie red line, conducting, same line type.



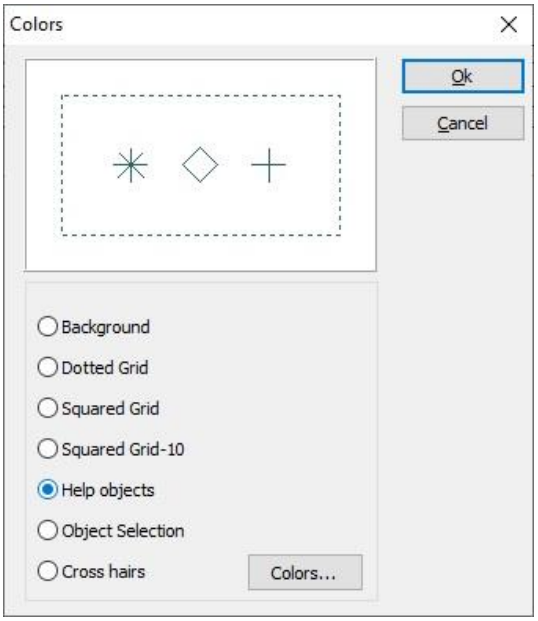
14.15 Shortcut to color settings from the vertical toolbar

When you rightclick on the icon for ref.symbols, you can select Colors ...



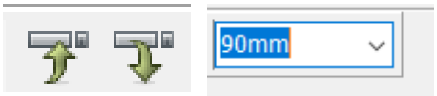
The menu opens the basic color setup, making it fast to change colors in the program.

The function is also available in the Automation Service program. [xxxix] [xl]



14.16 Height can be assigned by the Copy/Transfer properties icons

If you have placed a component in a certain height on a mechanical page, it is now possible to change the height by using the properties icons.



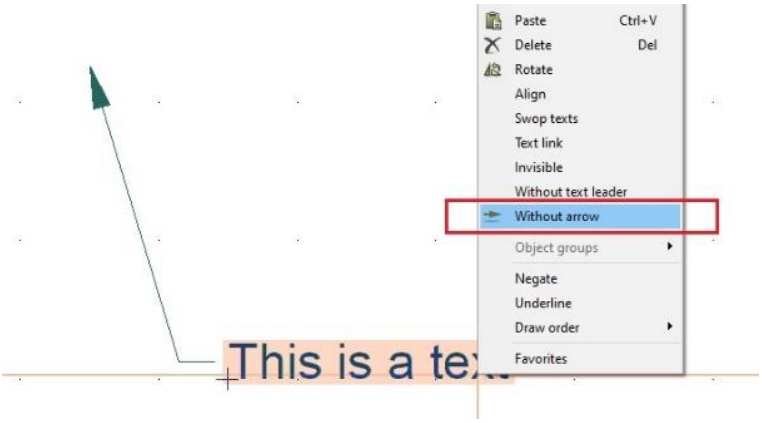
Ealier, you needed to Move the symbols to another height.

14.17 Text properties

Windows fonts kan now change height and width.

14.18 Leaders – with or without arrow

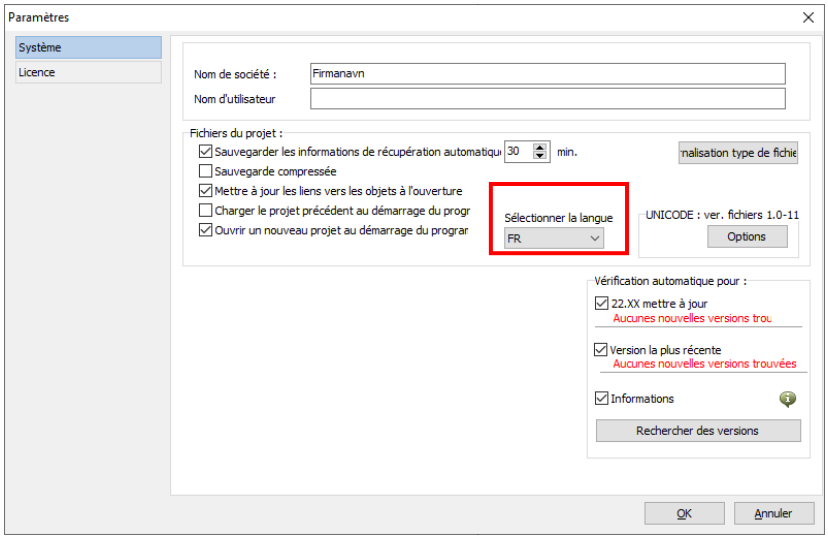
You can deselect the arrow from the text leader.



14.19 User interface is now also in French

The user interface is now also supported in French.

'Our' drawing header, however, has not yet been translated, but if someone out there could help ...



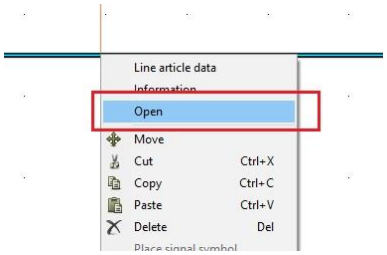
14.19.1 Drawing headers are now also in Croatian

A very big thank you to one of our customers for helping 😊

14.20 Direct access to article data in right-click menu

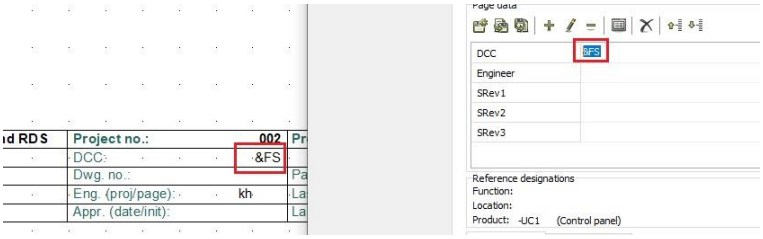
When you right-click on a symbol (old function) or on a line with article data (new function), you can open article data in either a unit drawing or a record in the database.

The function first searches in the project (unit drawings) then in the database.[xli]



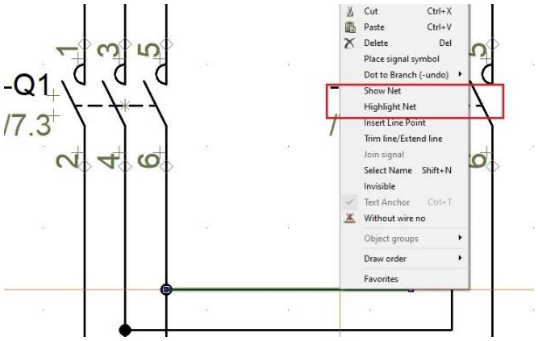
14.21 Double click in the drawing header opens Project or Page data

If you Hvis du dobbeltklikker on a link in the drawing header, project or page data (which ever is relevant) os automatically opened, and the selected data field is in focus.[xlii]



14.22 Cleaning out superflous menu items and functions

The menu items Show net and Highlight net have been deleted from the right-clcik menu. Simply, because the same functions are in the Netnavigator, and it is also possible to print the selected netlist from the Netnavigator.[xliii]



15 FROM OLD TEXT ADJUSTMENTS TO NEW ONES

We have converted text adjustments in the following way:

We have moved all +'s to the top, and the text are placed in the same position as before.

Connection names are still adjusted at the buttom, as the must adjust with the line on which they are standing,

If your lists have different text heights, this means that you might have to adjust the list, as the data fields are no longer aligned.

We have made some finer adjustments, so that you only very rarely need to adjust the list symbol itself.

15.1 Load of list pages

When you design a list (drawing header), you must follow some rules:

- All data fields for one set of data MUST be aligned (in the examples below, a data set is different data for the same name: article no., type, manufacturer etc.)
- The data fields are aligned when the +'s are aligned
- The first data fields must have the property 'Activate next)
- Data fields can be from the project and from the database

Read here, if you want the old list to work as before:

See the +'s in the picture: This is the old list and at a first glance it works fine (the +'s are at the bottom of the data field).

The list designer thought that data could be on one line only. If the data is more than one line, then contents will move upwards – because the + is at the bottom.

That's why we move all +'s to the top: we want multi-line data to grow downwards.

Name	Article No.	Type	Function
Name	Article No.	Type	Function
Name	Article No.	Type	Function
Name	Article No.	Type	Function
Name	Article No.	Type	Function
Name	Article No.	Type	Function
Name	Article No.	Type	Function

Name	Article No.	Type	Function
Name	Article No.	Type	Function
Name	Article No.	Type	Function
Name	Article No.	Type	Function

16 FORMULA EDITOR FOR LINE AND SYMBOL DATA FIELDS

16.1 General

An expression may consist of constants (texts, numbers or Boolean constants), variables and functions.

Supported are, arithmetic operators, boolean operators, comparison operators, equality operators, and string operator.

All function names, variable names, constant names and operators are case insensitive.

Please note that the expression must return a text value to be valid.

In order to do expressions with arithmetic sub expression, where data fields are included, one must convert the content of the data field to a numeric value, before it could be used in a sub arithmetic expression.

E.g. if the data field named: ‘Diameter’ contains the text: '4', this text value must be converted to a number before using it in a sub arithmetic expression.

So, a sub arithmetic expression like:

3.141 * VAL(DATAFIELD('Diameter'))

is valid, but:

3.141 * DATAFIELD('Diameter')

is invalid since one cannot multiple 3.141 with the text '3'.

Since the result of a data field expression must be a text, all results of sub arithmetic expressions must be converted from numbers to texts, if they are going to be part of the result.

So, a data field expression like:

' Area: ' + FORMAT(POW(VAL(DATAFIELD('Diameter')) / 2, 2) * 3.141) + ' mm2'

is valid, but:

'Area: ' + POW(VAL(DATAFIELD(‘Diameter’)) / 2, 2) * 3.141 + ' mm2'

is invalid, since only element of same type could be added together.

Predefined Boolean constants:

TRUE : Boolean true value

FALSE : Boolean false value.

Character strings must be encapsulated be the character: ' if a ' is to use in a text it must be done by specifying two ' in sequence.

So, a result test like: Don’t do this should be specified as: 'Don''t do this'

Numbers are implemented as floating point value. The decimal separator must be . (a period).

The range support is: -1.79e-308 to -1.79e+308, with 15 to 16 significant digits.

16.2 Operators

Operators behave like predefined functions that are part of the data field expression language.

For example, the expression: 1 + 2, is build from two number: 1 and 2 and the + operator.

This expression would return the number 3.

All operators except +, -, and not are demanding two operands, the + and – and not may be used as unary operator in from of expressions or numbers.

So: -(7 + 1) would return the value --8 and not (1 < 2) would return false.

To prioritize certain section of an equation one may use (..) to encapsulate sections which needs higher priority.

So, an expression like 1 + 2 * 3 would result in the number 7, where as (1 + 2) * 3 would result in the number 9.

So, for expressions with arithmetic operators: * and / are always solved before: + and – operators.

Boolean operator and is solved before xor which is solved before or.

The operator groups are solved in the following sequence: arithmetic operators, comparison operators, equality operators and last Boolean operators.

16.2.1 Arithmetic operators:

Operator	Operation	Operand type	Example
+	Addition	Number	1.2 + 5
-	Subtraction	Number	3.5 -2.1
*	Multiplication	Number	1.21E3 * 12
/	Division	Number	355 / 113

Example for addition (+):

FORMAT(VAL(DATAFIELD('Custom 1')) + VAL(DATAFIELD('Custom 2')))

The calculation result will be displayed as text in the data field where this formula is created for.

Example for subtraction (-):

FORMAT(VAL(DATAFIELD('Custom 1')) - VAL(DATAFIELD('Custom 2')))

The calculation result will be displayed as text in the data field where this formula is created for.

Example multiplication (*):

FORMAT(VAL(DATAFIELD('Custom 1')) * VAL(DATAFIELD('Custom 2')))

The calculation result will be displayed as text in the data field where this formula is created for.

Example division (/):

FORMAT(VAL(DATAFIELD('Custom 1')) / VAL(DATAFIELD('Custom 2')))

The calculation result will be displayed as text in the data field where this formula is created for.

16.2.2 Boolean operators:

Operator	Operation	Operand type	Example
not	Negation	Boolean	Not A
and	Conjunction	Boolean	A and B
or	Disjunction	Boolean	A or B
Xor	Exclusive disjunction	Boolean	A xor B

Expressions with Boolean operators **and** and **or** are always short circuited.

So, an **AND** expression like: (7 < 2) and (3 <4) would only solve the first part (1 < 2) and since this is false it would skip the remaining part.

Likewise, an **OR** expression like: (7 > 3) OR (1 <2) would only solve the first part (7 < 3) and since this is true it would skip the remaining part.

16.2.3 Comparison operators:

Operator	Operation	Operand type	Result type	Example
>	Greater than	String or Number	Boolean	233 > 122
>=	Greater than or equal to	String or Number	Boolean	‘ABC’ >= ‘ABCD’
<	Less than	String or Number	Boolean	Name < ‘HELLO’
<=	Less than or equal to	String or Number	Boolean	VAL(‘1212’) <= 2

Both side of a comparison operator must be of same type.

16.2.4 Equality operators:

Operator	Operation	Operand type	Result type	Example
=	Equality	String or Number	Boolean	233 = 122
<>	Inequality	String or Number	Boolean	‘ABC’ <> ‘ABCD’

Both side of an equality operator must be of same type.

16.2.5 String operator:

Operator	Operation	Operand type	Example
+	Concatenation	String	‘Hello ‘ + ‘world’

16.2.6 Variable:

The number of variables is fixed to 4 predefined variables.

For a symbol expression it is the default symbol fields (Name, ...).

For a line expression it is the default line fields (Name, ...).

In the formula editor one can select the valid ones from the combo box in the group Symbol data for symbol expression or Line data for line expression.

16.3 Functions:

The expression editor supports the following functions:

16.3.1 Function: DATAFIELD

Returns a text value of a used defined data field.

Syntax: DATAFIELD(‘name’), where name is a text constant.

Example: If a data field named Diameter contains the value text 123, the DATAFIELD(‘Diameter’) would return the text value 123.

The name of the data field is case insensitive.

Exceptions: If the name is not a valid data field name an Undefine data field error will returned.

16.3.2 Function: VAL

Returns the number of a text value if possible.

The VAL function accepts either comma or point as decimal separators, but not any thousand separators. The exponential notation is support as well.

Syntax: VAL(value), where value is a string.

Examples:

- VAL('-123,3') or VAL('-123.3') would both return the number -123.3
- VAL(DATAFIELD('Pressure')) would, if the value of data field 'Pressure' is the text 16 return the number 16.

- VAL(DATAFIELD('Pressure')) would, if the value of data field 'Pressure' is the text 16bar raise a Not a valid number exception.

Exceptions: If the specified text value is not a valid number or is out of supported number range.

16.3.3 Function: ISVAL

Returns a boolean true value if the value could be converted to a number and a Boolean false value otherwise.

The ISVAL function accepts either comma or point as decimal separators, but not any thousand separators. The exponential notation is support as well.

Please notice that the result of this function can only be used in Boolean sub equations.

Boolean values cannot be converted to texts.

Syntax: ISVAL(value), where value is a string.

Examples:

- ISVAL('-123,3') would return a Boolean true value, but ISVAL('HELLO') would return Boolean false value.
- ISVAL(DATAFIELD('Pressure')) would, if the value of data field 'Pressure' is the text 16, return a logical true value.
- ISVAL(DATAFIELD('Pressure')) would, if the value of data field 'Pressure' is the text 16bar, return a logical false value.
- IF(ISVAL(DATAFIELD('Pressure')), 'A number', 'Not a number'), would, if the value of data field 'Pressure' is the text 16, return 'A number'.
  If the value of data field 'Pressure' is the text 16bar, the result would be 'Not a number'.

Exceptions: None

16.3.4 Function: FORMAT

Returns a string with the specified format.

Syntax: FORMAT(number) or FORMAT(format, number), where number is the number to format and format is a string with the desired format.

If the format is not included, the format is %g

The allowed format is:

```
"%" ["-"] [width] ["."] prec] type
```

A format specifier begins with a % character. After the percent sign come the following elements, in this order:

1. An optional left justification indicator, ["-"]

2. An optional width specifier, [width].
3. An optional precision specifier, ["." prec].
4. The conversion type character, type.

The following types are possible:

Value	Meaning
e	Scientific The argument must be a floating-point value. The value is converted to a string of the form "-d.ddd...E+ddd". The resulting string starts with a minus sign if the number is negative. One digit always precedes the decimal point. The total number of digits in the resulting string (including the one before the decimal point) is given by the precision specifier in the format string; a default precision of 15 is assumed if no precision specifier is present. The "E" exponent character in the resulting string is always followed by a plus or minus sign and at least three digits.
f	Fixed The argument must be a floating-point value. The value is converted to a string of the form "-ddd.ddd...". The resulting string starts with a minus sign if the number is negative. The number of digits after the decimal point is given by the precision specifier in the format string—a default of 2 decimal digits is assumed if no precision specifier is present.
g	General The argument must be a floating-point value. The value is converted to the shortest possible decimal string using fixed or scientific format. The number of significant digits in the resulting string is given by the precision specifier in the format string; a default precision of 15 is assumed if no precision specifier is present. Trailing zeros are removed from the resulting string, and a decimal point appears only if necessary. The resulting string uses the fixed-point format if the number of digits to the left of the decimal point in the value is less than or equal to the specified precision, and if the value is greater than or equal to 0.00001. Otherwise the resulting string uses scientific format.

Examples:

- FORMAT(123.3) would return the string 123,3 or 123.3 depending on the region settings in windows.
- FORMAT(‘%.4f’, 123.3) would return the string 123,3000 or 123.3000 depending on the region settings in windows.
- FORMAT(‘%.0f’, 123.5) would return the string 124.
- FORMAT('%e', 122.5) would return the string 1,22500000000000E+002 or 1.22500000000000E+002 depending on the region settings in windows.

Exceptions: None

16.3.5 Function: EXP

Returns the exponent of the value where the base is **e**.

Syntax: **EXP(value)**, where value is a number.

Example:

- **EXP(1)** would return the number **2.71828182845905**
  If the result is too big, an infinite number would be returned.
- **FORMAT(EXP(VAL(DATAFIELD('Diameter'))))**
  The value of data field 'Diameter' will be used as base of **e**.
  The calculation result will be displayed as text in the data field where this formula is created for.

Exceptions: None

16.3.6 Function: POW

Returns base raised to the exponent.

Syntax: **POW(base, exponent)**, where base and exponent are both numbers.

Example:

- **POW(10, 2)** would return the number: 100
- **FORMAT(POW(VAL(DATAFIELD('Diameter')), VAL(DATAFIELD('Custom 1'))))**
  For base the value of data field 'Diameter', and for exponent the value of data field 'Custom 1' will be used. The calculation result will be displayed as text in the data field where this formula is created for.

Exceptions: If the result is too big, a positive or negative infinite number would be returned, based on the sign of the base.

16.3.7 Function: SQRT

Returns the square root of the value.

Syntax: **SQRT(value)**, where value is a number.

Example:

- **SQRT(81)** would return the number: 9
- **FORMAT(SQRT(VAL(DATAFIELD('Diameter'))))**
  Here, the value from data field 'Diameter' will be used for calculation of square root.

- FORMAT(SQRT(VAL(DATAFIELD('Diameter'))) * VAL(DATAFIELD('Custom 3')))
  Here, the result of SQRT from data field 'Diameter' is directly multiplied with value from data field 'Custom 3', and the calculation result is displayed as text in the data field where this formula is created for.

Exceptions: if value is negative a NaN (Not a Number) number would be returned.

16.3.8 Function: SIN

Returns the sine of the value in degrees.

Syntax: SIN(value), where value is a number.

Example:

- SIN(45) would return the number 0.707106781186547
- FORMAT(SIN(VAL(DATAFIELD('Angle'))))
  The value from data field 'Angle' will be used for sine calculation, and for display of calculation result as text in the data field where this formula is created for.

Exceptions: None

16.3.9 Function: COS

Returns the cosine of the value in degrees.

Syntax: COS(value), where value is a number.

Example:

- COS(45) would return the number 0.707106781186547
- FORMAT(COS(VAL(DATAFIELD('Angle'))))
  The value from data field 'Angle' will be used for cosine calculation, and for display of calculation result as text in the data field where this formula is created for.

Exceptions: None

16.3.10 Function: TAN

Returns the tangent of the value in degrees.

Syntax: TAN(value), where value is a number.

Example:

- TAN(45) would return the number 1

- FORMAT(TAN(VAL(DATAFIELD('Angle'))))
  The value from data field 'Angle' will be used for tangent calculation, and for display of calculation result as text in the data field where the formula is created for.

Exceptions: None

16.3.11 Function: ASIN

Returns the principal angle in degrees of the inverse sine of the value in the range -1 to 1, both included.

Syntax: ASIN(value), where value is a number.

Example:

- ASIN(0.707106781186547) would return the number 45
- FORMAT(ASIN(VAL(DATAFIELD('Angle'))))
  The value from data field 'Angle' will be used for inverse sine calculation, and for display of calculation result as text in the data field where the formula is created for.

Exceptions: If value is outside the range -1 to 1 a NaN (Not a Number) number would be returned.

16.3.12 Function: ACOS

Returns the principal angle in degrees of the inverse cosine of the value in the range -1 to 1, both included.

Syntax: ACOS(value), where value is a number.

Example:

- ACOS(0.707106781186547) would return the number 45
- FORMAT(ACOS(VAL(DATAFIELD('Angle'))))
  The value from data field 'Angle' will be used for inverse cosine calculation, and for display of calculation result as text in the data field where the formula is created for.

Exceptions: If value is outside the range -1 to 1 a NaN (Not a Number) number would be returned.

16.3.13 Function: ATAN

Returns the principal angle in degrees of the inverse tangent of the value.

Syntax: ATAN(value), where value is a number.

Example:

- ATAN(1E100) would return the number 90

Exceptions: None.

16.3.14 Function: ABS

Returns the absolute value of the value.

Syntax: ABS(value), where value is a number.

Example:

- ABS(-10) would return the number 10

Exceptions: None.

16.3.15 Function: LN

Returns the natural logarithm of the value.

Syntax: LN(value), where value is a number.

Example:

- LN(2.71828182845905) would return the number 1

Exceptions: if value is negative a NaN (Not a Number) number would be returned, if value is zero an infinite negative number would be returned.

16.3.16 Function: LOG

Returns the logarithm with base 10 of the value.

Syntax: LOG(value), where value is a number.

Example:

- LOG(100) would return the number 2

Exceptions: if value is negative a NaN (Not a Number) number would be returned, if value is zero an infinite negative number would be returned.

16.3.17 Function: TRUNC

Returns the truncated number.

Syntax: TRUNC(value), where value is a number.

Example:

- TRUNC(100.999) would return the number 100
- FORMAT(TRUNC(VAL(DATAFIELD('Custom 5')))) The value from data field 'Custom 5' will be used for truncation, and for display of result as text in the data field where the formula is created for.

Exceptions: None.

16.3.18 Function: ROUND

Returns the rounded value.

Syntax: ROUND(value), where value is a number.

Example:

- ROUND(100.5) would return the number 101
- FORMAT(ROUND(VAL(DATAFIELD('Custom 5')))) The value from data field 'Custom 5' will be used for rounding, and for display of result as text in the data field where the formula is created for.

Exceptions: None.

16.3.19 Function: IF

Return either the true or false result based on the condition

Syntax: IF(condition, true_expression, false_expression), where condition is a Boolean expression and true_value and false_value is either text or number values.

Examples:

- IF(ISVAL(DATAFIELD('Test1')) AND VAL(DATAFIELD('Test1')) > 100, 'Yes', 'No') would return the text value 'Yes', if the data field contains the text value 101, but if the data field contained the text value 100, the result would be the text value 'No'. Likewise the result would be 'No', if the data field contained a text value which could not be converted to a valid number, like e.g. 'hello'

If the order of the elements in the condition part is switched like:

- IF(VAL(DATAFIELD('Test1')) > 100 AND ISVAL(DATAFIELD('Test1')), 'Yes', 'No')
  the expression is only valid, if the data field 'Test1' contained a text value which could be converted to a valid number, like e.g. 123
  So if the data field contained the value 'ABC', the result would be !Invalid!, since the VAL(DATAFIELD('Test1')) function would throw a Not a valid number expression.
  This is not the case in the first example, since a logical AND operator is short circuited, when the first Boolean false result is met, and therefore the VAL(DATAFIELD('Test1')) part is never executed.

  This short circuit also applies for the IF function.
  So, if the condition is true, only the true_expression would be executed and if the condition is false, only the false_expression would be executed.
  So, if one wants to ensure that the value of a given data field is only used in an arithmetic sub expression, if it contains a valid text, which could be converted to a number, one could specify a data field expression like:
  IF(ISVAL(DATAFIELD('Test1')), FORMAT(VAL(DATAFIELD('Test1')) * 2), 'Invalid number in data field: Test1')
  So if the value of the data field named 'Test1' contains the text value 4, the result would be 4, and if the data field contained a text value which could not be convert to a number, the result would be 'Invalid number in data field: Test1'.

Exceptions: None.