User manual supplements - New features in ATPDraw v5

This document outlines some of the most important changes in ATPDraw version 4 and 5. The additionally required background document is the User Manual for version 3.5.

Hans Kr. Høidalen
Nov. 18th 2007, Trondheim-Norway

1. Introduction

ATPDraw is a graphical, mouse-driven preprocessor to the ATP version of the Electromagnetic Transients Program (EMTP) on the MS-Windows platform. The program is written in Borland Delphi 2007 and runs under Windows 9x/NT/2000/XP/Vista as well as under emulators in LINUX. In ATPDraw the user can construct an electrical circuit using the mouse and selecting components from menus, then ATPDraw generates the ATP input file in the appropriate format based on "what you see is what you get". The simulation program ATP and plotting programs can be integrated with ATPDraw.

ATPDraw supports multiple circuit modeling that makes possible to work on several circuits simultaneously and copy information between the circuits. All kinds of standard circuit editing facilities (copy/paste, grouping, rotate, export/import, undo/redo) are available. In addition, ATPDraw supports the Windows clipboard and metafile export. The circuit is stored on disk in a single project file, which includes all the simulation objects and options needed to run the case. The project file follows the PKZIP 2.0 standard format (zip-compressed) that makes the file sharing with others very simple.

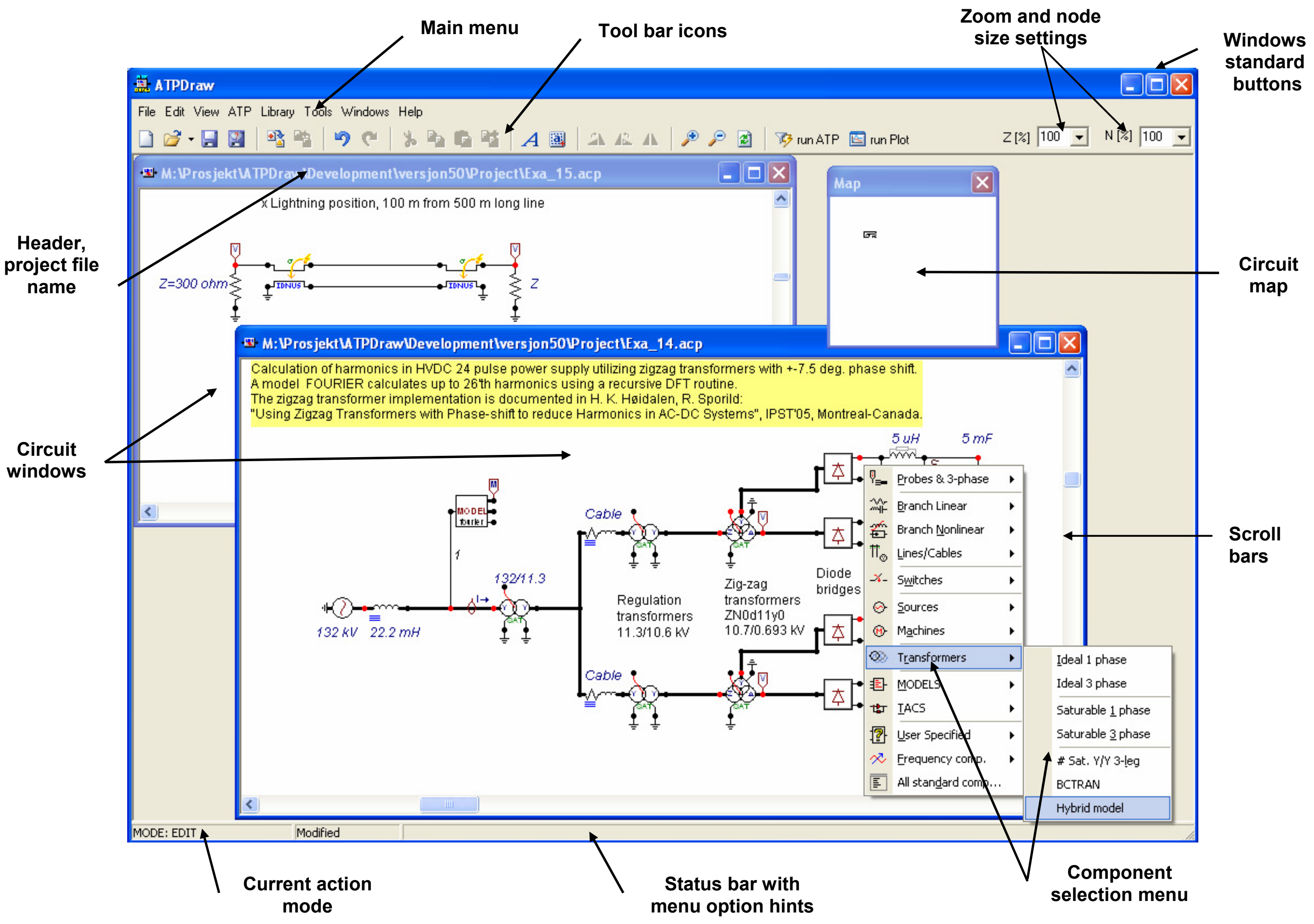
Most of the standard components of ATP (both single and 3-phase), as well as TACS are supported, and in addition the user can create new objects based on MODELS or $Include (Data Base Module). Line/Cable modeling (KCLee, PI-equivalent, Semlyen, JMarti and Noda) is also included in ATPDraw where the user specifies the geometry and material data and has the option to view the cross section graphically and verify the model in the frequency domain. Objects for Harmonic Frequency Scan (HFS) have also been added. Special components support the user in machine and transformer modeling based on the powerful Universal Machine and BCTRAN components in ATP-EMTP. A new advanced transformer component called Hybrid Transformer (XFMR) has recently been included.

ATPDraw supports hierarchical modeling by replacing selected group of objects with a single icon in unlimited numbers of layers. POCKET CALCULATOR and $PARAMETER features of ATP is also supported, allowing the user to specify a text string as input in a components' data field, then assign numerical global values to these texts strings later.

ATPDraw supports circuits with up to 10.000 components and connections, and components sizes of maximum 32 nodes (each with up to 26 phases) and 64 data. The bitmap icon can have a size of 41x41 pixels, while the vector graphics icon can be of size 255x255.

1.1 Operating windows

ATPDraw has a standard Windows user interface. This chapter explains some of the basic functionalities of the *Main menu* and the *Component selection menu* of the *Main window*.



Calculation of harmonics in HVDC 24 pulse power supply utilizing zigzag transformers with +-7.5 deg. phase shift.
A model FOURIER calculates up to 26'th harmonics using a recursive DFT routine.
The zigzag transformer implementation is documented in H. K. Høidalen, R. Sporild:
"Using Zigzag Transformers with Phase-shift to reduce Harmonics in AC-DC Systems", IPST'05, Montreal-Canada.

Fig. 1.1 - The Main and Circuit windows, Map, and the popup Component selection menu in version 5.3.

Fig. 1.1 shows the main window of ATPDraw containing two open circuit windows. ATPDraw supports multiple documents and offers the user to work on several circuits simultaneously along with the facility to copy information between the circuits. The size of the circuit window is much larger than the actual screen, as is indicated by the scroll bars of each circuit window. The *Main window* consists of the following parts:

Header + Frame:
As a standard Windows element, it contains the system menu on the left side, a header text and minimize, maximize, exit buttons on the right side. The main window is resizable.

System menu:	Contains possible window actions: Close, Resize, Restore, Move, Minimize, Maximize or Resize and Next. The last one exists only if multiple circuit windows are open.
Header text:	The header text is the program name in case of the main window and the current circuit file name in case of the circuit window(s). To move a window, click in the header text field, hold down and drag.
Minimize button:	A click on this button will iconize the main window.
Maximize button:	A click on this button will maximize the window. The maximize button will then be replaced with a resize button. One more click on this button will bring the window back to its previous size.

Corners:	Click on the corner, hold down and drag to resize the window.

Main menu:
The main menu provides access to all the functions offered by ATPDraw. The menu items are explained in detail in the Reference part of this Manual:

File:	Load and save circuit files, start a new one, import/export circuit files, create postscript and metafile/bitmap files, print the current circuit and exit.
Edit:	Circuit editing: copy/paste/delete/duplicate/flip/rotate, select, move label, copy graphics to clipboard and undo/redo etc.
View:	Tool bars, status bar and comment line on/off, zoom, refresh and view options.
ATP:	Run ATP, make and edit ATP-file, view the LIS-file, ATP-file settings (miscellaneous, file format, file sorting etc.), assign data to variables for $PARAMETER usage and specify RECORD for MODELS output requests.
Library:	Edit support files (default values, min/max limits, icon and help file), create new files for MODELS and User Specified Objects. Update (synchronize) the project icons.
Tools:	Icon editor, help file editor, text editor, setting of various program options.
Window:	Arrange the circuit windows and show/hide the Map window.
Help:	About box and ATPDraw help file system.

Zoom and node size:
In these menus you can type in zoom and node size in [%] or select predefined values in the popup box.

Circuit window:
The circuit is built up in this window. The circuit window is the container of circuit objects. From the *File menu* you can load (or import) projects from disk or simply create an empty window to start building a new circuit. When you start to use a new circuit you first of all have to specify a folder where the ATP results will be stored. This folder is called the *ResultDir* and its default value is set under *Tools|Options/Directories* as the ATP-folder. If several projects share the same *ResultDir* folder some complications can arise for components using $Include (LCC and User Specified), but data will never be lost. You need to have writing access to *ResultDir*. Circuit objects include standard ATP components, user specified elements, MODELS and TACS components, and connections. To move around in the circuit, you can use the window scrollbars, or drag the view rectangle of the *Map window* to another position.

Component selection menu:
This menu pops-up immediately when you click with the right mouse button in an empty space of the *Circuit window*. In this menu you select the circuit objects. After selecting an object in one of the sub-menus, the object is drawn in the circuit window in marked and moveable mode. Appendix A lists all available components in the component selection menu.

MAP window:
This window gives a bird's eye view of the entire circuit. The size of a circuit is 10000x10000 pixels (screen points); much larger than your screen would normally support. Consequently, the *Circuit window* displays only a small portion of the circuit. The actual circuit window is represented by a rectangle in the *Map window*.
Press and hold down the left mouse button in the map rectangle to move around in the map. When you release the mouse button, the circuit window displays the part of the circuit defined by the new rectangle size and position. The map window is a stay-on-top window, meaning that it will always be displayed on the top of other windows. You can show or hide the map selecting the *Map Window* option in the *Window* menu, or pressing Ctrl+*M* character,

Status bar - Action mode field:
The current action mode of the active circuit window is displayed in the status bar at the bottom of the main window, when the *Status Bar* option is activated in the *View* menu. ATPDraw can be in various action modes. The normal mode of operation is *MODE : EDIT* , in which new objects are selected and data are given to objects. Drawing connections brings ATPDraw into *CONN.END* mode and so on. ATPDraw's possible action modes are:

EDIT	The normal mode.
CONN.END	After a left click on a node, the action mode turns into *CONN.END* indicating that the program is waiting for a left mouse click to set the end-point of a new connection. To cancel drawing a connection, click the right mouse button or press the ESC key to return to *MODE : EDIT*.
EDIT TEXT	Indicates the text edit mode. It is used to add new text strings or select and move labels, node names, and texts overlapped by components. The mode is selected by *Edit*|*Edit text* or by the Toolbar icon. It is also activated if the user clicks in the circuit window while holding down the Alt key.
GROUP	Indicates region selection activated by *Edit*|*Select*|*Inside* or by double clicking the left mouse button in an empty space of the active circuit window. This enables you to draw a polygon shaped region. Click the left button to place corners and to close the selection area, click the right mouse button. Any objects within the selected region are then marked for selection. To cancel region selection, press the *Esc* key.
INFO.START	Indicates the start of a relation when *TACS* | *Draw relation* is activated in the selection menu. Clicking the left mouse button on a component node or on the end-point of another relation will initiate the drawing of a new relation. Relations are used to visualize information flow into FORTRAN statements and are drawn as blue connections, but do not influence the connections of components. (The process of drawing a *Relation* will probably soon disappear since this now can be handled with a normal connection instead and a property Relation.).
INFO.END	Indicates the end of a relation. The program is waiting for a left mouse button click to set the end-point of the new relation. To cancel drawing relation, click the right mouse button or press the *Esc* key.

Status bar - Modified and Hints field:

The middle field of the status bar is used to display the *Modified* state of the active circuit. As soon as you alter the circuit (moving a label, deleting a connection, inserting a new component, etc.), the text *Modified* appears, indicating that the circuit should be saved before exit. The field will be empty when you save the circuit or undo all modifications. The rightmost field of the status bar displays the menu option hints. If you check the *Drag over info* button in *View*|*Options* the *Hint* field will display information about the circuit component pointed on by the mouse. On large circuits this could slow down the response.

1.2 Operating the mouse

This chapter contains a summary of the various actions taken dependent on mouse operations. The left mouse button is generally used for selecting objects or connecting nodes; the right mouse button is used for specification of object or node properties.

<u>Left simple click:</u>

On component: Select the component.
    If the *Shift* key is pressed, the object is added to the current selection group.

On object node or connection: Begins to draw a connection.
    Move the mouse to the end node, left click to place, right to cancel.

In open area of the circuit window: Unselects object.

<u>Right simple click:</u>

In open area of the circuit window:
    Opens the *Component selection menu*, or
    Cancels the connection made if connection draw mode has been activated earlier.

On object node:
    Pops-up the *Node data* window.

On unselected object: Opens the *Component* dialog box.
    If *Shift* key is pressed simultaneously: opens the circuit window *Shortcut menu*.

On selected object(s): Rotates object(s).
    If *Shift* key is pressed simultaneously: opens the circuit window *Shortcut menu*.

<u>Left click and hold:</u>

On object: Moves the object or selected group of objects.
    If the *Alt* key is pressed, texts, labels and node names are preferred.

On connection: Select the connection.

On node: Resizes connection (it is often necessary to select connection first).

In open area of the circuit window: Draws a rectangle for group selection.
    Enclosed objects become members of the group when the mouse button is released.

<u>Left double click:</u>

On component node:
    Performs the *Node data* window.

On selected or unselected single component:
    Performs the *Component* dialog box.

On unselected connection:
    Perform the *Connection* dialog box.

On selected group of components:
    Performs an *Open Group* dialog box.

In open area of the circuit window:
    Starts the group selection facility. Click left to create an enclosing polygon, click right to close. Objects inside the polygon become a group.

1.3 Edit operations

ATPDraw offers the most common edit operations like copy, paste, duplicate, rotate and delete. The edit options operate on a single object or on a group of objects. Objects must be selected before any edit operations can be performed. Selected objects can also be exported to a disk file and any circuit files can be imported into another circuit. The Toolbar offers icons of frequent used actions. From version 5.3 the toolbar is customizable via *View|Toolbar Customize* followed by drag and drop operations from the appearing Customize dialog.

Tool bar	Shortcut key	Equivalent in menus
New	-	*File | New*
Open	Ctrl+O	*File | Open*
Save	Ctrl+S	*File | Save*
Save As	-	*File | Save As*
Import	-	*File | Import*
Export	-	*File | Export*
UNDO	Ctrl+Z	*Edit | Undo*
REDO	Ctrl+Y	*Edit | Redo*
Copy	Ctrl+C	*Edit | Copy*
Paste	Ctrl+V	*Edit | Paste*
Duplicate	Ctrl+D	*Edit | Duplicate*
Cut	Ctrl+X	*Edit | Cut*
Refresh	Ctrl+Q	V*iew | Refresh* (redraw the circuit)
Text	Ctrl+T	*Edit | Edit Text*
Select/All	Ctrl+A	*Edit | Select All*
Rotate R	Ctrl+R	*Edit | Rotate icon clockwise* (or right click)
Rotate L	Ctrl+L	*Edit | Rotate icon counter-clockwise*
Flip	Ctrl+F	*Edit | Flip icon left to right*
Zoom In/Out	NUM + / -	*View | Zoom In / Out*
Run ATP	*F2*	*ATP|run ATP*
Run Plot	*F8*	*ATP|run Plot*
Zoom	-	-
Node dot	-	-

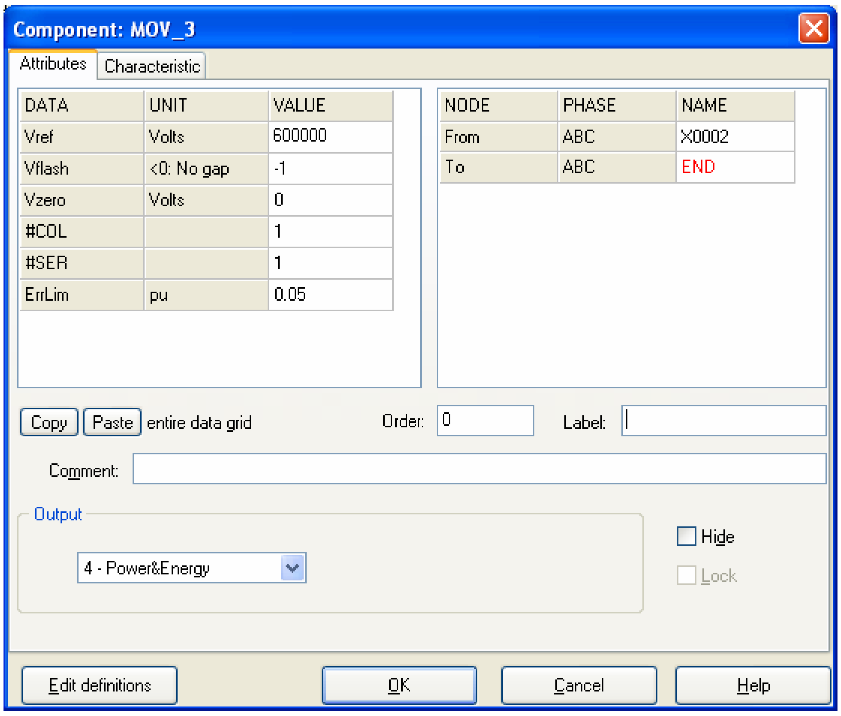
1.4 Overview of working with ATPDraw

After selecting a component in the *Component selection menu* the new circuit object appears in green in the middle of the circuit window enclosed by a rectangle. Click on it with the left mouse button to move, or the right button to rotate, finally click in the open space to unselect and place the object. Move to cursor to the icon border, click and hold to resize the icon, press ESC to reset the size.

To select and move an object, simply press and hold down the left mouse button on the object while moving the mouse. Release the button and click in an empty area to unselect and confirm its new position. The object is then moved to the nearest grid point (known as grid snapping). If two or more components overlap as a consequence of a move operation, you are given a warning message and can choose to proceed or cancel the operation. You are also warned about this in the process of making the ATP file and given the chance to find and select the overlapping objects in *Edit|Select|Overlapped.*

Selecting a group of objects can be done in three ways: 1) Holding down the *Shift* key while left clicking on the objects successively. 2) Pressing and holding down the left mouse button in an empty area enables the user to drag a rectangular outline around the objects of interest. 3) And finally, double-clicking the left mouse button in an empty area enables the definition of a polygon-shaped region by repeatedly clicking the left mouse button in the circuit window. To close the region, click the right mouse button. Objects that are defined to fall within the indicated region or rectangle become members of the group. For components this means that the centre point of a component icon must lie within the defined region or rectangle. For connections the region or rectangle must surround both end-points. To move the selected group of objects, press and hold down the left mouse button inside the group while moving the mouse. Unselect and confirm the new position by clicking in an empty area. Any overlapping components will produce a warning. To move objects outside of the visible part of the circuit, use the window scrollbars or the view rectangle in the map window. Any selected objects or group will follow the window to its new position. Objects or a group can be rotated by clicking the right mouse button inside the selected object or group. Other object manipulation functions, such as undo/redo and clipboard options can be found in the *Edit* menu. Additionally, the most frequently used object manipulation functions can be accessed by holding down the *Shift* key while clicking with the right mouse button on an object or on a selected group of objects. This will display and activate the circuit window shortcut menu.

Components and component nodes can be opened for editing by a right-click (or left double-click) on an unselected component, connection or node. The *Node data* or *Component* dialog box will appear, allowing the user to change object attributes and characteristics. The *Component* dialog box shown in Fig. 1.2 has the same layout for most circuit objects. In this window the user must specify the required component data. The number of DATA and NODES menu fields are the only difference between input windows for standard objects. The nonlinear branch components have a *Characteristic* page too, in addition to the normal *Attributes* page, where the nonlinear characteristics and some include file options can be specified. Some of the advanced components like transformers, lines&cables etc. have special dialog boxes for input.



DATA	UNIT	VALUE
Vref	Volts	600000
Vflash	<0: No gap	-1
Vzero	Volts	0
#COL		1
#SER		1
ErrLim	pu	0.05

NODE	PHASE	NAME
From	ABC	X0002
To	ABC	END

Order: 0

Label:

Comment:

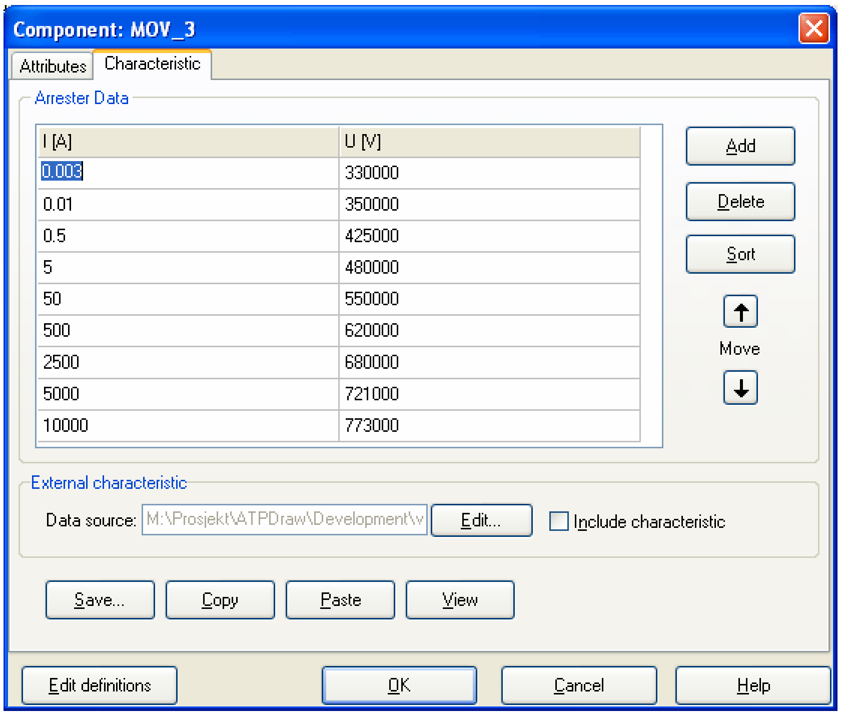
Output: 4 - Power&Energy

Fig. 1.2a) Component dialog box, attributes page.

The Component dialog box shown in Fig. 1.2a) consists of a Data part and a Node part. In the Data part the user can specify values using '.' as the decimal symbol. A variable name (6-char text string) can also be specified and given a global value under *ATP|Settings/Variables*. Specifying a variable is only possible if the *Param* property in the definitions is set to unity. The *Copy/Paste* buttons allows copying the entire data set via the Windows clipboard. Node names (6 or 5 characters) can be specified in the right grid. Node names drawn in a red color are already given a name by the user while black names are inherited. If the user wants to change a node name the red names/nodes should be preferred, otherwise name conflict warnings could appear. Node data are also given in the Node dialog box by clicking on the nodes. Multi-phase nodes can only take a 5 character name, and the phase sequence extension A..Z is added automatically.

Order is optionally used for sorting (*ATP|Settings/Format*; sorting by order), *Label* is a 12 character text string on screen, and *Comment* is a line of text written to the ATP file in front of the component's cards. The *Output* panel varies somewhat between components, but is usually used for branch output requests. In the lower left corner the icon seen in previous versions is replaced by *Edit definitions*. This gives access to all the local properties inherited from the support file, including the icon, local help, names of nodes and data, node positions, default values, param flags, limits, and units.

Nonlinear components also have a Characteristic page as shown in Fig. 1.2b). The user can add points manually by first clicking the *Add* button and then enter data in the string grid. Data can also be pasted in from the clipboard via the *Paste* button. It should be possible to select a two column characteristic in any text editor and copy it to the clipboard. The user can also choose to add an external characteristic from a file. This is done via the Edit button which will bring up a text editor where the user then can open a file or paste text in. The *Include characteristic* button has to be checked in this case. The *Data source* name is just for information of the original source of the external characteristic, since all data are stored locally in memory.



I [A]	U [V]
0.003	330000
0.01	350000
0.5	425000
5	480000
50	550000
500	620000
2500	680000
5000	721000
10000	773000

Data source: M:\Prosjekt\ATPDraw\Development\v

Fig. 1.2b) Component dialog box, characteristic page.

The nodes given a name by the user are marked as red. There is no need to give names to all nodes, only those of particular interest (for monitoring). ATPDraw will give the same name to all connected nodes automatically. A node can hold up to 26-phases (A..Z extension), but only three phase nodes can be transposed.

Components are connected if their nodes overlap or if a connection is drawn between the nodes. To draw a connection between nodes, click on a node with the left mouse button. A line is drawn between node and the mouse cursor. Click the left mouse button again to place the connection (clicking the right button cancels the operation). The gridsnap facility helps overlapping the nodes. Connected nodes are given the same name by the 'compile circuit' procedure used in the *Run ATP*, *Make Node Names* and *Make ATP File* options in the ATP menu. Nodes can be attached along a connection as well as at connection end-points. A connection should not unintentionally cross other nodes (what you see is what you get). A warning for node naming appears during the ATP-file creation if a connection exists between nodes of different names, or if the same name has been given to unconnected nodes. To resize a connection, click on its end-point with the left mouse button, hold down and drag. If several connections share the same node, the desired connection to resize must be selected first. Selected connection nodes are marked with squares at both ends of the selection rectangle. It is possible to draw a connection between an n-phase node and a single phase node. In this case the *Connection dialog* box will automatically appear and request the user to select which phase to connect to the single phase node. More information about multi-phase nodes and connections are given in chapter 3.

Selecting a single component and press the *Crtl+F1* key combination (or clicking *Help* in the *Component dialog* box) the component specific help is displayed. When double-clicking on a selected group of objects, the *Open Group* dialog box will appear, allowing the user to change attributes common to all components in that group, such as group number and hide state. Default component attributes are stored in support files. Access to create and customize support files is provided by the *Library* menu. The properties of the support file (icon, data and node definitions) are inherited by the component when adding it to the circuit. To edit these properties locally select the *Edit definitions* button in the *Component dialog* box.

Before trying to make simulations with ATP the proper ATP command must first be set under *Tools|Options/Preferences*. The normal setup is to choose a batch file distributed with ATPDraw for different ATP versions. The environment variable for ATP location must also be set. The user also needs to specify the timestep and simulation time under *ATP|Settings/Simulation*. ATP is then executed with *ATP|Run ATP* (shortcut F2). The first time the circuit is simulated the user needs to specify a file name, which is kept for subsequent runs. By default the file is stored in the *ResultDir* folder. The result file (PL4) will get the same name in the same location. To change the name of the file the command *ATP|Sub-process|Make ATP File* must be selected followed by F2. To plot the result a plotting command must initially be specified under *Tools|Options/Preferences*. The plot command (short cut F8) sends the current PL4 file as parameter to the call to the specified plotting program. Plotting programs pften have a refresh option as well that automatically reloads the current PL4 file.

2 Files and Data

All the result files from ATPDraw required to run an ATP simulation is stored in the *ResultDir* which the user has to select each time a circuit is opened. The default *ResultDir* location is specified in *Tools|Options/Directories* as the *ATP folder*. If several open projects share the same *ResultDir* some conflicts could occur for User specified components and LCC which both depend on lib-files written to the *ResultDir* and added to the ATP file with $INCLUDE. It is however no longer possible to overwrite and lose circuit data as a result of multiple opened circuits. In v5 ATPDraw all circuit data is stored in memory and writing to files on disk is avoided (except for $Include files). A clear distinction is made between circuit data and files in libraries on disk as shown in Fig. 2.1. The ATPDraw.ini file is now stored under the environment variable APPDATA\ATPDraw (\Documents and settings\user\ application data\atpdraw (folder name depends on Windows version)) folder where the user has writing privileges.

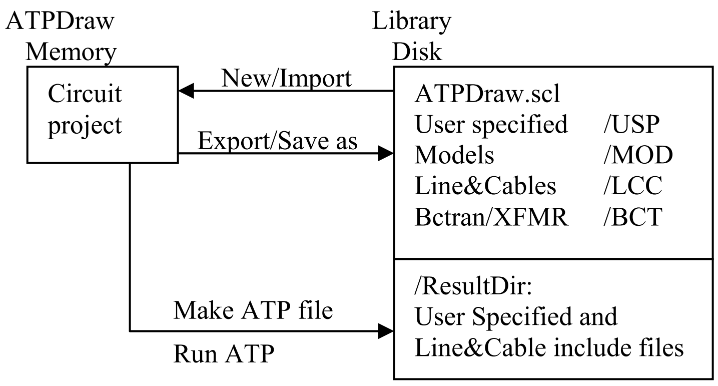


Fig. 2.1. Distinction between circuit project in memory and library files on disk.

2.1 Project file

The project file stores all the circuit data. In ATPDraw v5 the extension is changed to *.acp (**A**TPDraw **c**ircuit **p**roject) to avoid conflicts with the "black listed" *.adp extension. A single file is required to distribute the project. The project file is a zipped file following the PKZIP 2.0 standard using the *deflate* compression method. The file consists of the circuit and component data and library files for LCC components. Older versions of ATPDraw used the PKZIP 1.0 standard and the *shrink* method. Some eager anti-virus programs were starting to complain about this format. The switch to the new format is invisible to the user.

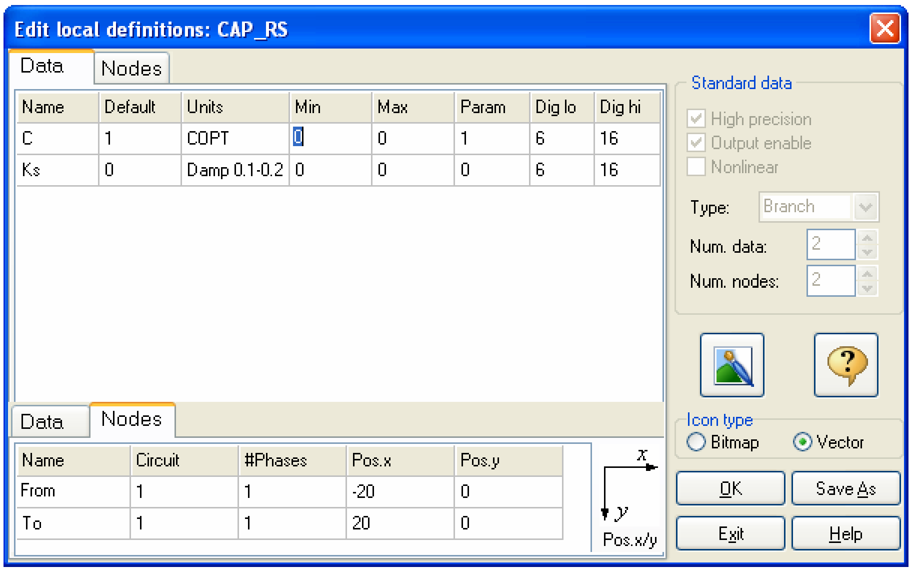
2.2 Support file

Each type of component (RESISTOR, DIODE, BCTRAN etc.) is associated with a support file. This file contains the default icon, help and definitions of nodes and data. All standard components have their support files stored in the zipped ATPDraw.scl (**s**tandard **c**omponent **l**ibrary) file. User specified components require a separate support file on disk. For Model components the support file is optional (more about that in the Models chapter), and Group components do not use a support file at all. When a component is added to the circuit, all the properties of the support file are copied to a structure in memory and the support file is never accessed again (except for Help of Standard components). In ATPDraw version prior to v5 the support files were accessed also during circuit editing and the support file for non-standard components were also stored in the project file.

The properties of components can be edited in two ways: 1) By editing the support file; *Library|Edit*. This will affect all components created afterwards. 2) By editing the definitions of the component locally (select *Edit Definitions* in the component dialog box). In both cases the same dialog box is used as shown in Fig. 2.2.

The support file format is updated and some parameters added or extended. A 12 character unit is added to all data with special support of the key words COPT, XOPT, and SOURCE. COPT will result in a unit of µF or µS dependent on the user's choice of COPT value in the

ATP|Settings dialog. Node positions are extended from the default position 1-12 on the icon border to relative coordinated Pos.x and Pos.y (-120..120 in steps of 10). The maximum length Name of nodes and data are extended from 6 to 12 characters. This is of particular importance for Models. The user can also change the icon mode from Bitmap to Vector graphics, and this is explained more later on.


Edit local definitions: CAP_RS

Name	Default	Units	Min	Max	Param	Dig lo	Dig hi
C	1	COPT	0	0	1	6	16
Ks	0	Damp 0.1-0.2	0	0	0	6	16

Name	Circuit	#Phases	Pos.x	Pos.y
From	1	1	-20	0
To	1	1	20	0

Standard data: High precision ✓, Output enable ✓, Nonlinear ☐; Type: Branch; Num. data: 2; Num. nodes: 2

Icon type: Bitmap [✓ Vector]

Fig. 2.2. Edit definitions/Edit support file.

2.3 Data file

In the earlier days of ATPDraw only the User Specified Components and Models contained external data in a lib- and mod-file respectively. As ATPDraw developed more advanced components were added requiring more data than the memory structure could hold. Consequently additional external files were added to some components like LCC (Lines&Cables), BCTRAN, XFMR (new advanced transformer model). Circuit components could also share common external files. The data files were zipped into the project file to make the project self-contained and easily distributable. When a project was opened these files had to be packed out to disk and over-write conflicts occurred resulting in "File on disk is older/newer than… Replace?" warnings. This required some active selections from the user. Normally the "Overwrite all" was safe but with several simultaneously open projects some bad mistakes were possible.

To avoid all this it was decided to simply avoid all files on disk. All previously external data (lib-files for User Specified, mod-files for Models, optional include files for some non-linear components, alc-, bct-, xfr-files for LCC, BCTRAN, and XFMR components respectively) are now stored locally in memory for each component. A clear distinction is made between library files on disk and component data in memory. All components with additional data mentioned above must be given a name by the user. For User specified components, Models, and LCC this name is important. Edited data will be copied to objects with the same name. If the user copies a component (User specified, Models or LCC) the copy will get the same name and data. When the user later changes the data in one of the copy the other is automatically updated. This is required since all these components are either associated with a lib-file on disk or by a Model definition in the ATP file. The two components can be separated by giving them different names.

For User specified components and LCC objects some library file have to be send to disk since $INCLUDE is used in the ATP-file. In the case of User specified component the lib-file is sent to disk when the ATP file is created (Run ATP selected). While for LCC the lib-file is created from the ATP generated punch file in the editing process. The lib-files are sent to the *ResultDir* folder. To avoid unnecessary conflicts between User specified and LCC a letter 'u' is added in front of the User specified component's name. BCTRAN data (punch file created by ATP) is also sent to the *ResultDir* folder. The handling of $include and punch files has been developed in several steps in version 5 of ATPDraw.

The user has the option to export the local data to an external library on disk. A new and important feature is that UNDO/REDO now also handles the additional "external" data.

2.4 Help file

Each component type is associated with a help text stored in the support file. For standard components this is editable, but will be overwritten when a new ATPDraw.scl file is installed. In v5 of ATPDraw the user can add help in three different ways. 1) Global component type in the support files. 2) Local component help via the *Edit definitions* button in the component dialog box. When a component is copied the user created help follows the copy. 3) Global user specific help stored in the /HLP directory and with name equal to the component e.g. CAP_RS.TXT. When the user selects the *Help* button all three help texts are displayed.

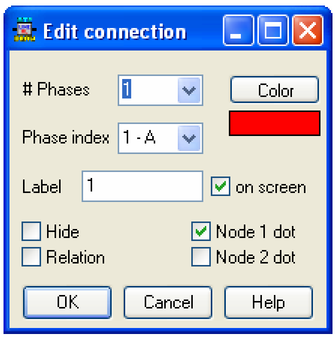
TAB. I. EDIT, FILES, DATA AND HELP FOR ATPDRAW COMPONENTS OF VARIOUS TYPES.

Com. type	Edit data	Name	File on disk	Data	Help
Standard	Component dialog (C)	Not used	No	Standard	ATPDraw.scl+ Definitions+ /HLP
Model	C+Edit → Text editor Write/import	MODEL <name> in text editor	No	Standard+ Model text	Sup-file+ Definitions+ /HLP
USP	C+Edit → Text editor Write/import	$Include field	Yes ResultDir+'u' +Name+'lib'	Standard + Include file	Sup-file+ Definitions+ /HLP
LCC	LCC dialog	Run ATP request Shown in header	Yes ResultDir +Name+'lib'	ALC data	ATPDraw.scl+ Edit defin.+ /HLP/lcc_n.txt
BCTRAN	BCT dialog	Run ATP request Shown in header	Yes ResultDir+'pch' Time stamp test	BCT data	ATPDraw.scl+ Edit defin.+ /HLP/bctran3.txt
XFMR	XFMR dialog	Not used	No	XFR data	ATPDraw.scl+ Edit defin.+ /HLP/xfmr.txt
Group	Compress	Field in dialog Just for icon	No	Standard	Definitions + /HLP

ATPDraw version 5.3 has an updated help file system. The new help file ATPDraw.chm is on the HTML platform supported as default in Windows Vista. This file replaces the old ATPDraw.hlp and ATPDraw.cnt files. The ATPDraw.chm file is substantially updated as well and the HTML standard allows the user to better search for information in the file.

3 Multi-phase and Connections

In ATPDraw version 5 a node can have up to 26 phases (A..Z node name extension). This applies also to MODELS nodes. A more generalized *Connection* is introduced with a special handling between single phase and n-phase nodes. Transpositions will only take place through 3-phase connections.
Color, label, and phase properties are given to the *Connection* as well as the possibility to force node dots on. The connection can also be turned into a *Relation* (no node connection only visualization of flow of information drawn as a dotted line) by the Relation check box. Fig. 3.1 shows the Connection dialog that appears after a right click on the connection and automatically when the user draws a connection between a single phase and a multi-phase node.


Edit connection

Phases: 1
Phase index: 1 - A
Label: 1 [✓ on screen]
Hide [ ]
Relation [ ]
[✓ Node 1 dot]
Node 2 dot [ ]

Fig. 3.1. The Connection dialog box.

Fig. 3.2 illustrates the various options for (3-phase in this case) multiphase circuits in ATPDraw. The flag DEF is set of the source node to the left this means that all connections marked with 1 will carry the phase D and so on. The color of the connections is user selectable as shown in Fig. 3.1, but as default the color and phase sequence are inherited when the user clicks on one to connection to draw a new one. There is no check if the linked connections really have the same phase sequence. Only the phase sequence of the last connection into a component is used.

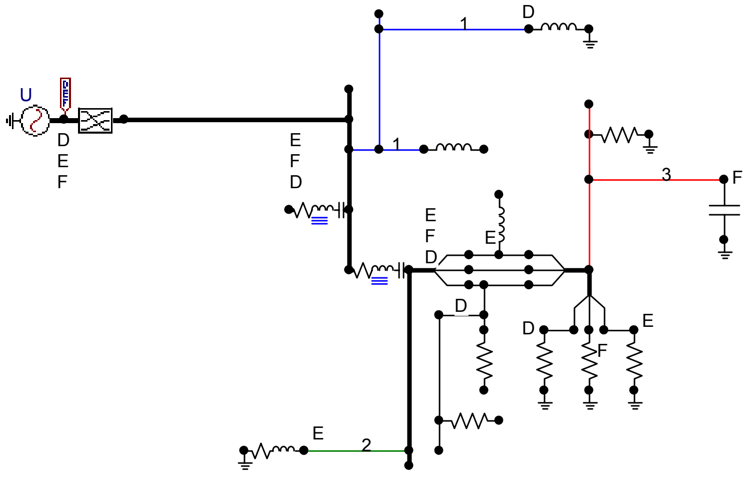


Fig. 3.2. Illustration of various phase options in ATPDraw.

Multi-phase nodes are first of all important for *MODELS* and *GROUPS*. An *n*-phase connection could also be useful just to clear up the circuit drawing. As an initial example a 6-phase connection is shown in Fig. 3.3 for communication between a 6-pulse thyristor bridge and its control circuit. This will make the drawing much easier to read. This example is also used later on to show how the thyristor bridge and the control circuit could be split in two parts.

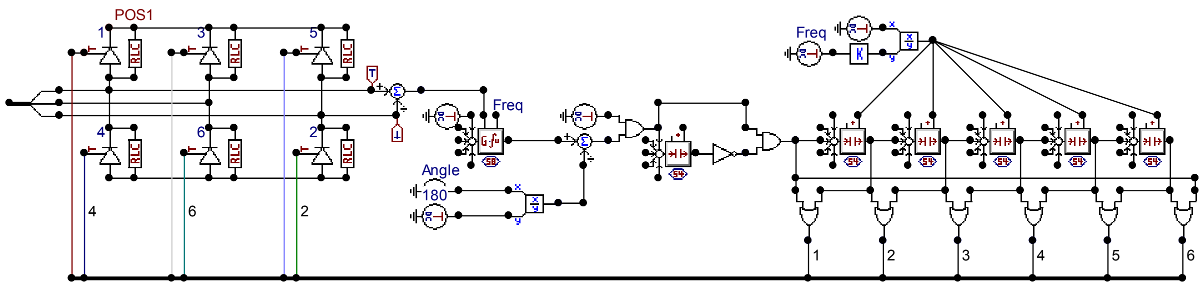


Fig. 3.3. Communicating a 6-phase signal between a thyristor bridge and its control circuit.

A 26-phase node will allow *MODEL* variables `X[1..26]` and this will allow much more signals to be communicated with a model than the previous 3-phase nodes. A typical example could be a model that calculates and output the harmonics of a signal by FFT as shown in Fig. 3.4. The current in phase A on the high voltage side is sent as input to a FFT model and some harmonics (1, 5, 7, 11, and 13) are just recorded in this case. The example in Fig. 3.4 could also to some extent have been handled in previous ATPDraw versions as well, but the FFT result would then only be available as internal variables. Using the RECORD option would made it possible to plot the result, but the harmonics could not be send as outputs and communicated with other models or the circuit.

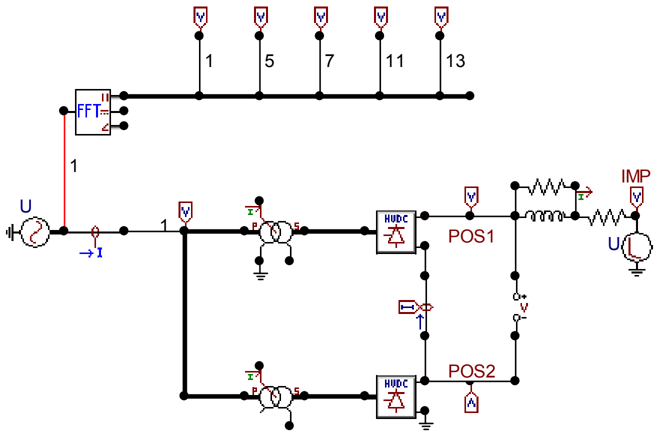


Fig. 3.4. Utilizing the multi-phase feature to obtain harmonics of a signal.

The multi-phase node and connection could also be useful to communicate more nodes to and from a *GROUP*. If the *GROUP* has a component with an *n*-phase node, this node could be set as an external connection point. A connection (bus) could not be specified as an external node however. To overcome this limitation a new dummy component called *COLLECT* (●) can be created. This component will have no data and just a single 26-phase node. The actual number of phases in use is not specified for this component, but the *#Phases* for *Connections* in Fig. 3.1 could optionally be changed.

Figures 3.5 and 3.6 show an example where the thyristor bridge with control in Fig. 8 is split in two more general parts; One for the Thyristor Bridge and one for the Control. This will allow more general building blocks for future use.

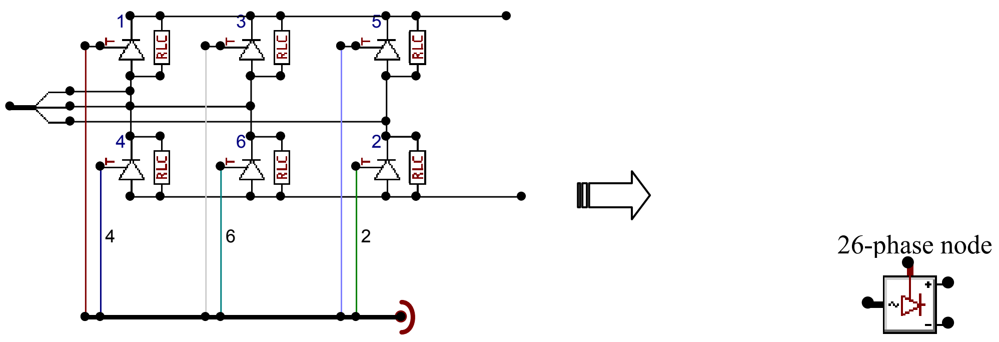


Fig. 3.5. Creating a *GROUP* out of the Thyristor Bridge.

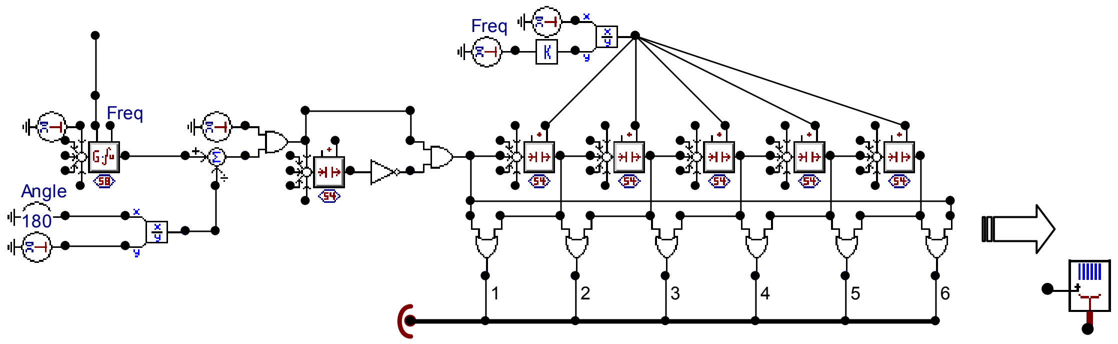


Fig. 3.6. Creating a *GROUP* out of the Control.

Putting these two *GROUPS* together will be straight forward as shown in Fig 3.7. Since the *CONNECT* nodes are 26-phase the Connection between will also be 26-phase by default, but this could optionally be changed to 6. The circuit in Fig. 3.7 looks more complicated than when the Thyristor Bridge and Control are embedded in the same *GROUP*. The flexibility is, however, increased and more user control is given to for instance the synchronization (zero crossing detection).

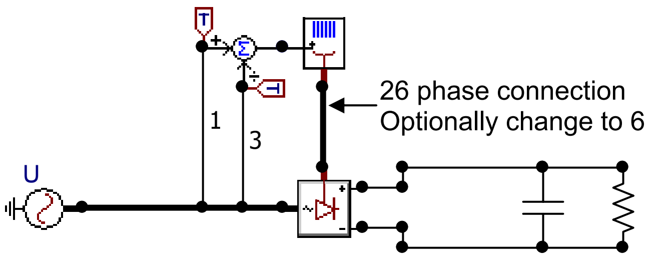


Fig. 3.7. Creating a Thyristor Rectifier from the two *GROUPS* in Figs. 3.5 and 3.6.

4 Help and Texts

Additional user information can be given to objects and circuits in several ways. Each component has a 12 character text on screen called *Label*. This label is specified in the Component dialog box and it is directly movable and editable on screen (left click). If the label appears underneath the component select *Edit|Edit Text* to enter the *Edit text* action mode (or press the Alt-key and click on the label). Components also have a *Comment* property. This is a maximum 78 character text written in the ATP file in front of the data. Help (special or general) can be given to components in three ways. 1) In the support file on disk. Standard component have there global help stored in the file ATPDraw.scl which is over-written when new versions are installed. Consequently the user is advised not to modify this global properties. 2) Global type-specific help can also be given in the /HLP directory in files with the same name as the component CAP_RS.TXT and a txt extension. 3) Local help can be given in the *Component dialog* box under the *Edit definitions*. When the *Help* button in the *Component dialog* box is pressed all the three help texts are displayed.

Connections also have this *Label* property edited in the same way as for components.

The user can also add text strings directly in the circuit window. This is done by selecting *Edit|Edit text* followed by a click in open circuit space. The text is then directly typed in the appearing edit field. From ATPDraw version 5.3 multi-lined text (with no practical size limit) is supported with font and color properties. These text strings are selectable, movable, and editable (copy/paste, compress sec.) as any other objects, but not rotateable. The text can also be compressed as part of a group. Click and hold to move, click and release to edit. Right click to open the text property dialog. The default font of the texts is set under *View|Set circuit font.* Multi-lined text from version 5.3 can not be displayed properly by older ATPDraw versions which also limit the texts to 255 characters.

To edit an existing text the user just has to click on it with the left mouse button. This direct edit option also work for labels and node names. Fig. 4.1 shows an example where texts (black) are place on screen for illustrative purposes and the node name HVBUS is edited directly on screen.

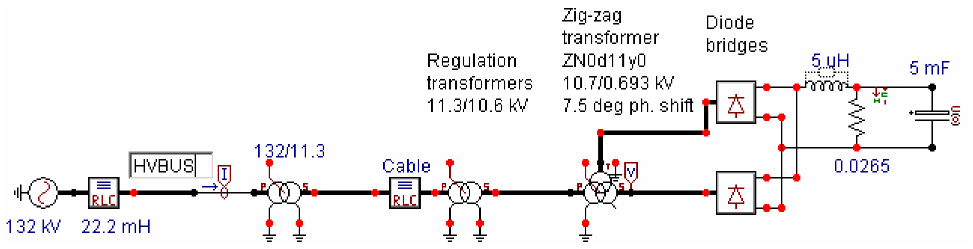


Fig. 4.1 Illustration of texts on screen and direct edit of node names.

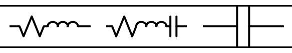
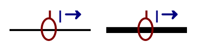
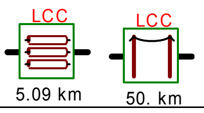
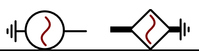
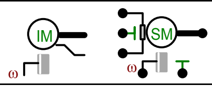
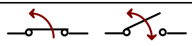
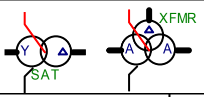
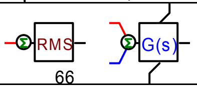
5 Vector graphics

Circuit symbols represented by vector graphics are introduced in ATPDraw from version 5.0. This was done to improve zooming capabilities and allow more dynamic symbol appearance. Components in old projects will remain Bitmaps while new added components will be Vectors. Old projects can be updated to Vector Graphics via *Library|Syncronize|Reload icons*. This will update the standard components in the current project with the icons stored in ATPDraw.scl. A component can have a bitmap or a vector icon but not both. The old bitmap format is still fully supported, but its usage is a bit limited for larger components.

5.1 Dynamic icons

Vector graphics allows more dynamics as individual elements can be turned on/off. The components shown in Tab. II make usage of this feature and respond to user changes in their parameters:

TAB. II DYNAMIC VECTOR GRAPHIC ICONS.

RLC, RLC3, RLCD3, RLCY3; R, L, C, RL, RC, LC, RLC appearance.	
PROBE_I (Current probe); Single phase or three phase appearance.	
LCC; Overhead line, single core cable, or enclosing pipe appearance. Length of transmission line optionally added.	
All sources; current (rhomb) or voltage (circle) source appearance.	
Universal machines; manual/automatic initialization, neutral grounding.	
TSWITCH (Time controlled switch); opening/closing indications.	
Transformers; Coupling (Wye, delta, auto, zigzag), two/three windings.	
TACS summation. Positive (red), negative (blue), or disconnected input. Click on the nodes to activate.	

Each time a circuit component is edited a post-processing routine is called that checks if certain hard coded parameters have changed and based on this sets the *Visible* parameter for icon elements with certain tags.

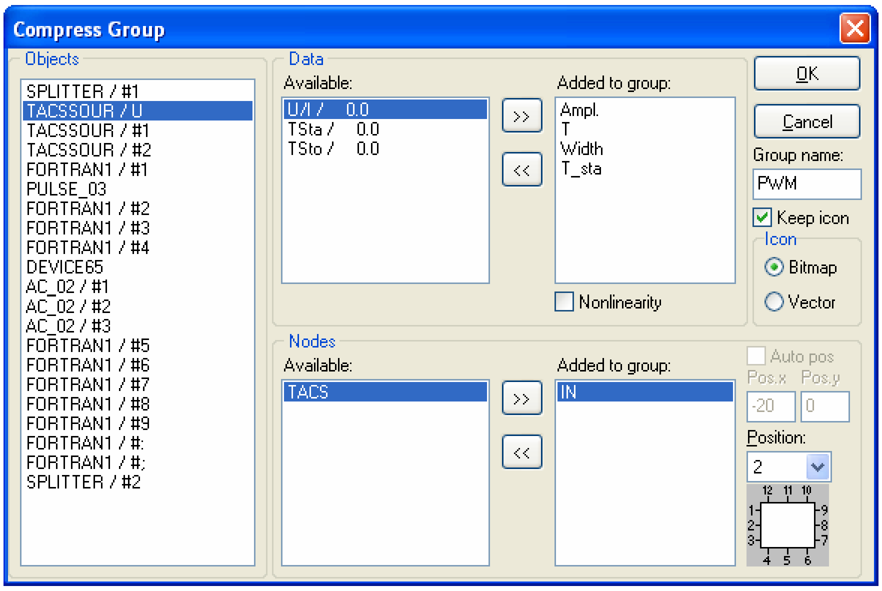
Vector graphic icons can be individually magnified (×1, ×2, ...). The user has to select the icon and move the mouse cursor to the icon border. Press ESC to go back to original size.

5.2 Automatically created icons

Groups and also Models will get their icons created automatically. Selecting Compress for a group will allow the user to identify data and input nodes, as shown in Fig. 5.1. The user can select Bitmap or Vector icon and the position of the nodes. The icon itself is however automatically created. When more nodes than 12 are used Vector icons are strongly recommended. To specify a node location outside the standard 1-12 position, specify *Position* 0 and enter the exact co-ordinates (rounded to nearest 10 pixels). For Vector graphic icons *Auto pos* is selectable, and this will assign a position to the nodes automatically. The *Keep icon* button is introduced since an already compressed group can be selected and re-compressed (edited). In order to edit the Group's icon *Edit definitions* in the component dialog box must be selected and the icon edited locally. For Groups there are no longer files on disk and the folder /GRP is no longer created by ATPDraw. The Vector graphic icon will consists of a box with the texts: GROUP and Group name (the only purpose of this name).

Model icons can be created both manually (*Library*|*New*|*Model*) and automatically. It is expected that automatic creation will dominate. This process was outlined in the previous section. The automatically created Vector graphic icon will consist of a box with the texts: MODEL and the name of the model (as found in the model's header).

Customizing the automatically created icons is somewhat challenging. The main problem is the node positions which can deviate from the traditional -20, -10, 0, 10, 20 locations in x and y coordinates and with the position (0, 0) not necessarily in the centre of the icon. The reason for the deviation is the need to allow more node positions than 12 (actually 16 predefined positions are located on both the left and right sides of the automatically created icons). The link between node positions and icon appearance could be improved. In a future graphical vector editor the nodes should preferably be a part of the icon and connection lines from the nodes to the component symbol should moved with the nodes.


Compress Group

Objects:
SPLITTER / #1
TACSSOUR / U
TACSSOUR / #1
TACSSOUR / #2
FORTRAN1 / #1
PULSE_03
FORTRAN1 / #2
FORTRAN1 / #3
FORTRAN1 / #4
DEVICE65
AC_02 / #1
AC_02 / #2
AC_02 / #3
FORTRAN1 / #5
FORTRAN1 / #6
FORTRAN1 / #7
FORTRAN1 / #8
FORTRAN1 / #9
FORTRAN1 / #:
FORTRAN1 / #;
SPLITTER / #2

Data
Available:
U/I / 0.0
TSta / 0.0
TSto / 0.0

Added to group:
Ampl.
T
Width
T_sta

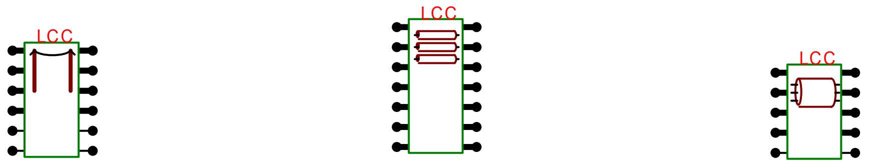
[ ] Nonlinearity

Group name: PWM
[✓] Keep icon
Icon: [✓ Bitmap] Vector

Nodes
Available: TACS
Added to group: IN
[ ] Auto pos
Pos.x -20 Pos.y 0
Position: 2

Fig. 5.1. The Compress dialog.

The LCC module is extended to 21-phase lines and the icon will adapt to the selection of Overhead line, single core cable, or enclosing pipe. All these icons are predefined with the option to switch between the types coded in the Layer parameter. The Length of the line/cable is optionally added to the icon.

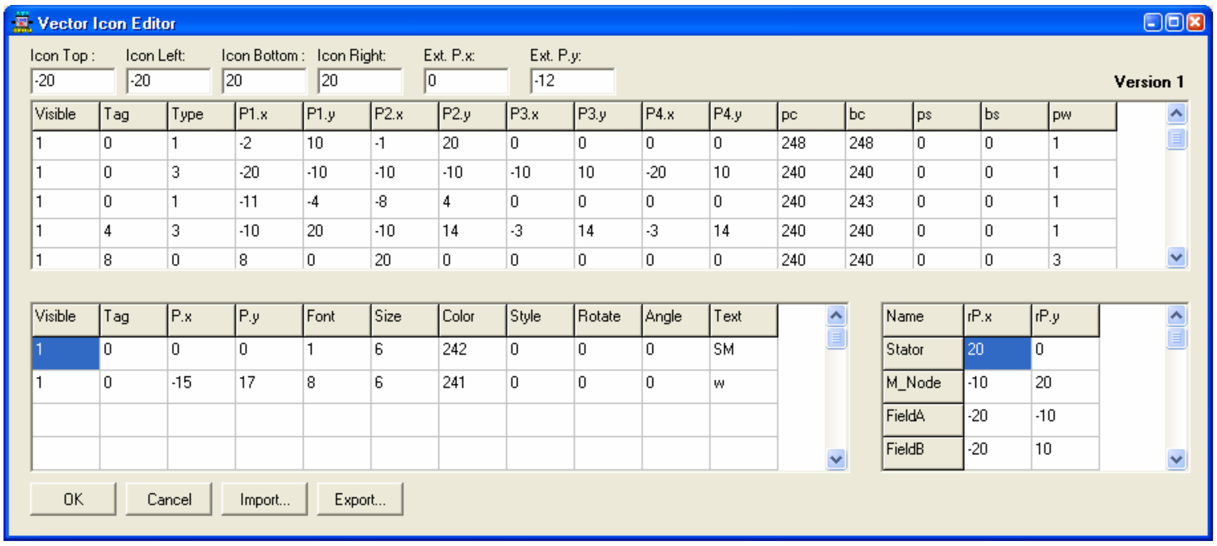


14-phase overhead line 21-phase single core cable 13-phase enclosing cable
Fig. 5.2. Examples of LCC object icons

5.3 Editor

Version 5.0 of ATPDraw had a text based vector editor as shown in Fig. 5.3. The user had to type in values in a string-grid. This can be used for minor modifications, but requires generally detailed knowledge. The sequence of the elements defines their order. The first elements are drawn first and will thus appear in the back. The elements are divided in two types: shapes and texts. The shapes are drawn first followed by the texts. Double clicking a row enables to insert a row in front or delete the row.

From ATPDraw version 5.2/5.3 a graphical editor replaces this text based editor. The vector icon is still based on the concept of elements, divided in shapes and text. The content of next sub-chapters is therefore also relevant for the new editor.



Icon Top: -20 Icon Left: -20 Icon Bottom: 20 Icon Right: 20 Ext. P.x: 0 Ext. P.y: -12

Visible	Tag	Type	P1.x	P1.y	P2.x	P2.y	P3.x	P3.y	P4.x	P4.y	pc	bc	ps	bs	pw
1	0	1	-2	10	-1	20	0	0	0	0	248	248	0	0	1
1	0	3	-20	-10	-10	-10	-10	10	-20	10	240	240	0	0	1
1	0	1	-11	-4	-8	4	0	0	0	0	240	243	0	0	1
1	4	3	-10	20	-10	14	-3	14	-3	14	240	240	0	0	1
1	8	0	8	0	20	0	0	0	0	0	240	240	0	0	3

Visible	Tag	P.x	P.y	Font	Size	Color	Style	Rotate	Angle	Text
1	0	0	0	1	6	242	0	0	0	SM
1	0	-15	17	8	6	241	0	0	0	w

Name	rP.x	rP.y
Stator	20	0
M_Node	-10	20
FieldA	-20	-10
FieldB	-20	10

Fig. 5.3. Vector graphic text based editor

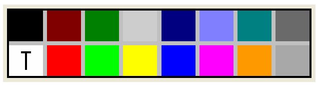
5.3.1 Common properties

Visible: Is a flag that tells if the element should be initially visible. ATPDraw uses this flag at runtime to turn on/off elements in response to user selections. The graphical routines will only draw elements marked with Visible>0.

Tag/Layer: This is used to identify and group various elements and ATPDraw uses this at runtime to set the Visible flag. The handling of Tag/Layer is hard coded in ATPDraw.

Colors: Colors are described by 8-bits. For the standard icons only the old bitmaps colors are used. These are given values from 240 to 255. ATPDraw 5.3 offers mouse click shortcuts to the standard colors (left click on the palette selects pen color and right click selects brush/fill).

240: Black	248: White
241: Dark red	249: Light read
242: Dark green	250: Light green
243: Light gray	251: Yellow
244: Dark blue	252: Blue
245: Sky blue	253: Magenta
246: Cyan	254: Orange
247: Dark gray	255: Gray

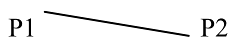
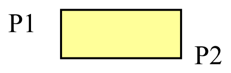
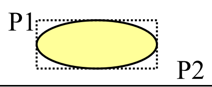
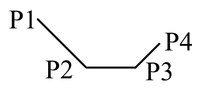
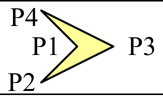
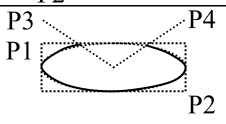
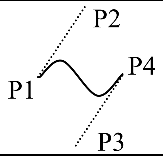
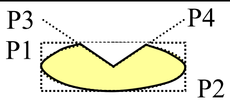


In addition color codes from 0 to 63 are used in RGB format. The lower 6-bits are used to set Red-Green-Blue and this is more in line with how Windows presents 8-bits colors.

5.3.2 Shapes

These can be of the types; line, rectangle, ellipse, polyline, polygon, arc, Bezier curve, and pie (added from version 5.3). Each shape is described by a maximum of four points. Lines, rectangles and ellipses require only two points, and polylines, polygons (as well as arcs, Bezier, and pie) must always use four points (repeat the last point if necessary).

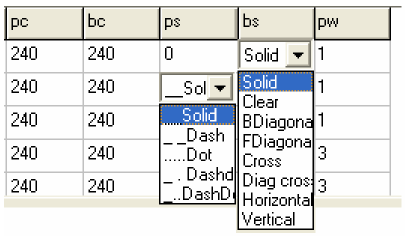
TAB. III DESCRIPTION OF THE EIGHT SHAPE TYPES.

Type	**Appearance**	**Styles and colors**
Line		pc= color of line, ps=dashing/dotting of line, pw=thickness bc= not used, bs= not used.
Rectangle		pc= color of line, ps=dashing/dotting of line, pw=thickness bc= color of fill, bs= type of fill (set to clear for no fill).
Ellipse		pc= color of line, ps=dashing/dotting of line, pw=thickness bc= color of fill, bs= type of fill (set to clear for no fill).
Polyline		pc= color of line, ps=dashing/dotting of line, pw=thickness bc= not used, bs= not used.
Polygon		pc= color of line, ps=dashing/dotting of line, pw=thickness bc= color of fill, bs= type of fill (set to clear for no fill).
Arc		pc= color of line, ps=dashing/dotting of line, pw=thickness bc= not used, bs= not used. Draws counter clockwise from the intersection between the ellipse and the line defined by P3 to the intersection formed by P4.
Bezier		pc= color of line, ps=dashing/dotting of line, pw=thickness bc= not used, bs= not used. P1 & p4 end points of curve. P2 & P3 give the extent of the tangents at the end points.
Pie		pc= color of line, ps=dashing/dotting of line, pw=thickness bc= color of fill, bs= type of fill (set to clear for no fill).

pc, bc: pen and brush color, ps, bs: pen and brush style

5.3.3 Pen and brush

The pen is used to draw the enclosure of a shape, and brush is used to fill it. Only Rectangle, Ellipse, Polygon, and Pie use the Brush. The Pen and Brush color (pc and bc) follow the color coding shown above. The Pen and Brush style (ps and bs) are selected in combo boxes appearing when the grid is clicked as shown in Fig. 5.5. These are the standard windows style types. The pen style is only well reproduced on screen if the brush style is set to clear. The same type of dropdown lists is also available in the new graphical editor.



pc	bc	ps	bs	pw
240	240	0	Solid	1
240	240	__Sol		1
240	240			1
240	240			3
240	240			3

Fig. 5.4. Selecting the pen and brush styles.

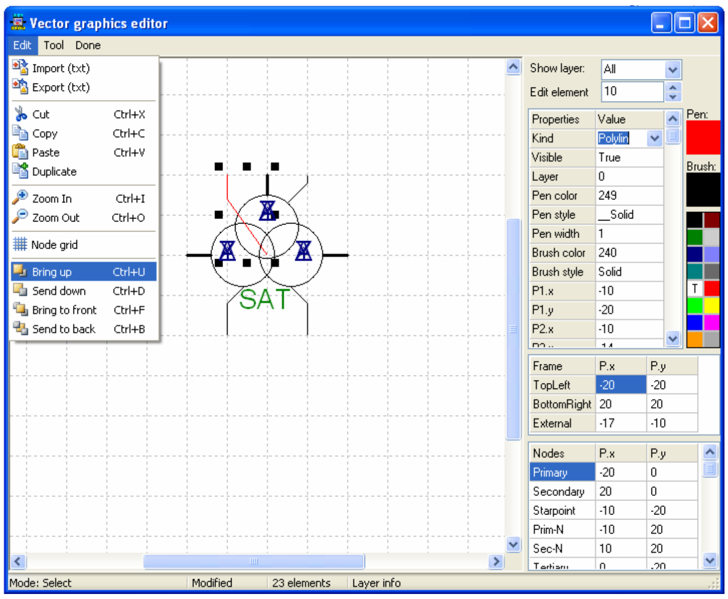
5.3.4 Texts

Visible, Tag, and Color attributes have the same meaning as for Shapes. The point P defines the middle point of the text (not the lower left corner).

Font:
0,1: 'Arial'; 2: 'System'; 3: 'MS Sans Serif'; 4: 'Arial Narrow';
5: 'Comic Sans MS'; 6: 'Courier'; 7: 'Times New Roman'; 8: 'Symbol';
Size: The height of the font in pixels (normal zoom). 5 is barely readable in 100 % zoom.
Style: 0: normal, 1: bold, 2: italic, 4: underline, 8: strike out. Any combination 0..15 possible.
Rotate: 1= rotate text with icon, 0= no rotation.
Angle: Initial rotation angle of text. 0, 90, 180 and 270 deg. supported.
Text: Up to 8 character text string.

5.4 New editor

Version 5.2 of ATPDraw introduces a new vector graphic editor as shown in Fig. 5.5.



Show layer: All
Edit element 10

Properties	Value
Kind	Polylin
Visible	True
Layer	0
Pen color	249
Pen style	__Solid
Pen width	1
Brush color	240
Brush style	Solid
P1.x	-10
P1.y	-20
P2.x	-10

Frame	P.x	P.y
TopLeft	-20	-20
BottomRight	20	20
External	-17	-10

Nodes	P.x	P.y
Primary	-20	0
Secondary	20	0
Starpoint	-10	-20
Prim-N	-10	20
Sec-N	10	20

Fig. 5.5 New Vector graphic editor. From version 5.2.

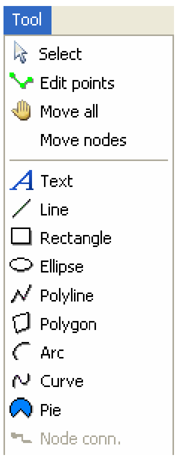


Fig. 5.6 The Tools menu.

The benefit is visual feedback of the data entered in the properties grids. In addition better support of the colors is introduced. The color menu to the right is a short cut to the standard ATPDraw colors (left click for pen and right click for brush color). The order of the elements can be controlled by the user from the *Edit|Bring/Send up/down* menu items. The data is entered manually in the properties grid similar to Fig. 5.3 for the selected element. An element of the icon is selected either by the *Edit element* spin-edit box to the top-right or by clicking the element in the icon window. Once selected Elements can also be moved and resized by the mouse. Holding down the Shift enables using the arrow keys to move the element one pixel at the time. Click on the black selection squares to resize. Individual points can edited when selecting the *Tool|Edit points* mode. In this case the individual points are marked with a lime color. Fine tuning can of course also be performed in the property grid.

The nodes can also be moved with the mouse by setting the drop-down list *Show element* to *all+nodes* and choosing the *Tool|Move nodes* mode. The nodes only move in steps of the grid snap size (10). Turn on the grid via *Edit|Node grid*. The node positions can also be edited directly in the Nodes grid in the low-right corner of the editor.

All elements can also be moved simultaneously by selecting the *Tool|Move all* mode. This applies also to nodes when they are visible.

New elements are added to the icon from the Tools menu shown in fig. 5.6.

After selecting on of the tools the user has to click in the icon grid to place the points (click and release). Remember that the polyline, Bezier, arc, and pie require 4 points (double click to place two points). See Tab. III to understand how the point locations (P1..P4) are related to the shape. Drawing arcs, curves, and pies requires some training.

Rectangles and Ellipses can also be rotated by specifying an angle -90..90 in P3.x position if the Property grid. Unrotated rectangles can also be rounded by specifying the P3.y value. Rotated elements are represented by Bezier curves and polygons internally and moving such elements towards the border can result in some unexpected behavior. Rotated ellipses/rectangles also behave a bit strange under resize.

The *Frame* specified in the middle grid consists of a Rectangle characterized by *TopLeft* and *BottomRight* and an *External* point. The rectangle is the selection area the user has to click inside to select the component in the circuit window and that also encloses the component when it is selected. The External point is used to attach the branch output request indicators.

Note:

- Positioning of text is somewhat imprecise and a problem with alignment seems to exist. +- 1 pixel adjustments are often required. Rotation of texts is possible only by 90, 180, and 270 degrees. A single text can hold only one color and one font, but mixing several different texts is possible.
- Working with arcs, pies and bezier curves very often requires the Edit point mode. For arcs and pies the first and second points control the top-left and bottom-right of the enclosing ellipsis while the third and forth point controls the cutting. The end result is not shown until the last point is placed. For a bezier curve the first and forth poins are the endpoints while the second and third point control the end point slopes.
- When converting from a bitmap icon to a vector icon, the vector icon is initially filled with empty elements. To empty the icon you have to go to the last element (the number of elements is listen in the control bar at the bottom) and manually delete all elements by pressing and holding down CTRL+X. The frame must also be set to reasonable values. **Important**: If the frame has no extension it is not possible to select the icon in the circuit.
- The icon does not necessarily have to be placed in the centre of the icon window. Nodes at any grid position are accepted.
- When turning on the node names and frame (Layer: All+Nodes) the node names are oriented outwards from the frame. If the frame is set to strange values the node name orientation and position could be awkward. The best result is obtained if the nodes are placed on the frame border.

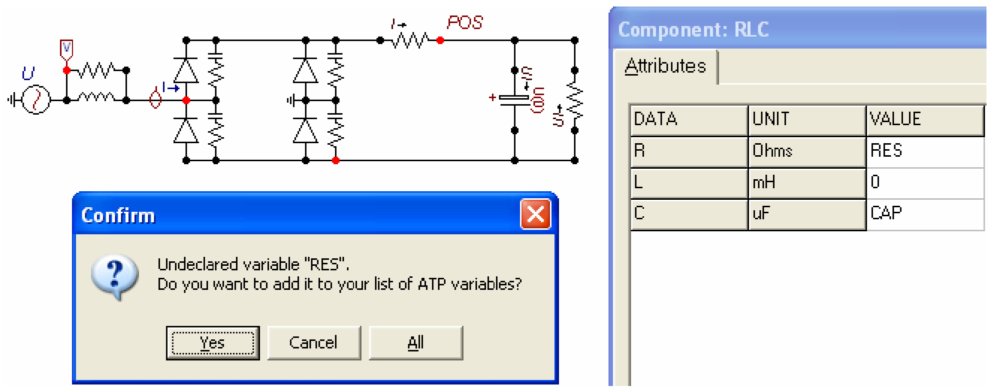
Things not supported in this version are:

- Selection of multiple elements (just one or all)
- General rotation of elements (enabled only for ellipsis and rectangles)
- Handling of layers
- Undo/Redo
- Node connections (it is planned to add a special shape that is connected to a node and moved with this)
- Import of standard graphics (metafiles/bitmaps). Probably an additional image background will be added at some future point.

6 $Parameter and Pocket calculator

$Parameters is a new feature in ATP that allows the user to assign text variables to data and declare these variables for the whole data case later. This feature is particularly useful when a data value is used several times in the circuit. Earlier the user had to open all dialog boxes of the involved components in such cases. This was time consuming and might lead to errors if the user forgets to change values of some components. Since version 3.1 of ATPDraw, the user is also free to assign a 6 (or less) characters text string or variable to most data for standard components instead of a data value in the component dialog box. This is permitted as long as the parameter option is set in the support file. When specifying the data variable the user does not have to think about the number of allowed characters in the ATP-file. ATPDraw will add underscore characters to fill the maximum number of characters. Values can later be assigned to these variables and this is written to the ATP input file within a `$PARAMETER – BLANK` block.

A typical example is shown in Fig. 6.1. This is a single phase rectifying bridge distributed with ATPDraw as `Exa_1.adp`. It consists of 4 diodes with snubber circuits. The RC values of these snubbers are identical for all diodes in the practice and can be specified with text variables `RES` and `CAP` respectively. When the user specifies a variable name for the first time, a message box appears and the user is requested to confirm the operation before the new entry were added to the global list of variables. If you try to enter special characters in this field, an error message prevents this.



Undeclared variable "RES".
Do you want to add it to your list of ATP variables?

DATA	UNIT	VALUE
R	Ohms	RES
L	mH	0
C	uF	CAP

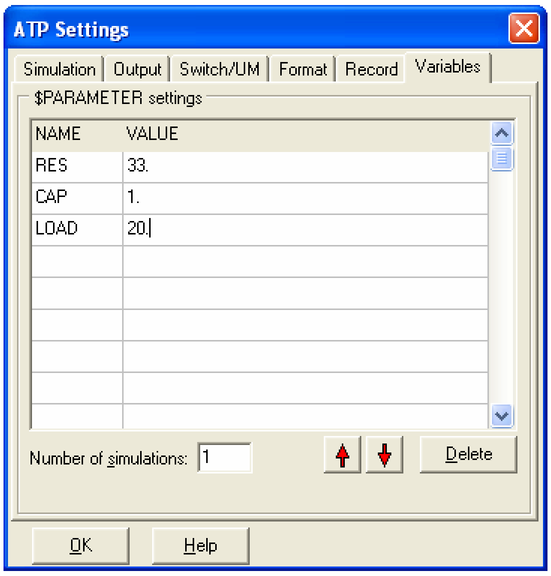
Fig. 6.1 - Specifying text variables `RES` and `CAP` in the component dialog box for an RLC object

Numerical values can be assigned to variables on the *Variables* page under main menu *ATP | Setting* as shown in Fig. 6.2**Feil! Fant ikke referansekilden.**. Variable names are declared in the left column and you can specify data values or a text string in free format in the right column on this page. Sorting the declarations is possible with the arrow buttons. Deleting declarations is also supported.

If variables `RES` and `CAP` are declared twice with different precision settings, i.e. *$Vintage,1* is checked in a component dialog box and unchecked in another, it will be declared twice with 3 and 13 underscore characters added in the $Parameter declaration. This process is hidden, however, but the result is seen in the final ATP-file.

If you change the names in the left side column this will affect the text strings (variables) specified in the components and you will be requested about what action to take (see Fig. 6.3).

Available actions are: reset the variable to zero or the default value (from the support file), or select parameter and then decide which variables should replace the no longer defined one.



NAME	VALUE
RES	33.
CAP	1.
LOAD	20.

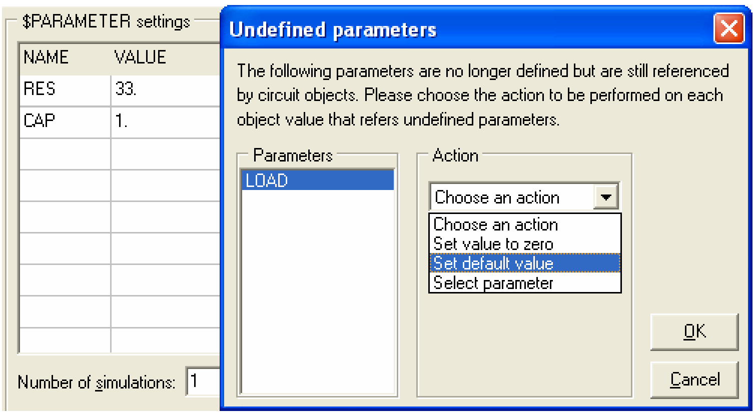
Number of simulations: 1

The *Number of simulations* field is for POCKET CALCULATOR feature of ATP. When this is higher than unity the variable `KNT` can be used in the right column for the current simulation number. This allows multiple ATP simulations where specific data variables can be a function of the simulation number. The specified variables `RES`, `CAP` and `LOAD` are written in the ATP-file followed by underscore characters to enable maximum precision. The `$PARAMETER` cards are written at the bottom of the ATP-file with after a `/REQUEST` card, as shown below.

Fig. 6.2 - Assigning values to the variables.

```
/REQUEST
$PARAMETER
RES________________= 33.
RES___= 33.
LOAD__= 20.
CAP________________= 1.
CAP___= 1.
BLANK $PARAMETER
```

IMPORTANT! Always use a period '.' after a number in the value field.



NAME	VALUE
RES	33.
CAP	1.

Number of simulations: 1

The following parameters are no longer defined but are still referenced by circuit objects. Please choose the action to be performed on each object value that refers undefined parameters.

Parameters: LOAD

Action: Choose an action
- Choose an action
- Set value to zero
- Set default value
- Select parameter

Fig. 6.3 - Action to take when a parameter no longer defined.

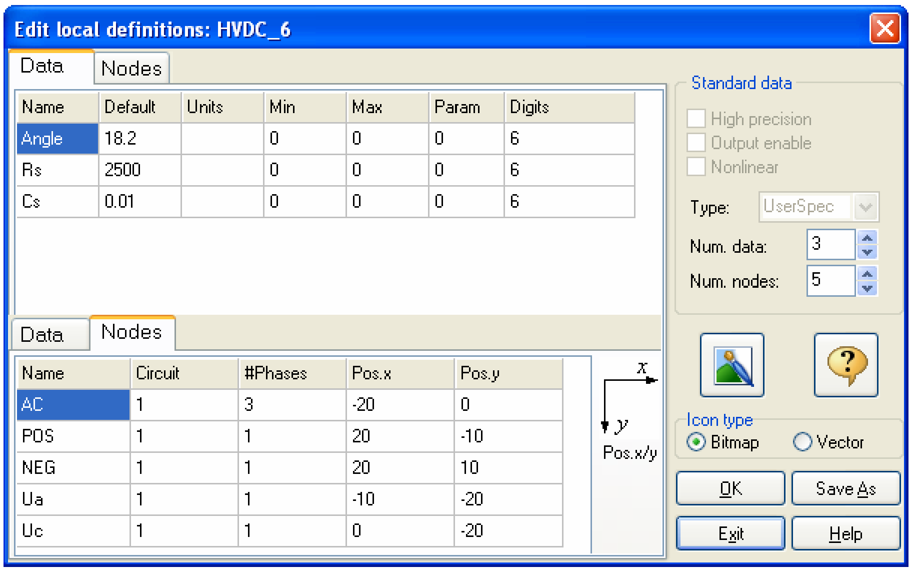
7 Basic components

The component dialog box is a bit updated. All data now has a UNIT field and it is possible to Copy/Paste the entire data grid. Instead of the icon-editor symbol in the lower left corner, the Component dialog has an *Edit definitions* button instead. Clicking this button displays the *Edit support file* dialog, but now only the local definitions are edited. Of most importance are the icon and help buttons as well as the UNIT field. The special keywords XOPT, COPT and SOURCE are supported for the UNIT.

For nonlinear components the Component dialog has a second page called Characteristics. In this page you can specify individual points in a string grid or enter a formatted text by clicking the *Edit* button. This will bring up the text editor where you can enter your points or import an existing file. Remember to complete the points by '9999.' Since ATPDraw now supports up to 64 data for each component the need for the formatted text is reduced. It is possible to select data points from any text file editor followed by Ctrl+C and paste it into the standard characteristics by clicking the *Paste* button. The *Copy* button can be used to copy to other components or a text file.

8 User specified components

User specified components are based on the Data Base Module feature of ATP. It is in principle a stripped ATP file with the option to pass in parameters (node names and data), much like a procedure. It is this communication of parameters that makes User Specified components hard to use. The end result of a Data Base Module execution in ATP is a text file that consists of the original ATP file, a header explaining where the parameters should be inserted, and a trailer listing the arguments. This text file (called library file) must be created externally (unless the user is an expert in the library file format). In addition a support file must be created via *Library|New|User specified|Sup*. This file must give the number of nodes and data as well as the component icon (and help file). Fig. 8.1 shows the support file dialog for the controlled HVDC Thyristor Bridge from ex_4.acp. 3 data and 5 nodes are specified which must match the trailer in the library file. Also note that the node positions are specified as xy-coordinates relative to the icon (rounded off to 10 pixels). Left side (1-3): (-20,-10),(-20,0),(-20,10), Bottom (4-6): (-10,20),(0,20),(10,20), Right side (7-9): (20,10),(20,0),(20,-10), Top (10-12): (10,-20),(0,-20),(-10,-20)…etc. Version 5.3 of ATPDraw introduce short cut keys Atl+F1..F12 for the old border positions (1..12).



Name	Default	Units	Min	Max	Param	Digits
Angle	18.2		0	0	0	6
Rs	2500		0	0	0	6
Cs	0.01		0	0	0	6

Name	Circuit	#Phases	Pos.x	Pos.y
AC	1	3	-20	0
POS	1	1	20	-10
NEG	1	1	20	10
Ua	1	1	-10	-20
Uc	1	1	0	-20

Fig. 8.1 Support file specification for the HVDC_6.sup object.

To add a User specified component to the circuit the *Files* item in the Fig. 8.2 should be selected. The *Library* item results in a predefined component without parameters.

Fig. 8.3 shows the Component dialog box for the above specified and selected HVDC_6.SUP component. The library file must next be loaded into this component. This is done under by clicking the Edit button in Fig. 8.3 which will bring up the embedded text editor as shown in Fig. 8.4. A predefined library file (usual case) is loaded by selecting *File|Import*. The file is loaded into memory and can be modified in the text editor without affecting the original file on disk. Select *Done* when finished. The $Include field in Fig. 8.3 visualize the original name of the file but the user is free to specify a new name here (.lib is not important). The specified name becomes the name of the new library file created in the ResultDir folder when the final ATP file is created. Actually, the letter ‘u’ is added in front of the final lib-file name to avoid name conflicts with Lines&Cables (LCC) which uses the same lib-file principle. If two or more User Specified components share the same $Include name the lib-file data is copied between the components when one of them is edited. The *Send parameter button* is checked to tell ATPDraw to send the node and data parameters in the $Include call. And the *Internal phase seq.* button tells that n-phase nodes should be sent as a single 5 character node name without the A..Z node name extensions that are added internally in the library file.

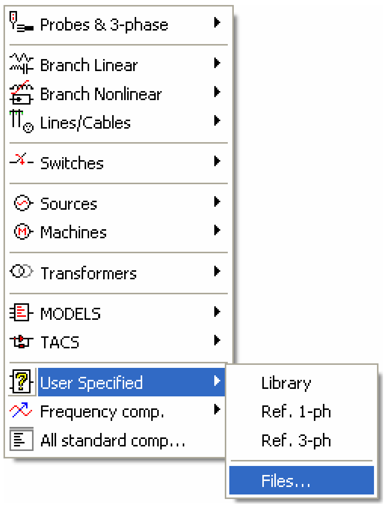
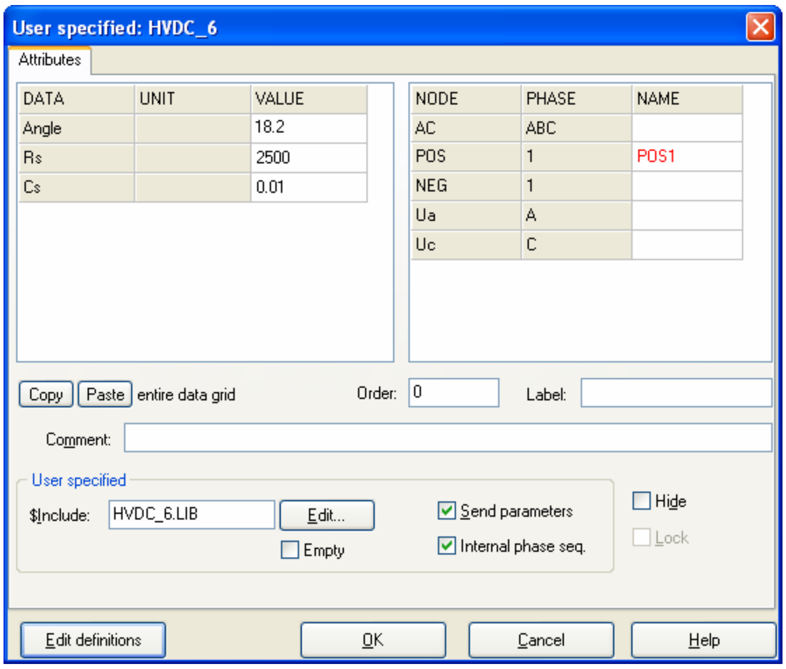


Fig. 8.2 Selecting User Specified (predefined) components.



User specified: HVDC_6

Attributes

DATA	UNIT	VALUE
Angle		18.2
Rs		2500
Cs		0.01

NODE	PHASE	NAME
AC	ABC	
POS	1	POS1
NEG	1	
Ua	A	
Uc	C	

Copy Paste entire data grid Order: 0 Label:

Comment:

User specified

$Include: HVDC_6.LIB Edit...

☐ Empty

☑ Send parameters

☑ Internal phase seq.

☐ Hide

☐ Lock

Edit definitions OK Cancel Help

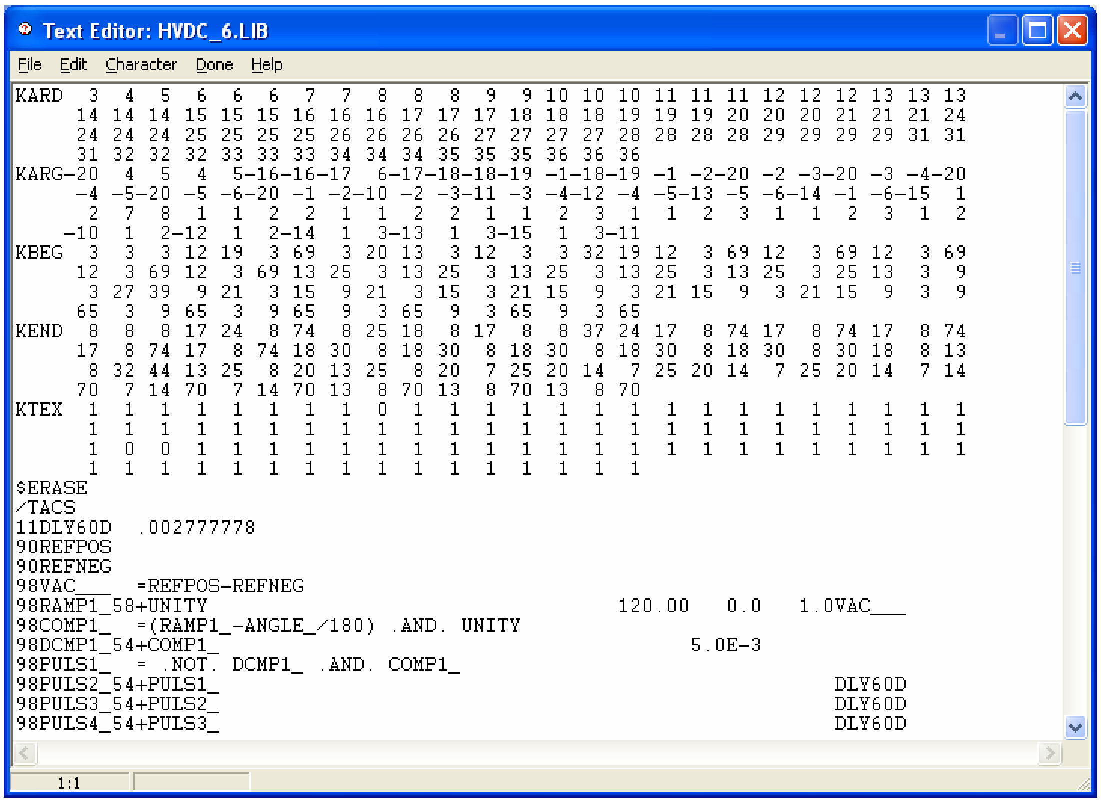
Fig. 8.3 Component dialog box for User Specified component HVDC_6.

Fig. 8.4 shows the header of the library “file” (Data Base Module data). The first argument KARD tells the line number after the header the argument should be inserted. The second argument KARG tells the argument number to be inserted where a negative number means a dymmy argument (used for internal node names). The third and fourth argument tell the first (KBEG) and last (KEND) column numbers of the argument. The sixth and last argument tells if the argument is a text string (node name part) or a data parameter. The trailer looks as follows:

```
$EOF   User-supplied header cards follow.         01-Dec-95  20.11.59
ARG,U____,POS___,NEG___,REFPOS,REFNEG,ANGLE_,Rsnub_,Csnub_
NUM,ANGLE_,Rsnub_,Csnub_
```

```
DUM,PULS1_,PULS2_,PULS3_,PULS4_,PULS5_,PULS6_,MID1__,MID2__,MID3__
DUM,GATE1_,GATE2_,GATE3_,GATE4_,GATE5_,GATE6_,VAC___,RAMP1_,COMP1_
DUM,DCMP1_,DLY60D
```

ARG lists all the arguments sent in the $Include call. NUM list which of these arguments are data parameters. DUM lists additional dummy arguments. The special request in the ATP file $DUMMY, XYZ000 tells ATP to replace all DUM declared arguments to be replaced by XYZ fooled by a three digit number incremented sequentially. In this way internal node name is avoided to be duplicated when the same User Specified component is used several time. The main challenge with User Specified components and Data Base Modules in ATP is the strict rules on the number of columns and insufficient error messages from ATP on this point.



```
KARD  3  4  5  6  6  6  7  7  8  8  8  9  9 10 10 10 11 11 11 12 12 12 13 13 13
     14 14 14 15 15 15 16 16 16 17 17 17 18 18 18 19 19 19 20 20 20 21 21 21 24
     24 24 24 25 25 25 25 26 26 26 26 27 27 27 27 28 28 28 28 29 29 29 29 31 31
     31 32 32 32 33 33 33 34 34 34 35 35 35 36 36 36
KARG-20  4  5  4  5-16-16-17  6-17-18-18-19 -1-18-19 -1 -2-20 -2 -3-20 -3 -4-20
     -4 -5-20 -5 -6-20 -1 -2-10 -2 -3-11 -3 -4-12 -4 -5-13 -5 -6-14 -1 -6-15  1
      2  7  8  1  1  2  2  1  1  2  2  1  1  2  3  1  1  2  3  1  1  2  3  1  2
    -10  1  2-12  1  2-14  1  3-13  1  3-15  1  3-11
KBEG  3  3  3 12 19  3 69  3 20 13  3 12  3  3 32 19 12  3 69 12  3 69 12  3 69
     12  3 69 12  3 69 13 25  3 13 25  3 13 25  3 13 25  3 13 25  3 25 13  3  9
      3 27 39  9 21  3 15  9 21  3 15  3 21 15  9  3 21 15  9  3 21 15  9  3  9
     65  3  9 65  3  9 65  9  3 65  9  3 65  9  3 65
KEND  8  8  8 17 24  8 74  8 25 18  8 17  8  8 37 24 17  8 74 17  8 74 17  8 74
     17  8 74 17  8 74 18 30  8 18 30  8 18 30  8 18 30  8 18 30  8 30 18  8 13
      8 32 44 13 25  8 20 13 25  8 20  7 25 20 14  7 25 20 14  7 25 20 14  7 14
     70  7 14 70  7 14 70 13  8 70 13  8 70 13  8 70
KTEX  1  1  1  1  1  1  1  1  0  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1
      1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1
      1  0  0  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1
      1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1
$ERASE
/TACS
11DLY60D  .002777778
90REFPOS
90REFNEG
98VAC___  =REFPOS-REFNEG
98RAMP1_58+UNITY                                  120.00   0.0   1.0VAC___
98COMP1_  =(RAMP1_-ANGLE_/180) .AND. UNITY
98DCMP1_54+COMP1_                                       5.0E-3
98PULS1_  = .NOT. DCMP1_ .AND. COMP1_
98PULS2_54+PULS1_                                                   DLY60D
98PULS3_54+PULS2_                                                   DLY60D
98PULS4_54+PULS3_                                                   DLY60D
```

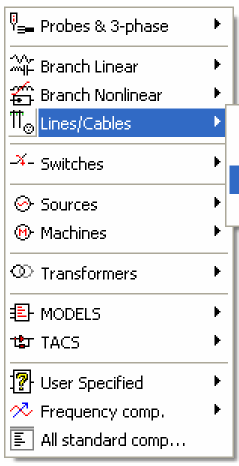
Fig. 8.4 The embedded text editor and the library “file”.

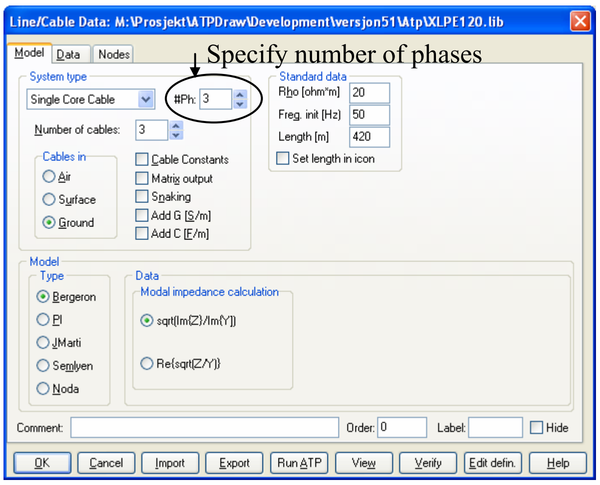
9 Lines and Cables

Up to version 3.7 the user had to first select the number of phases for the LCC components. To change the number of phases (typically investigate the influence of ground wires in a transmission line) the user had to select a new component and *Import* the data from the previous LCC component. Several users found this confusing and felt rather uncomfortable with the *Import* option. In ver. 3.8 the user just has to select a single LCC component with an internally changeable number of phases. From ATPDraw v5 21 phases are supported in LCC.

Every time the number of phases is changed the original icon is reloaded. If the user has modified the icon (*Edit defin.* button in Fig. 9.2) he risks loosing his artwork if this is not kept in a copy (*Undo* should be possible though). Fig. 9.2 shows that the *Order* and *Label* edit boxes as well as the *Hide* button has been added during the development of ver. 3. The *Comment* is now written to the ATP file.

Support of multi-section Pi-models for cable constants was improved in ver. 3.7. This involved a substantial update of the procedure that converts punch files to data base module files. Semlyen models are now supported in cable constants. Version 5.1 introduced a choice of how the surge impedance of a Bergeron cable model should be calculated.





Line/Cable Data: M:\Prosjekt\ATPDraw\Development\versjon51\Atp\XLPE120.lib

System type: Single Core Cable
#Ph: 3
Number of cables: 3
Cables in: Air / Surface / [✓ Ground]
Cable Constants ☐
Matrix output ☐
Snaking ☐
Add G [S/m] ☐
Add C [F/m] ☐

Standard data
Rho [ohm*m]: 20
Freq. init [Hz]: 50
Length [m]: 420
Set length in icon ☐

Model
Type: [✓ Bergeron] / PI / JMarti / Semlyen / Noda
Data – Modal impedance calculation: [✓ sqrt(Im(Z)/Im(Y))] / Re(sqrt(Z/Y))

Comment:
Order: 0
Label:
Hide ☐

Fig. 9.1. Selecting the LCC comp. Fig. 9.2. The LCC dialog box where the number of phases is changeable.

From ATPDraw version 5.3 a Nodes page is available where the user can give node names and more important; assign conductor numbers to terminal nodes. This could be useful for larger cable systems where the user otherwise has little control of the nodes. For cables, as default, the sequence rule of ATP is followed; the cables are sorted so that the one with most conductors comes first (the user should try to follow this rule as well). All cores are then numbered first followed by the sheaths and the armors.

9.1 Line Check

First, the user selects the line he wants to test and then clicks on *ATP|LineCheck* as shown in Fig. 9.3. Then the input/output selection dialog box shown in Fig. 9.4 appears.

The LineCheck feature in ATPDraw supports up to 3 circuits. ATPDraw suggests the default quantities. The leftmost nodes in the circuit are suggested as the input nodes, while the rightmost nodes become the output. The circuit number follows the node order of the objects. For all standard ATPDraw components the upper nodes has the lowest circuit number. The user also has to specify the power frequency where the line/cable is tested. Finally, the user can check the *Exact phasor equivalent* button which will result in a slightly better results for long line sections.

When the user clicks on OK in Fig. 9.4 an ATP-file (`/LCC/LineCheck.dat`) is created and ATP executed. For a 3-phase configuration 4 sequential data cases are created (Z+, Y+, Z0, Y0) while for a 9-phase configuration 24 cases are created (Z11+, Y11+, Z110, Y110, Z12…, Z22…, Z13…, Z23…, Z33…), since symmetry is assumed. Finally the entire LIS-file is scanned. The calculated values are then presented in result window as shown in Fig. 9.5. The user can switch between polar and complex coordinates and create a text-file of the result. The mutual data are presented on a separate page. The unit of the admittances is given in Farads or Siemens (micro or nano) and the user can scale all values by a factor or by the length.

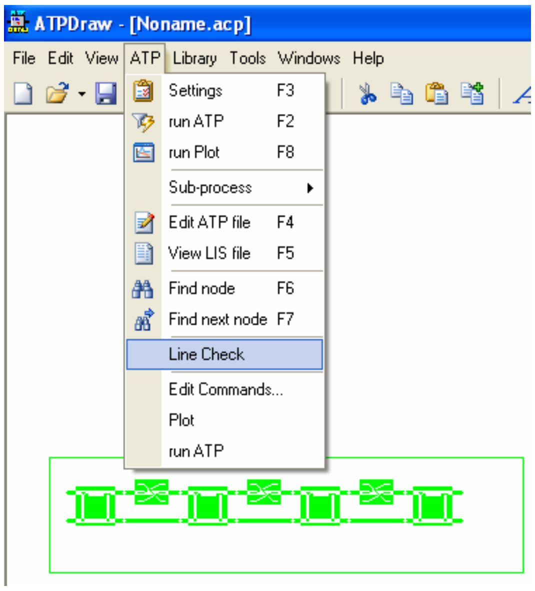
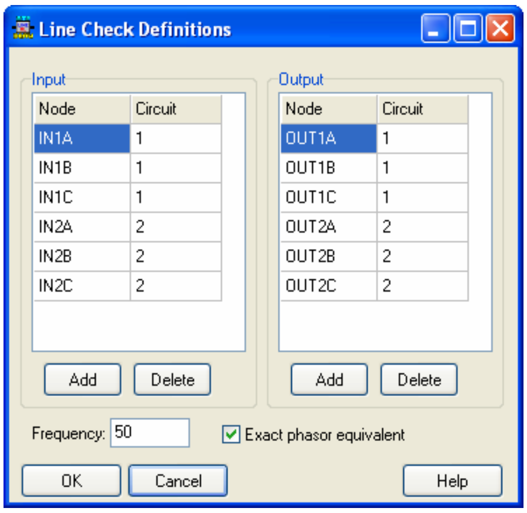


Fig. 9.3 Selecting a line.



Input

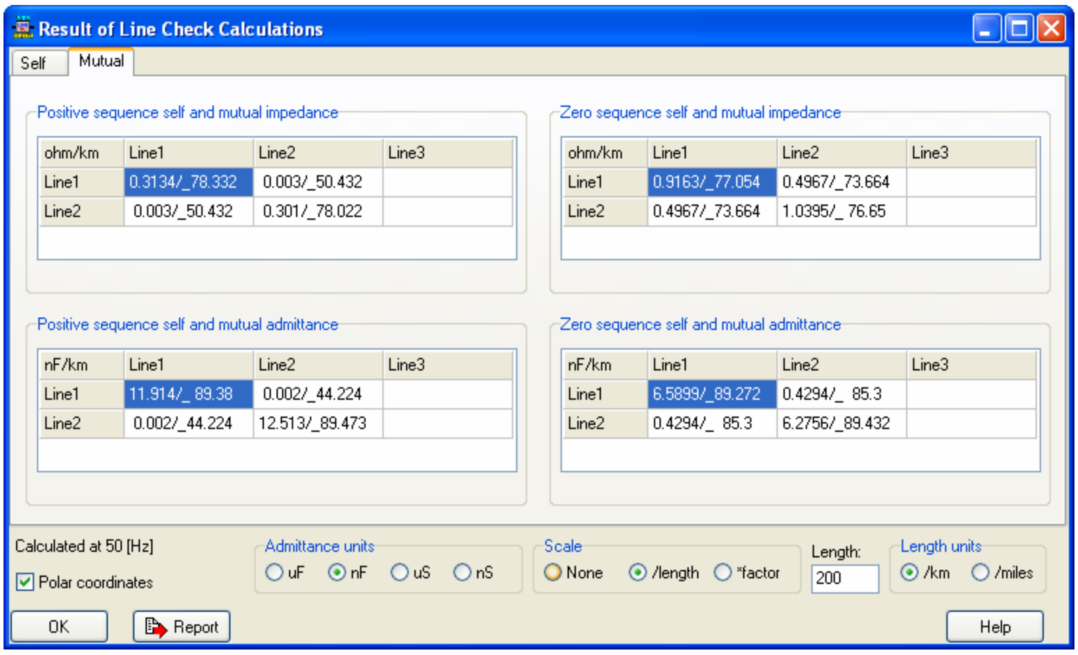
Node	Circuit
IN1A	1
IN1B	1
IN1C	1
IN2A	2
IN2B	2
IN2C	2

Output

Node	Circuit
OUT1A	1
OUT1B	1
OUT1C	1
OUT2A	2
OUT2B	2
OUT2C	2

Frequency: 50
[✓] Exact phasor equivalent

Fig. 9.4 Selecting the input and output.



Positive sequence self and mutual impedance

ohm/km	Line1	Line2	Line3
Line1	0.3134/_78.332	0.003/_50.432	
Line2	0.003/_50.432	0.301/_78.022	

Zero sequence self and mutual impedance

ohm/km	Line1	Line2	Line3
Line1	0.9163/_77.054	0.4967/_73.664	
Line2	0.4967/_73.664	1.0395/_ 76.65	

Positive sequence self and mutual admittance

nF/km	Line1	Line2	Line3
Line1	11.914/_ 89.38	0.002/_44.224	
Line2	0.002/_44.224	12.513/_89.473	

Zero sequence self and mutual admittance

nF/km	Line1	Line2	Line3
Line1	6.5899/_89.272	0.4294/_ 85.3	
Line2	0.4294/_ 85.3	6.2756/_89.432	

Calculated at 50 [Hz]
[✓] Polar coordinates
Admittance units: uF [✓ nF] uS nS
Scale: None [✓ /length] *factor
Length: 200
Length units: [✓ /km] /miles

Fig. 9.5 Presentation of the results.

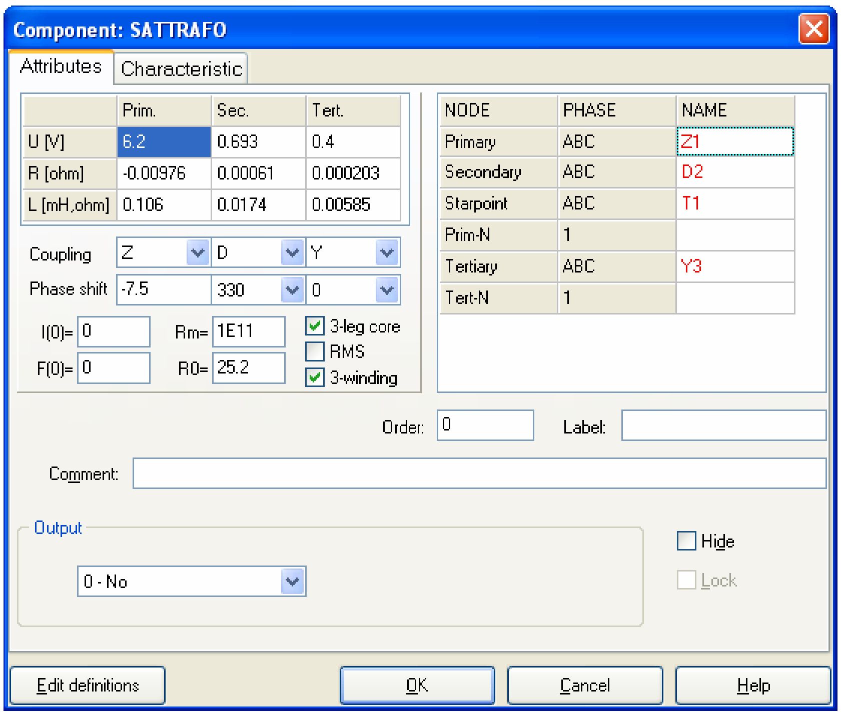
The series impedances are obtained by applying 1 A currents on the terminals and the output ends are grounded (the other circuits are left open and unenegized). For mutual coupling, 1 A is applied at both circuits. On the other hand the shunt admittances are obtained by applying a voltage source of 1 V at one terminal leaving the output end open. For mutual coupling, 1V is applied at one circuit while a voltage of 1E-20 is applied at the other.

Special attention must be paid to long lines and cables. This applies in particular to PI-equivalents. Usage of Exact phasor equivalent is recommended, but is no guarantee of success. No attempt is made in ATPDraw to obtain a better approximation since the line/cable system to be tested in general is unknown. The mutual coupling in the positive sequence system is in symmetrical cases very small and vulnerable to the approximations made.

10 Transformers

10.1 Saturable transformer

A new component called SATTRAFO is introduced in ATPDraw from version 4.0. This is a general 3-phase saturable transformer component with 2 or 3 windings. The component completely replaces the old GENTRAFO and that component is automatically replaced by SATTRAFO when loading an older circuit. The new SATTRAFO component supports all phase shifts between Y- and D- windings (not just Y, Dlead, Dlag, Y180 as in GENTRAFO) as well as Autotransformers and zigzag windings. For a zigzag winding the user can specify the phase shift <-60,0> and <0,60> degrees and the voltage and short circuit impedance distribution is automatically calculated by ATPDraw. The phase shift is specified related to a Y-winding. ATPDraw does not recalculate the magnetizing branch and zero sequence reluctance. This must be done manually by the user. The internal 3-phase node of the zigzag winding is given a name Txxxx where the number xxxx is the incremented transformer number. Several zigzag windings are supported in a single transformer. The dialog box of the new SATTRAFO component is shown in Fig. 10.1.



	Prim.	Sec.	Tert.
U [V]	6.2	0.693	0.4
R [ohm]	-0.00976	0.00061	0.000203
L [mH,ohm]	0.106	0.0174	0.00585
Coupling	Z	D	Y
Phase shift	-7.5	330	0

I(0)= 0 Rm= 1E11 ✓ 3-leg core
F(0)= 0 R0= 25.2 ☐ RMS ✓ 3-winding

NODE	PHASE	NAME
Primary	ABC	Z1
Secondary	ABC	D2
Starpoint	ABC	T1
Prim-N	1	
Tertiary	ABC	Y3
Tert-N	1	

Order: 0 Label:

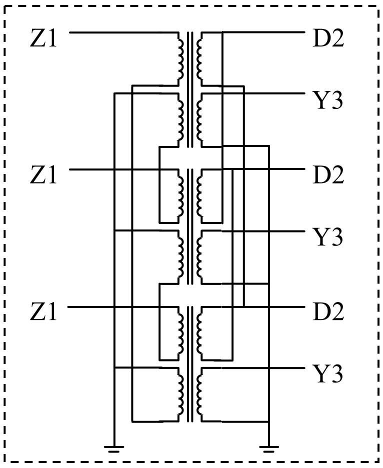
Comment:

Output: 0 - No ☐ Hide ☐ Lock

Fig. 10.1 ATPDraw dialog box of the new SATTRAFO component.

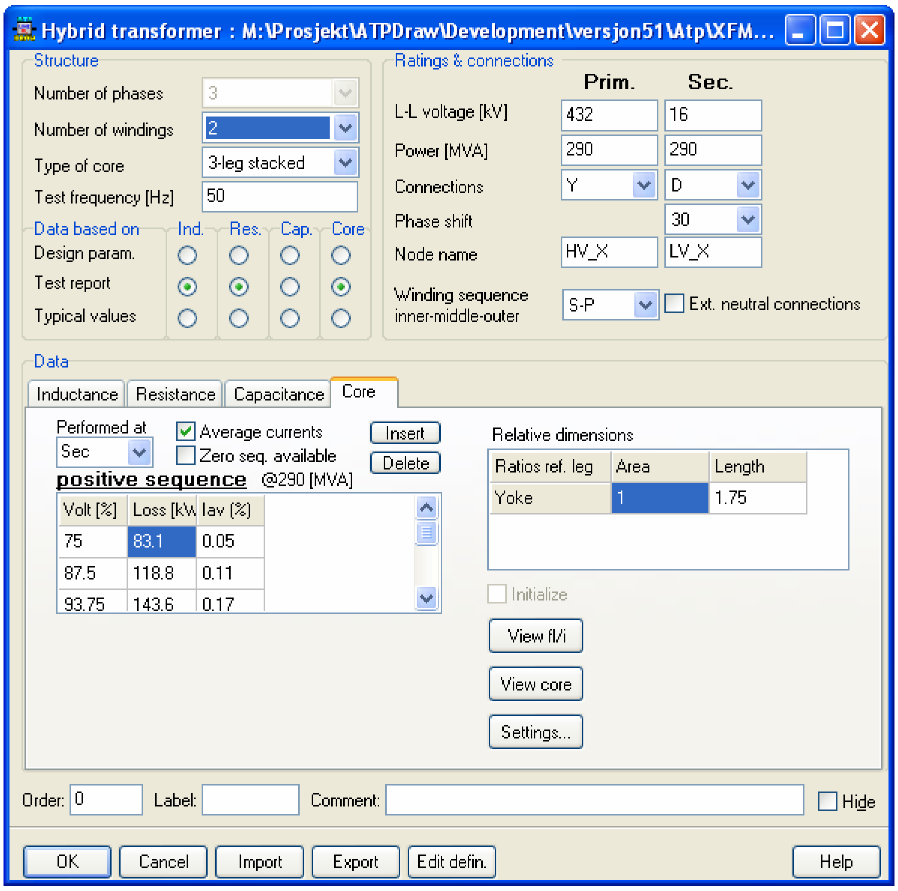
The data specified in Fig. 10.1 will produce an ATP-file as shown below:

```
 TRANSFORMER THREE PHASE TX0001  25.2
 TRANSFORMER                         T1A    1.E11                              0
             1.            1.28
           9999
1Z1A   T0002C            -.0084.103215.6797
2      T0002A            -.0014.00279.93446
3D2A   D2C               .00061 .0174  .693
4Y3A                      .0002.00585    .4
 TRANSFORMER T1A                     T1B
1Z1B   T0002A
2      T0002B
3D2B   D2A
4Y3B
 TRANSFORMER T1A                     T1C
1Z1C   T0002B
2      T0002C
3D2C   D2B
4Y3C
```



10.2 XFMR

The Hybrid Transformer XFMR is an advanced model with a topologically correct core with individual saturation of legs and yokes based on relative core dimensions and the Frolich equation. The model can be based on Design parameters, Test reports and Typical text book values. The input dialog of this component is shown in Fig. 10.2.


Hybrid transformer : M:\Prosjekt\ATPDraw\Development\versjon51\Atp\XFM...

Structure

Number of phases	3
Number of windings	2
Type of core	3-leg stacked
Test frequency [Hz]	50

Data based on	Ind.	Res.	Cap.	Core
Design param.	○	○	○	○
Test report	●	●	○	●
Typical values	○	○	○	○

Ratings & connections

	Prim.	Sec.
L-L voltage [kV]	432	16
Power [MVA]	290	290
Connections	Y	D
Phase shift		30
Node name	HV_X	LV_X

Winding sequence inner-middle-outer: S-P ☐ Ext. neutral connections

Data

Inductance | Resistance | Capacitance | Core

Performed at: Sec ☑ Average currents ☐ Zero seq. available Insert Delete

positive sequence @290 [MVA]

Volt [%]	Loss [kW	Iav [%]
75	83.1	0.05
87.5	118.8	0.11
93.75	143.6	0.17

Relative dimensions

Ratios ref. leg	Area	Length
Yoke	1	1.75

☐ Initialize

View fl/i View core Settings...

Order: 0 Label: Comment: ☐ Hide

OK Cancel Import Export Edit defin. Help

Fig. 10. 2 Input dialog of the Hybrid Transformer model, XFMR.

The name Hybrid model comes from the fact that the model is modular with separate handling of leakage inductance, winding resistance, capacitance and core. The source of data could be *Design*, *Test report* and *Typical values*. Using the *Test report* input with no capacitance (click with right mouse button) and a Triplex core should give results comparable to BCTRAN. The XFMR component gets its default settings from the XFMR.sup file stored in ATPDraw.scl. The XFMR component has a 3-phase node for each winding and a single phase node for the neutral. The notation primary, secondary, tertiary, and core winding is used, called PSTC from now on. The identities and the positions of the nodes are specified in the standard component support file XFMR.sup. Note that the core actually has an external node (where the main flux could be measured). This support file also contains 21 data parameters which mainly serves as initial values for the transformer rating, two-winding test report, yoke and limb relative lengths and areas.

The externally connected winding resistances require additional nodes. This is solved by using 3-phase nodes with index 'OPQ', 'RST', 'UVW', and 'XYZ' in addition to the standard extension 'ABC'/'DEF'.

The overall node structure of the XFMR component in the final ATP file is shown in Fig. 10.3.

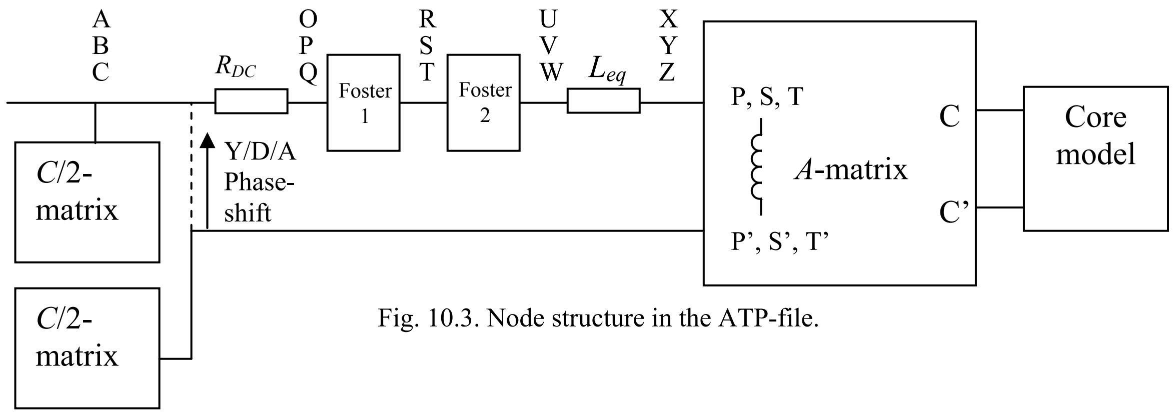


Fig. 10.3. Node structure in the ATP-file.

The following restrictions apply to the node naming:

- It is not legal to ground the nodes directly
- It is not legal to connect two or more transformers directly to the same bus.

The user is requested to connect switches to the transformer in such cases.

In order to establish the artificial core winding the user must specify the sequence of the windings from the core. Normally the winding with the lowest voltage is the inner. The connection point of the core winding is established based on the leakage channel between the inner winding and the core.

When the user exits the XFMR dialog box ATPDraw calculates the required transformer model data (matrices and core elements) based on the user input (design, test report, typical values, or none) and stores the result in the *.xfr data file also containing the input data. No calculations are performed when the final ATP-file is written.

10.2.1 Inverse inductance matrix, *A*

The A-matrix is written to the ATP file with the AR notation with the resistances set to zero. The matrix has dimension $(nw+1)\cdot np$ where nw is the number of physical windings, and the core is connected to the nw +1 winding, and np is the number of phases. In ATPDraw the coupling and phase shift is produced directly when writing the A-matrix. All possible phase shifts are supported. The A-matrix has the following structure for a three phase transformer

$$A = \begin{array}{c} \begin{array}{ccc} \text{A} & \text{B} & \text{C} \end{array} \\ \begin{bmatrix} A_w & 0 & 0 \\ 0 & A_w & 0 \\ 0 & 0 & A_w \end{bmatrix} \end{array} \text{ where } A_w = \begin{array}{c} \begin{array}{cccc} \text{P} & \text{S} & \text{T} & \text{C} \end{array} \\ \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix} \end{array} \text{ for a 3-winding transformer} \qquad (1)$$

The A-matrix is assumed to have no mutual coupling between the phases. The entire zero-sequence system is modelled in the core part. The A_w-matrix is established according to the BCTRAN approach from the EMTP RuleBook and section 5.2.4 p. 31 in MTU7 [4].

Typical values

The leakage reactance is established from table 1, p. 28 in MTU7 [4] (from J.J. Graininger & W.D. Stevenson: *Power System Analysis*, McGraw-Hill 1994) using the lowest value in the typical range. If the type of cooling is unknown forced-air cooling is assumed. In the case of a three-winding transformer the leakage reactance between the inner and outer winding is approximated as the sum of the other two.

X_{PS} [pu] = interpX(table1, max(U_P,U_S), Cooling)/100; (2)
X_{ST} [pu] = interpX(table1, max(U_S,U_T), Cooling)/100;
X_{PT} [pu] = X_{PS}+X_{ST};

It is assumed that the secondary winding is the middle winding.

Test report

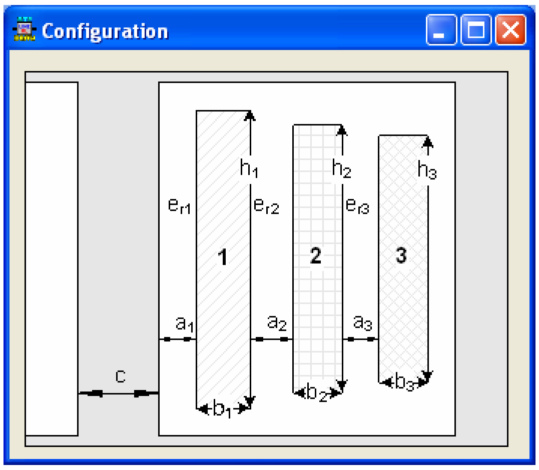
The leakage reactance is calculated from the standard test report data (positive sequence).

$$X[pu] = \sqrt{Z[\%]^2 - (P[kW]/(10 * S[MVA]))^2}\ /100 \qquad (3)$$

In the case of an autotransformer the reactances are scaled according to the TheoryBook [7].
X_{ST} =1/(U_P-U_S)*(X_{ST}*U_P*U_S/(U_P-U_S)+X_{PS}*U_P-X_{PT}*U_S); (4)
X_{PS} =X_{PS}*sqr(U_P/(U_P-U_S));

Design data

The leakage reactances are calculated according to MTU7 [4]. Pancake windings can be arranged in various ways, but the simple layout illustrated in MTU7 [4] Fig. 9 and 10 is assumed. The reactances are scaled to per unit based on the voltage across the PST windings.



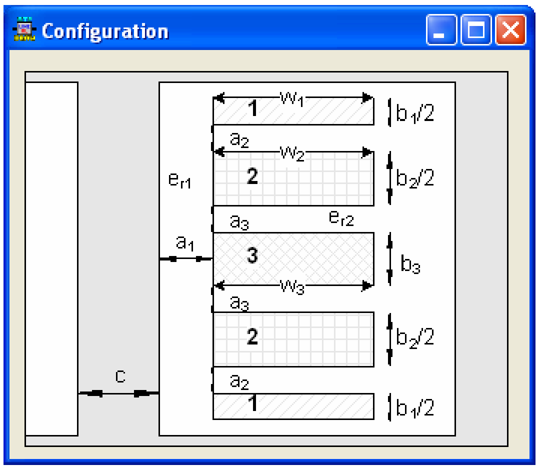


Fig. 10.4. Configuration of the winding (left: cylindrical, right: pancake)

Cylindrical:

$$X_{HM} = \frac{\omega\mu_0 N^2 L_{mt}}{h} \cdot \left(\frac{L_{mt,H} \cdot b_H}{3} + \frac{L_{mt,M} \cdot b_M}{3} + L_{mt,3} \cdot a_3 \right) \tag{5}$$

$$X_{ML} = \frac{\omega\mu_0 N^2 L_{mt}}{h} \cdot \left(\frac{L_{mt,M} \cdot b_M}{3} + \frac{L_{mt,L} \cdot b_L}{3} + L_{mt,2} \cdot a_2 \right) \tag{6}$$

$$X_{HL} = \frac{\omega\mu_0 N^2 L_{mt}}{h} \cdot \left(\frac{L_{mt,H} \cdot b_H}{3} + \frac{L_{mt,M} \cdot b_M}{3} + L_{mt,3} \cdot a_3 + L_{mt,2} \cdot a_2 + L_{mt,M} \cdot b_M \right) \tag{7}$$

The indexes H, M and L refer to the outer, middle, and inner winding respectively. h is the average height of the two windings involved, L_{mt} is the equivalent circumference of the winding or leakage channel ($2\pi \cdot r$), a and b is the width of the leakage channel and winding respectively.

Pancake:

$$X_{HM} = \frac{\omega\mu_0 N^2 L_{mt}}{2nW} \cdot \left(\frac{b_H}{6} + \frac{b_M}{6} + a_3 \right), \quad X_{LM} = \frac{\omega\mu_0 N^2 L_{mt}}{2nW} \cdot \left(\frac{b_M}{6} + \frac{b_L}{6} + a_2 \right) \tag{8}$$

$$X_{HL} = \frac{\omega\mu_0 N^2 L_{mt}}{2nW} \cdot \left(\frac{b_H}{6} + \frac{b_L}{6} + \frac{b_M}{2} + a_2 + a_3 \right) \tag{9}$$

W is the average width of the two windings involved. a height of the leakage channel and b is the total height of the winding.

Handling of the core winding

The core winding is related to the leakage channel between the inner physical winding and the core. A parameter $K=a_1/a_2$ is used in MTU6/7 [3,4] with a_1 is the width of the inner leakage channel and a_2 is the width of the leakage channel between the inner and the outer/middle winding. Various values are used in the MTU reports, but a fixed value K=0.5 is used in ATPDraw. If the pu leakage reactances X_{ML}, X_{MH}, and X_{HL} (L=inner, M=middle, H=outer) for a three winding transformer are given then the leakage reactances to the core winding are assumed to be

$X_{LC} \approx K \cdot X_{ML}$, $X_{MC} \approx X_{LC} + X_{ML} = (K+1) \cdot X_{ML}$, and

$$X_{HC} \approx X_{MC} + X_{HM} = (K+1) \cdot X_{ML} + X_{HM} \tag{10}$$

10.2.2 Winding resistance

If the user selects a frequency dependent winding resistance the resistance is approximated by a two-cell Foster equivalent. A negative compensating (minimizing the imaginary part) inductance L_{eq} is added in series.

Typical values

The typical winding resistances (at power frequency) are in principle based in the MTU7 [4] table 5 at page 53 (from A. Greenwood: *Electrical Transients in Power Systems*, Wiley 1991). However, since the kV and kVA values tabulated are rather limited (up to 50 MVA and 230 kV) a simple interpolation approach as used for the reactance can not be utilized here. Instead a function *Rw* is established that takes in the parameter *u* [kV] and *s* [MVA] and returns the resistance in %. Data for a 290 MVA/ 430 kV transformer was used along with table 5 to calibrate the function:

$$Rw = 0.7537 \cdot \left(\frac{u}{15}\right)^{0.0859} \cdot s^{-0.2759} \quad [\%] \tag{11}$$

Test report

The test report data are given at power frequency. The per unit short circuit resistances are calculated from the test report data (positive sequence)

$$\begin{aligned} R_{PS} &= P_{PS}[\mathrm{kW}]/(1000 * S_{PS}[\mathrm{MVA}]) \text{ [pu]} \\ R_{ST} &= P_{ST}[\mathrm{kW}]/(1000 * S_{ST}[\mathrm{MVA}]) \text{ [pu]} \\ R_{PT} &= P_{PT}[\mathrm{kW}]/(1000 * S_{PT}[\mathrm{MVA}]) \text{ [pu]} \end{aligned} \tag{12}$$

The winding resistance is assumed to be equally shared between the windings in the case of a two-winding transformer (R_P=R_{PS}/2 and R_S= R_{PS}/2. In the case of a 3-winding transformer the traditional star-equivalent approach is used:

$$\begin{aligned} R_P &= (R_{PS}+R_{PT}-R_{ST})/2 \text{ [pu]} \\ R_S &= (R_{PS}+R_{ST}-R_{PT})/2 \text{ [pu]} \\ R_T &= (R_{PT}+R_{ST}-R_{PS})/2 \text{ [pu]} \end{aligned} \tag{13}$$

In the case of an auto-transformer the short circuit resistances are scaled according to eq. (51) in MTU7 [4]. The approach used for reactances (from the Theory Book [7]) did not work out for the resistances.

Design data

The user can specify the winding conductivity σ, the equivalent cross section of each turn A, the average length of each turn L, number of turns of the inner winding, and the rated voltages of all windings. The DC resistance is then simply calculated as in eq. (36) p. 51 of MTU7 [4] and scaled to the power frequency using the frequency dependency below (with *nl*=1). The user has to specify the conductor height and width and the number of conductors in parallel. In the resistance is assumed to be frequency independent only the total area matters (product of width, height and #parallel).

$R_{DC} = \dfrac{N \cdot L}{\sigma \cdot (wt \cdot ht \cdot np)}$ where N is the number of turns, L is the average length per turn, *wt* and *ht* is the width and height of each conductor respectively, and *np* is the number of parallel conductors.

Frequency dependency

The frequency dependent resistance is calculated at 11 logarithmically space frequency points from 0.1 to 10 kHz. The frequency behaviour is then fitted with a two-cell Foster equivalent.

The typical values and test report resistances are assumed to follow $R(f) = R_0 \cdot \sqrt{f/f_0}$ where R_0 is the resistance at the power frequency f_0. This expression results in considerably lower values that suggested in Fig. 27 in MTU7 [4]. This needs to be further investigated. In version 5.3 it was realized that the imaginary part of the Foster cells needs to be minimized as well. This resulted in poorer fitting for the real part.

The design data resistances are assumed to follow eq. (37) is MTU7 [4] that in ATPDraw is formulated as (Delphi does not support complex numbers) (is there a sign mistake somewhere?):

$$u = wt \cdot \sqrt{\pi \cdot f \cdot \sigma \cdot \mu_0}$$

$$R(f) = R_{DC} \cdot u \cdot \left[\frac{\sinh(u) \cdot \cosh(u) + \sin(u) \cdot \cos(u)}{\cosh^2(u) - \cos^2(u)} + \frac{2}{3} \cdot \frac{\sinh(u) \cdot \cosh(u) - \sin(u) \cdot \cos(u)}{\sinh^2(u) + \cos^2(u)} + \right.$$
$$\left. \frac{2}{3} \cdot nl^2 \cdot \frac{\sinh(u/2) \cdot \cosh(u/2) - \sin(u/2) \cdot \cos(u/2)}{\sinh^2(u/2) + \cos^2(u/2)} \right] \tag{14}$$

Here *wt* is the width of a single conductor. Further *nl* is the number of layers in the winding which should be set to unity for a disk winding. The above expression approaches $\sim \sqrt{f}$ at frequencies higher than approximately twice the power frequency.

The eleven *R*(*f*) and *f* value pairs are fitted to the function (two-cell Foster equivalent) $R(\omega) = R_0 + \dfrac{R_1 \cdot \omega^2 \cdot L_1^2}{R_1^2 + \omega^2 \cdot L_1^2} + \dfrac{R_2 \cdot \omega^2 \cdot L_2^2}{R_2^2 + \omega^2 \cdot L_2^2}$ with the resistances R_1 and R_2, and inductances L_1 and L_2 as unknowns. The fitting routine is based on the Lavenberg-Marquardt method and a simplified version of `mrqmin` from Numerical Recipes, ver. 2. This routine further uses a Gauss-Jordan routine for solving a linear system of equations. A constraint is put on the total inductance $L_1 + L_2 < C$ where C is set as the inverse of the A_w-matrix element, MTU7 [4] eq. (35). The constraint is handled simply by setting L_1=L_2=0.5·C when the constraint is violated and then continue to obtain new optimized values for R_1 and R_2.

10.2.3 Capacitance

The *C*-matrix is split in two halves and connected to each terminal of the transformer. The type 1, 2 … PI-equivalent is used with the series resistance set to infinity (1e8 Ω) to accomplish this. The capacitance matrix *C* is based on the following two matrices

$$C_w = \begin{bmatrix} C_{11} & C_{12} & C_{13} \\ C_{21} & C_{22} & C_{23} \\ C_{31} & C_{32} & C_{33} \end{bmatrix} \text{ and } C_p = \begin{bmatrix} C_{AA} & C_{AB} & C_{AC} \\ C_{BA} & C_{BB} & C_{BC} \\ C_{CA} & C_{CB} & C_{CC} \end{bmatrix} \qquad (15)$$

The C_w matrix contains the capacitances between windings 1-3 that are equal for all phases. The C_p matrix contains capacitances that are specific to phase A, B, or C. These are typically connected to the outer windings. The total *C*-matrix is then built on these two matrices dependent on the type of winding (pancake/cylindrical).

Typical values

A capacitive coupling factor K_c can be specified by the user with a default value of 0.3. The MTU reports use the concept of transient recovery voltage (TRV) to obtain the effective capacitance when the inductance is known. The IEEE standard C37, Fig. B2 is used to obtain the frequency of the TRV for a known voltage level and fault current. In order to use these curves in ATPDraw a fitting function had to be established. The function is on the form

$$C_{eff}(U,S,X_{pu},f) = \frac{f}{2\pi}\cdot\frac{3\cdot I}{U\cdot\left(f_{TRV}(U,I)\right)^2} \text{ [μF]} \qquad (16)$$

with U in [kV], S in [MVA] and $I = \dfrac{S}{3\cdot U\cdot X_{pu}}$ [kA]

The essential fitting function is approximated by

$$f_{TRV}(U,I) = 31.6\cdot U^{-0.469} + 45.9\cdot U^{-0.55}\cdot I^{\frac{1}{1+0.552\cdot U^{0.062}\cdot\log(I)}} \text{ [kHz]} \qquad (17)$$

In the case of typical values the C_p matrix (between phases) is always set to zero in lack of any better choice.

For a two-winding transformer the C_w matrix is calculated as

$$\begin{aligned} C_w[1,2] &= C_{PS} = K_c\cdot C_{eff}(U_S,S,X_{PS,pu},f) \\ C_w[1,1] &= C_{PP} = C_{eff}(U_P,S,X_{PS,pu},f) - C[1,2] \\ C_w[2,2] &= C_{SS} = (1/K_c - 1)\cdot C[1,2] \end{aligned} \qquad (18)$$

For a three winding transformer the typical capacitance is more complicated. Here a simple approach is used and the two other coupling factors used in MTU7 [4], p. 34 are avoided. Instead the following relations are used (This approach could be further discussed):

$$\begin{aligned} C_w[1,3] &= C_{PT} = 0 \\ C_w[2,3] &= C_{ST} = C_{eff}(U_S,S,X_{ST,pu},f) - C[2,2] \\ C_w[3,3] &= C_{TT} = C_{eff}(U_T,S,X_{ST,pu},f) - C[2,3] \end{aligned} \qquad (19)$$

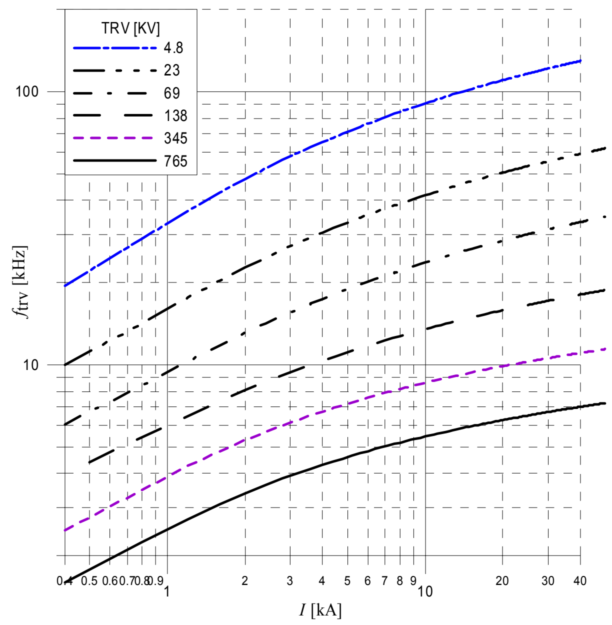


Fig. 10.5. Result of the fitting function above. To be compared to IEEE C37, Fig. B2.

Test report

In the test report the capacitances between each winding and ground and between all windings is assumed to be directly specified while the C_p matrix is set to zero. All values must be specified per phase. An option for specifying C_p is built into the code, but this is hidden at the moment due to lack of a test case.

Design data

The calculation of design data capacitances are based on MTU7 [4] chapt. 5.3, p. 33-42. The user has to specify the winding geometry as well as the various equivalent permittivities of insulation system. The same configuration as given in Fig. 10.4 is used here.

Cylindrical windings:
Capacitance between two windings:

$$C_{H-L} = \frac{2\pi\varepsilon_r \cdot \varepsilon_0 \cdot (h + \Delta h)}{\ln\left(\dfrac{d_{H,i}}{d_{L,o}}\right)} \tag{20}$$

The subscript syntax is H=outer, L=inner (the capacitance between the inner and outer in a three-winding situation is assumed to be zero). Where h is the average height of the two windings involved and Δh is the width of the leakage channel between them (added to compensate for end effects). $d_{H,i}$ is the inner diameter of the outer winding, while $d_{L,o}$ is the outer diameter of the inner winding. The capacitance between the inner winding and the core is approximated the same way, but with Δh set to the width of the inner leakage channel (the capacitance between the other windings and the core is ignored).

Capacitance between the outer windings of adjacent legs:

$$C_{HA-HB} = \frac{\pi\varepsilon_r \cdot \varepsilon_0 \cdot (h + \Delta h)}{\ln\left(\frac{p}{d_{H,o}} + \sqrt{\left(\frac{p}{d_{H,o}}\right)^2 - 1}\right)} \tag{21}$$

Where p is the phase distance and $d_{H,o}$ is the outer diameter of the outer winding. In this case the end effect factor becomes $\Delta h = p - d_{H,o}$.

The remaining capacitances considered are the one between the outer winding and the tank. The capacitance between the end phase (outer winding) and the tank is calculated from (20) replacing $d_{H,i}$ and $d_{L,o}$ by $d_{H,o}$ and t, respectively, where t is the tank diameter. The capacitance is further multiplied by a factor 0.75 to account for a non-circular tank. The capacitance between the middle phase (outer winding) and the tank is calculated from (21) replacing p by t, setting Δh to the distance between the outer winding surface and the tank wall $\Delta h = (t - d_{H,o})/2$. In this case the capacitance is multiplied by a factor 0.25.

Pancake windings:
Capacitances between all windings and the core are calculated by (20) replacing h by the winding width b, and Δh by the width of the inner leakage channel, a_1.

The capacitance between two windings is calculated by a parallel plate approach:

$$C_{H-L} = \frac{\varepsilon_r \cdot \varepsilon_0 \cdot A}{a} = \frac{\varepsilon_r \cdot \varepsilon_0 \cdot \pi \cdot \left[\left(\frac{d_{L,o}}{2}\right)^2 - \left(\frac{d_{L,i}}{2}\right)^2\right]}{a} \cdot 2 \cdot np \tag{22}$$

where a is the height of the leakage channel between the windings, and np is the number of pancakes. $d_{L,i}$ is the inner diameter of the L-winding (c+2·a_1 in Fig. 10.4).

The capacitance between the adjacent phases is assumed to be between windings 1 (L=inner) and this can be calculated by (22) replacing a by the distance between the outer parts of the windings.

Capacitances between windings and the tank are approximated by (20) replacing h+Δh by the total winding width b. (the number of pancakes is included in b). Further $d_{H,i}$ is replaced by the tank width t, and finally a reduction factor 0.25 is applied. In addition the winding 1 (L=inner) of the outer phases will have a capacitance to the end tank wall. This can be approximated by (22) replacing a by the distance to the tank wall q, and by setting 2·np to unity.

The capacitance matrix C_w is built up like a nodal admittance matrix consisting of those capacitances that are equal for all phases. The capacitance matrix C_p consists of the quantities that are unique for each phase. Thus C_p contains the capacitances between the outer windings on adjacent legs and between these windings and the tank. The concept 'outer winding' will be different for pancake and cylindrical windings.

10.2.4 CORE

The core model is connected to the 3-phase node C in Fig. 10.3. This node is also available externally for measurement of the voltage and flux linkage.

Only stacked cores with three and five legs are supported at this point as shown in Fig. 10.6. Basically the inductive and resistive core parts are treated independently. The core losses are at this point assumed to be linear and the nonlinear inductances are modelled by the Frolich equation. Each part of the core is modelled with its own core loss resistance and nonlinear inductance using information about their relative cross section and length to scale the values.

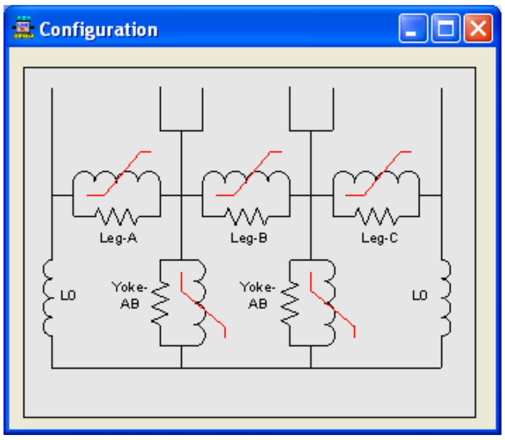


Fig. 10.6) 3-legged stacked core.

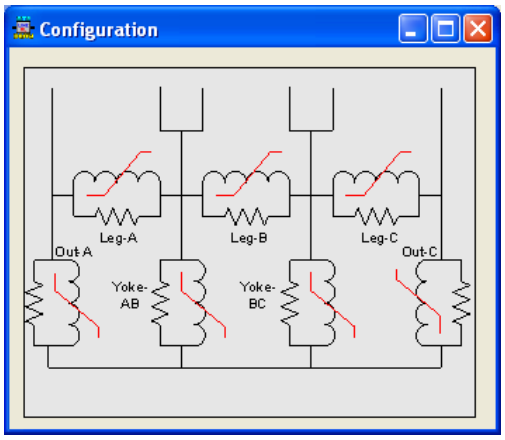


Fig. 10.6b) 5-legged stacked core.

It is assumed that the magnetic material is characterized by four parameters a, b, d and e. A library of typical steel materials is developed (M2, M4, M6, and METGLASS 2605TCA). Nippon&ARMCO steels turned out to be quite similar and are grouped as one. This list is based on fitting the manufacturer's data from state of the art catalogues. Older steel materials will have a different characteristic and the losses are typically higher. The material library will be used for design data and typical values. The *B*/*H* relationship is assumed to follow the Frolich equation $B = \dfrac{H}{a + b \cdot |H|}$ and the specific loss is assumed to follow

$$p\,[W/kg] = \left(\frac{f}{50}\right)^{1.5} \cdot \left(d \cdot B^2 + e \cdot B^{10}\right)$$, where *f* is the power frequency (23)

The specific loss is traditionally (for instance Westinghouse T&D reference book, 1964) assumed to be $p = K_e \cdot (f \cdot t \cdot B_{\max})^2 + K_e \cdot f \cdot B_{\max}^x$ with *x* said to be 3 for modern materials in the year of 1964. In the above expression *t* is the thickness of the laminates. The traditional expression was tested on modern material data with little success.

Fig. 10.7 shows the fit of the specific losses and DC-magnetization curve of ARMCO M4 steel. The Frolich fitting is not very good, and in Fig. 7b) fitting around the knee point (nominal flux) was preferred at the sacrifice of high field fitting (*B*=1.9 T). Similar fitting is performed for the other core materials.

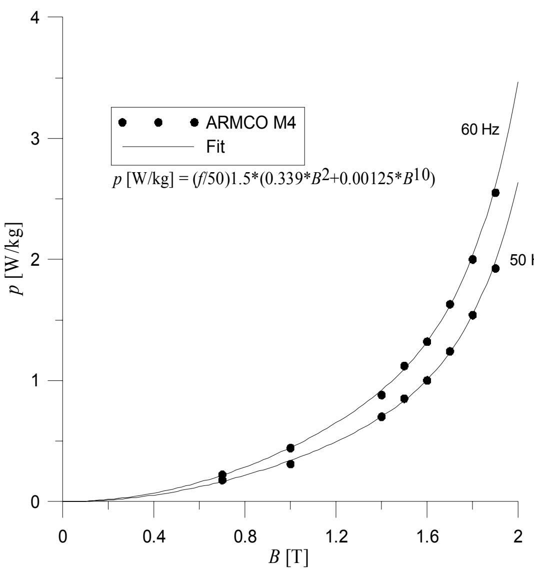


Fig. 10.7a) Core loss curves

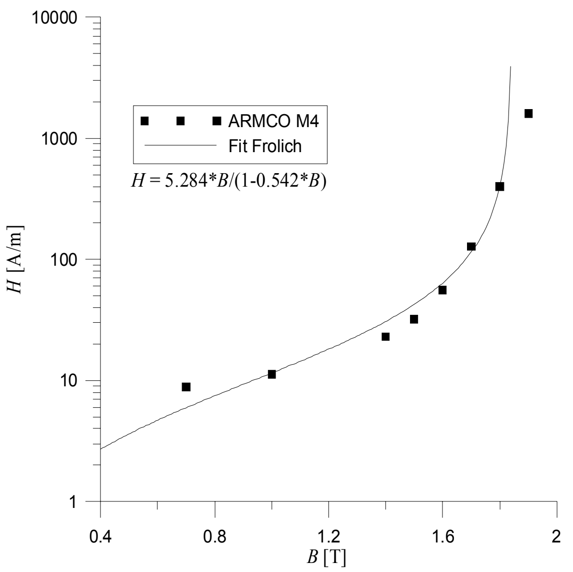


Fig. 10.7b) DC-magnetization curve

As explained in the MTU7 [4] report the essence of the inductive core modelling is to obtain the two values *a'* and *b'* in the Frolich equation for the λ–*i* characteristic by an optimization technique. The measured or assumed rms value of the magnetizing current is compared with a calculated quantity and iterated until a best fit is found. A method called Golden Search was originally implemented in ATPDraw, but replaced by a Genetic Algorithm from version 5.1. This method is robust and relatively simple to implement, but the optimization process takes a few seconds. An important part of the work was to establish effective constrains for the *a'* and *b'* values.

Inductance modelling:

The basic Frolich equation for the magnetic fields is $B = \dfrac{H}{a + b\cdot|H|}$. The flux linkage is $\lambda = B \cdot A \cdot N$ and the current is $i = H \cdot l / N$ where *N* is the number of turns of the inner winding, *A* is the cross section of the core, and *l* is the length of the core.

As a result of this the Frolich equation for the fluxlinkage-current relationship is defined as $i = l_r \cdot \dfrac{a' \lambda / A_r}{1 - b' \lambda / A_r}$ where l_r and A_r are the relative core length and cross section related to the leg.

The constants to be determined in the optimization process are $a' = a \cdot \dfrac{l_L}{N^2 \cdot A_L}$ and $b' = b \cdot \dfrac{1}{N \cdot A_L}$ based on the absolute length and cross section of the core leg.

The final result is

$\lambda_l = \dfrac{i}{a' + b' i}$ for the leg, $\lambda_o = \dfrac{i \cdot A_{ro}}{a' l_{ro} + b' i}$ for the outer leg, and $\lambda_y = \dfrac{i \cdot A_{ry}}{a' l_{ry} + b' i}$ for the yoke. (24)

The nonlinear inductances are implemented as optional type 93 or type 98 inductances in ATP. Studies performed at MTU have revealed type 98 inductances the as most stable.

Core loss modelling

The basic core loss equations from Fig. 5 and MTU7 [4] when the inductances and resistances are treated separately are

5-legged core:

$$P_{loss} = \frac{3\cdot V^2}{R_l} + \frac{3\cdot V^2\cdot R_o}{2\cdot(R_o+R_y)^2} + \frac{V^2}{2\cdot R_y\cdot(R_o+R_y)^2}\cdot(4\cdot R_y^2 + 2\cdot R_o\cdot R_y + R_o^2)\,,$$

3-legged core:

$$P_{loss} = \frac{3\cdot V^2}{R_l} + \frac{V^2}{2\cdot R_y} \qquad (25)$$

If we assume that the core loss is proportional to the volume of the core we can set the outer leg (R_o) and yoke resistances (R_y) proportional to the leg resistance (R_l):
$R_o = R_l/(A_{rl}\cdot l_{rl})$ and $R_y = R_l/(A_{ry}\cdot l_{ry})$ where l_r and A_r are the relative dimensions.

Inserting this into the P_{loss} equations we get:
5-legged core:

$$P_{loss} = \frac{V^2}{R_l}\cdot\left(3 + \frac{4/(A_{ry}^2\cdot l_{ry}^2) + 5/(A_{ry}\cdot l_{ry}\cdot A_{ro}\cdot l_{ro}) + 1/(A_{ro}^2\cdot l_{ro}^2)}{2/(A_{ry}\cdot l_{ry})\cdot(A_{ry}\cdot l_{ry} + A_{ro}\cdot l_{ro})^{-2}}\right) = \frac{V^2}{R_l}\cdot K_5 \qquad (26)$$

3-legged core:

$$P_{loss} = \frac{V^2}{R_l}\cdot\left(3 + \frac{A_{ry}\cdot l_{ry}}{2}\right) = \frac{V^2}{R_l}\cdot K_3 \qquad (27)$$

This gives the resistances in the core model directly as proportional to the core losses.

Typical values

The estimation of the magnetizing current (I_m) is based on table 9 in MTU7 [4] (from A. Greenwood: *Electrical Transients in Power Systems*, Wiley 1991). Some fitting of the data is performed which results in

$I_m = 0.73\cdot\left(\frac{BIL}{350}\right)^{0.2933}\cdot\left(\frac{s}{20}\right)^{-0.2154}$ [%] when the basic insulation level (*BIL*) is known and (28)

$I_m = 0.855\cdot\left(\frac{u}{150}\right)^{0.2283}\cdot\left(\frac{s}{20}\right)^{-0.2134}$ [%] when *BIL* must be estimated. (29)

BIL is in [kV], *u* is the rated voltage in [kV] and *s* is the rated power in [MVA].

For a typical core model the user has to specify the maximum *B*-field (normally 1.5-1.7 Tesla) and the maximum core loss density. First a core material has to be guessed and this gives the *a* and *b* values in the Frolich equation (and possibly also the *c* and *d* values that would replace p_{max}).

The following relations are then assumed:

$$\lambda_{\max} = \frac{\sqrt{2}\cdot U_{rms}}{\omega} = B_{\max}\cdot A\cdot N \Rightarrow A\cdot N = \frac{\sqrt{2}\cdot U_{rms}}{\omega\cdot B_{\max}} \qquad (30)$$

$$H_{\max} = \frac{a\cdot B_{\max}}{1-b\cdot B_{\max}} = \sqrt{2}\cdot i_{rms}\cdot\frac{N}{l} \Rightarrow \frac{N}{l} = \frac{a\cdot B_{\max}}{(1-b\cdot B_{\max})\cdot\sqrt{2}\cdot i_{rms}} \qquad (31)$$

which a bit doubtfully assumes a sinusoidal magnetizing current.

This gives the parameter of the fluxlinkage-current characteristic:

$$a' = a\cdot\frac{l}{A\cdot N^2} \approx \omega\cdot(1-b\cdot B_{\max})\cdot\frac{i_{rms}}{u_{rms}} \text{ and } b' = b\cdot\frac{1}{A\cdot N} \approx b\cdot\frac{\omega\cdot B_{\max}}{\sqrt{2}\cdot u_{rms}} \qquad (32)$$

We see that the *a'* expression is independent on the magnetic material property *a*.

The core loss is estimated to

$$P_{loss} = p \cdot \rho \cdot A \cdot l = p \cdot \rho \cdot \frac{(1 - b \cdot B_{\max}) \cdot 2 \cdot u_{rms} \cdot i_{rms}}{\omega \cdot a \cdot B_{\max}^2} \tag{33}$$

where p [W/kg] and ρ [kg/m^3] are given the volume $A{\cdot}l$ is estimated.

Test report

The user specifies the excitation voltage in [%], the current in [%] and the core loss in [kW]. The core loss is used directly as explained above to obtain the core resistances. At the moment the core resistances are assumed to be linear and the core loss value at 100 % excitation is used (interpolation implemented if this point is not available).

The inductive magnetizing current for each point is calculated as

$$I_{rms} = \sqrt{I[\%]^2 - \left(\frac{P[kW]}{10 \cdot S[MVA]}\right)^2}\ [\%] \tag{34}$$

This results in a sequence of excitation points (U_{rms} and I_{rms}). If a single point is specified the core is assumed to be linear. If more than one point is specified these are sent to the optimization routine that returns the *a'* and *b'* values used to describe the λ–i relationship for the core inductances as shown above and in MTU7 [4] eq. (44).

Design data

For design data the user specifies the core material directly with is B/H relationship (a and b values in the Frolich equation). The absolute core dimensions and the number of inner-winding turns N are known, so the inductances can be found directly as

$$\lambda_l = \frac{i}{\frac{a \cdot l_l}{A_l \cdot N^2} + \frac{b}{A_l \cdot N} \cdot i} \text{ for the leg, } \lambda_o = \frac{i}{\frac{a \cdot l_o}{A_o \cdot N^2} + \frac{b}{A_o \cdot N} \cdot i} \text{ for the outer leg and}$$

$$\lambda_y = \frac{i}{\frac{a \cdot l_y}{A_y \cdot N^2} + \frac{b}{A_y \cdot N} \cdot i} \text{ for the yoke} \tag{35}$$

Based on manufacturer data the core losses can be established as $p\,[W/kg] = \left(\frac{f}{50}\right)^{1.5} \cdot \left(d \cdot B^2 + e \cdot B^{10}\right)$ with $B = \frac{\sqrt{2} \cdot U_{rms}}{\omega \cdot A \cdot N}$ and known values of the core weight (volume and density) the core loss can be estimated.

References

[3] F. Gonzalez-Molina, D. Ishchenko, B. Mork: *Parameter estimation and advancements in transformer models for EMTP simulations, MTU6: Parameter Estimation*, December 23rd 2003.

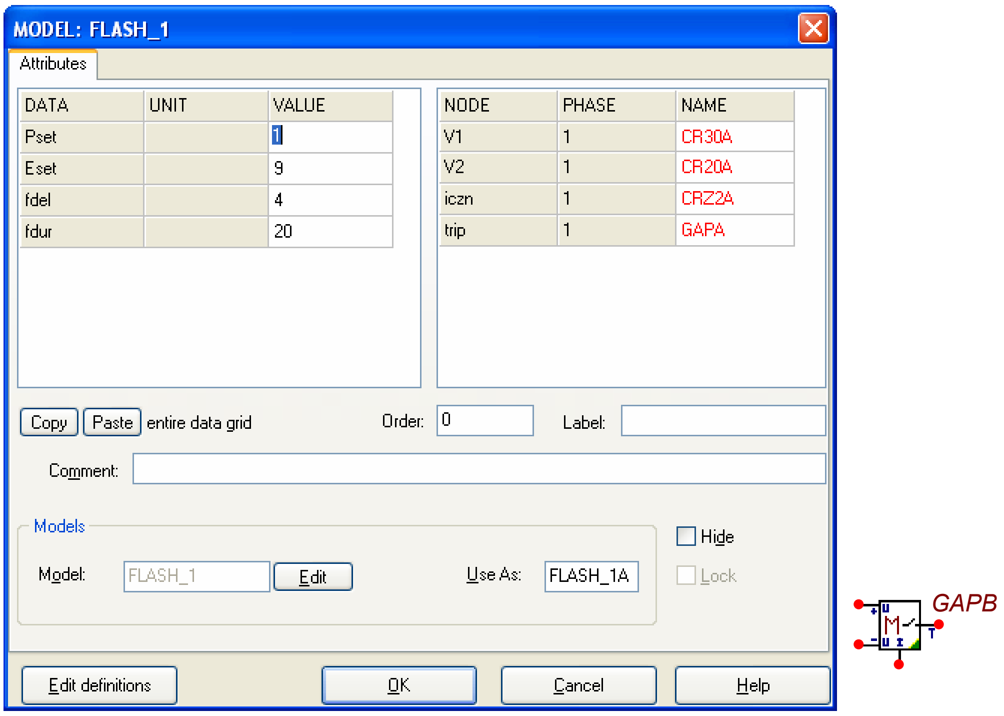
[4] F. Gonzalez-Molina, D. Ishchenko, B. Mork: *Parameter estimation and advancements in transformer models for EMTP simulations, MTU7: Model Performance and Sensitivity Analysis*, June 22nd 2004.

11 Electrical machines

The type 56 electrical machine is supported from version v5.

12 MODELS

The handling of Models has been fundamentally changed. In v5 of ATPDraw the user can edit the Model from within ATPDraw. The icon will automatically adapt to changes in the Model's header (name, inputs, outputs, and data). No files are involved in the process of working with Models, but the user can optionally base a Model on pre-existing support and mod- files. The Model's icon is preferably based on a default vector graphics symbol with inputs on the left side and outputs on the right side. The default icon contains the text MODEL and the name of the model. When a Model is copied the data (the Model's text, previously called the mod-file) is of course also copied. If data in one of the Model is changed, this will also affect the copied data. Actually ATPDraw looks for Models having the same name and copies data over to them after a warning to the user. The name of the Model is written in grey in the component dialog box shown in Fig. 12.1. To actually edit the Model the user clicks on *Edit*. This will bring up the text editor shown in Fig. 12.2. If the user changes the number of inputs or outputs the icon is automatically updated to the default vector graphic icon.



DATA	UNIT	VALUE
Pset		1
Eset		9
fdel		4
fdur		20

NODE	PHASE	NAME
V1	1	CR30A
V2	1	CR20A
iczn	1	CRZ2A
trip	1	GAPA

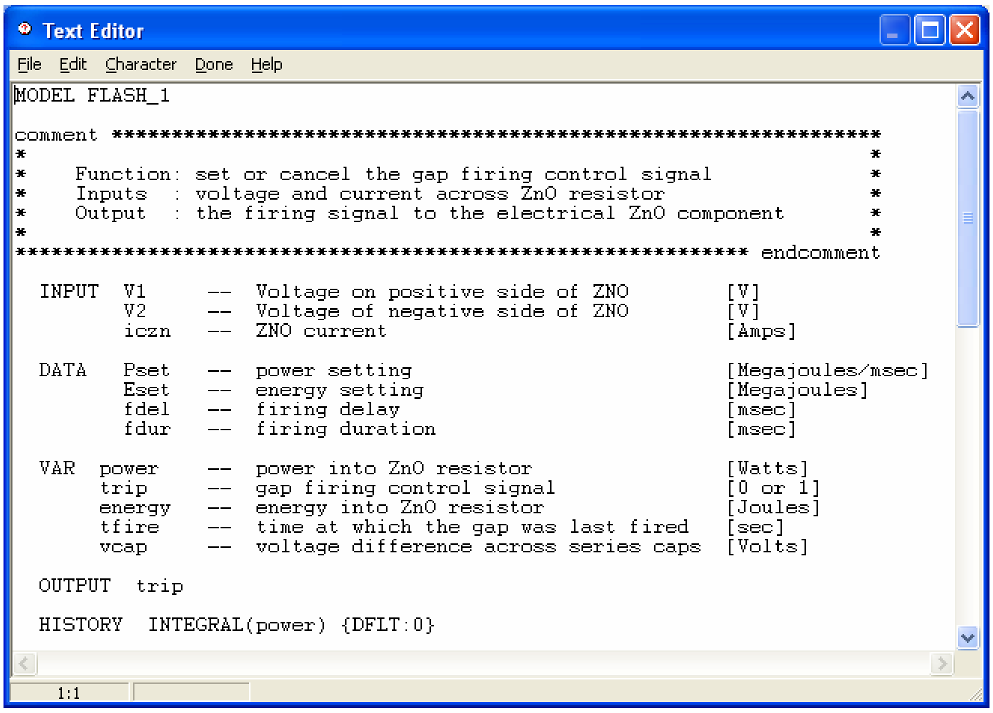
entire data grid Order: 0 Label:

Comment:

Models

Model: FLASH_1 Use As: FLASH_1A

Fig. 12.1. The Model FLASH_1 from the EXA_8.ACP project.

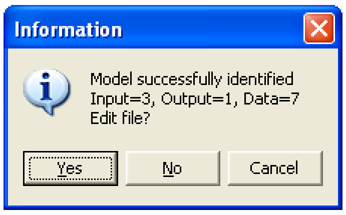


```
MODEL FLASH_1

comment ****************************************************************
*                                                                      *
*    Function: set or cancel the gap firing control signal             *
*    Inputs  : voltage and current across ZnO resistor                 *
*    Output  : the firing signal to the electrical ZnO component       *
*                                                                      *
************************************************************ endcomment

  INPUT  V1     --  Voltage on positive side of ZNO         [V]
         V2     --  Voltage of negative side of ZNO         [V]
         iczn   --  ZNO current                             [Amps]

  DATA   Pset   --  power setting                           [Megajoules/msec]
         Eset   --  energy setting                          [Megajoules]
         fdel   --  firing delay                            [msec]
         fdur   --  firing duration                         [msec]

  VAR   power   --  power into ZnO resistor                 [Watts]
        trip    --  gap firing control signal               [0 or 1]
        energy  --  energy into ZnO resistor                [Joules]
        tfire   --  time at which the gap was last fired    [sec]
        vcap    --  voltage difference across series caps   [Volts]

  OUTPUT  trip

  HISTORY  INTEGRAL(power) {DFLT:0}
```

Fig. 12.2. The text editor for the FLASH_1 model. The Model data when *Edit* is clicked in Fig. 12.1.

In Fig. 12.2 the user can edit the Model and also save it to an external library file for later use (*File*|*Export*). Model data can also be imported.

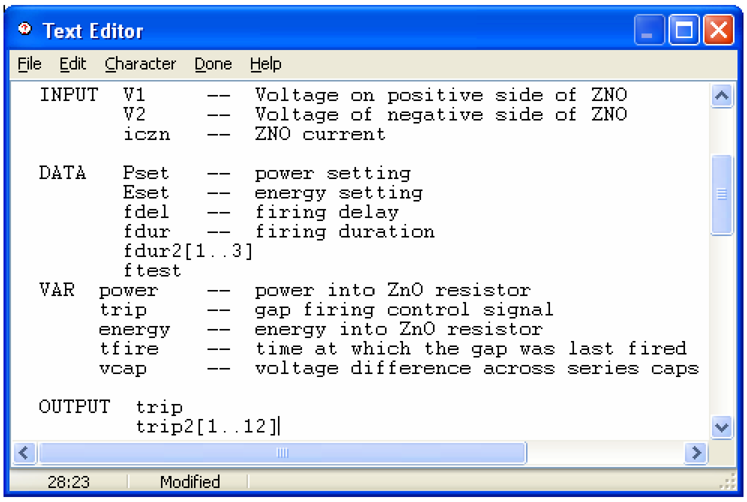
After editing the Model header the user clicks on *Done* in Fig. 12.1 ATPDraw examines the new header and replies with a message box given in Fig. 12.3 if some changes are discovered.



Model successfully identified
Input=3, Output=1, Data=7
Edit file?

Fig. 12.3. Model identification message. Click on *Yes* to edit the definitions for the model (icon, node positions etc.). Click on *No* to use the default vector graphic icon.

Changing the number inputs or outputs will affect the icon as is shown in Fig. 12.4. The new added 12-phase output trip2 will appear on the right side of the new vector graphic icon. Changing the variables and data will not affect the icon. Note that indexed data are supported in ATPDraw v5.



```
INPUT  V1     --  Voltage on positive side of ZNO
       V2     --  Voltage of negative side of ZNO
       iczn   --  ZNO current

DATA   Pset   --  power setting
       Eset   --  energy setting
       fdel   --  firing delay
       fdur   --  firing duration
       fdur2[1..3]
       ftest
VAR   power   --  power into ZnO resistor
      trip    --  gap firing control signal
      energy  --  energy into ZnO resistor
      tfire   --  time at which the gap was last fired
      vcap    --  voltage difference across series caps

OUTPUT  trip
        trip2[1..12]
```

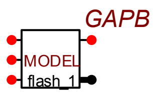
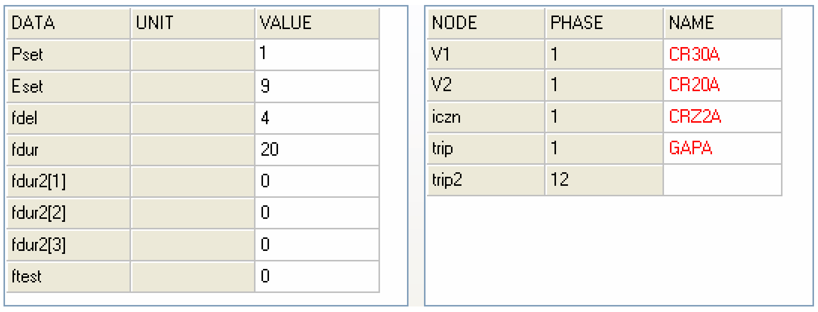


Fig. 12.4. Changing the Model's header.

The component dialog box of the Model shown in Fig. 12.1 will immediately be updated as shown in Fig. 12.5.



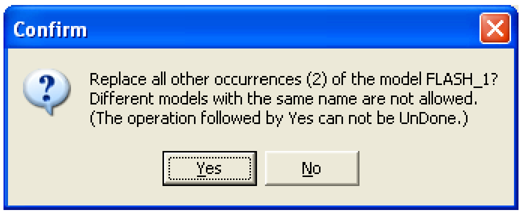
DATA	UNIT	VALUE
Pset		1
Eset		9
fdel		4
fdur		20
fdur2[1]		0
fdur2[2]		0
fdur2[3]		0
ftest		0

NODE	PHASE	NAME
V1	1	CR30A
V2	1	CR20A
iczn	1	CRZ2A
trip	1	GAPA
trip2	12	

Fig. 12.5. New component dialog (data and node parts) as a response to Fig. 12.4.

When clicking on *OK* in Fig. 12.1 all occurrences of models with the same name will be updated. If the user clicks on No in Fig. 12.6 he can go back into *Edit* to change the name of the Model and thus avoid updating the other Models.

When selecting *Library|Synchronize* the Models' icons are not updated. The same update feature also applies to User Specified objects and Line&Cables (LCC).



Replace all other occurrences (2) of the model FLASH_1?
Different models with the same name are not allowed.
(The operation followed by Yes can not be UnDone.)

Fig. 12.6. Copy data to other occurrences warning message.

13 TACS

Several new FORTRAN type components have been added to version 5, see appendix A. There is also a change in how positive/negative inputs to summations are visualized. The blue colouring of nodes is no longer used, and instead the icon connection is drawn as red for positive, blue for negative and white for disconnected inputs.

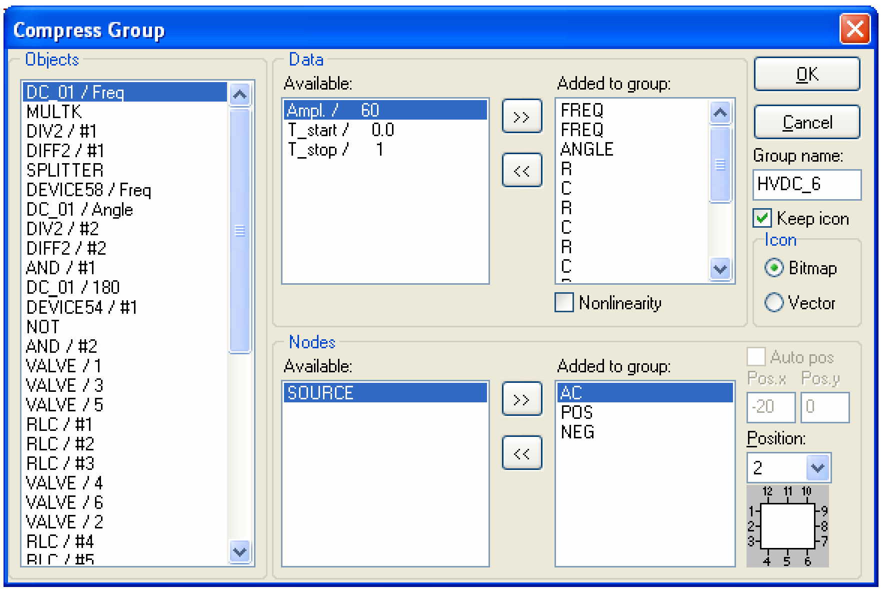
Relations are used to visualize information flow into Fortran statements and are drawn as blue connections, but have no influence on components connectivity. Relations are drawn in the same way as drawing a short circuit connection between nodes, except that you have to select the *TACS|Draw relation* option in the component selection menu to start the relation drawing. You can then draw multiple relations until you click the right mouse button or press *Esc* key.

14 Grouping

A new feature in version 5 is Regroup of existing groups. It is possible to select an existing group followed by *Edit|Compress* which will enable editing of the group (change the nodes and data sources). The Group icon can be on bitmap or vector graphics formats.

The user selects the external/global data and nodes as in earlier versions. Note that if some data share the same name they will be treated as one in the Component dialog box. Change the name (in the Added to group fields) by double clicking on them. Make sure that the nodes have unique positions (warnings are given). To locate a node outside the predefined 1-12 border positions specify position 0 and enter the co-ordinates directly. Vector graphic icons enable automatic node positions.

The *Keep icon* button is convenient when Recompressing a group with a user designed bitmap/vector icon.



Compress Group

Objects
DC_01 / Freq
MULTK
DIV2 / #1
DIFF2 / #1
SPLITTER
DEVICE58 / Freq
DC_01 / Angle
DIV2 / #2
DIFF2 / #2
AND / #1
DC_01 / 180
DEVICE54 / #1
NOT
AND / #2
VALVE / 1
VALVE / 3
VALVE / 5
RLC / #1
RLC / #2
RLC / #3
VALVE / 4
VALVE / 6
VALVE / 2
RLC / #4
RLC / #5

Data
Available:
Ampl. / 60
T_start / 0.0
T_stop / 1

Added to group:
FREQ
FREQ
ANGLE
R
C
R
C
R
C

[ ] Nonlinearity

Nodes
Available:
SOURCE

Added to group:
AC
POS
NEG

OK
Cancel
Group name: HVDC_6
[✓] Keep icon
Icon
[✓ Bitmap]
Vector

[ ] Auto pos
Pos.x -20 Pos.y 0
Position: 2

Fig. 14.1 The updated Compress dialog

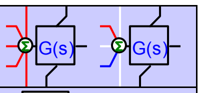
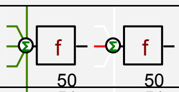
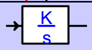
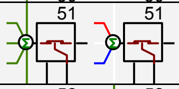
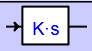
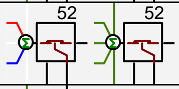
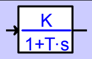
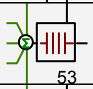
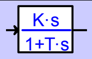
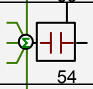
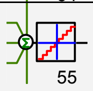
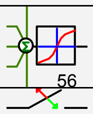
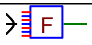
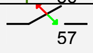
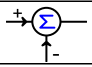
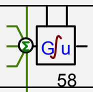
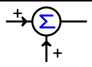
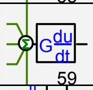
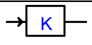
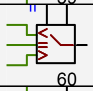
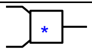
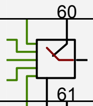
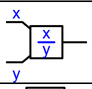
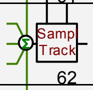
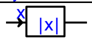
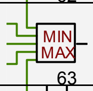
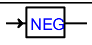
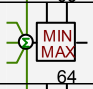
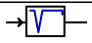
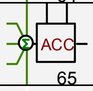
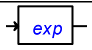
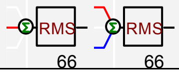
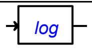
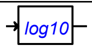
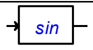
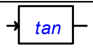
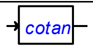
Appendix A

List of Selection menu components

Selection	Name	Icon	Selection	Name	Icon
Probes&3-phase			*Branch linear*		
Probe volt	PROBE_V	V	Resistor	RESISTOR	
Probe Branch	PROBE_B	i < +	Inductor	IND_RP	
Probe Current	PROBE_I	I	Capacitor	CAP_RS	
Probe TACS	PROBE_T	T	RLC	RLC	
Probe MODELS	PROBE_M	M	RLC 3-ph	RLC3	
Splitter (3 ph)	SPLITTER		RLC –Y 3-ph	RLCY3	Y Y
Collector	COLLECT		RLC-D 3-ph	RLCD3	
Transp1 ABC-BCA	TRANSP1		C: U(0)	CAP_U0	U(0) I +
Transp2 ABC-CAB	TRANSP2		L: I(0)	IND_I0	i(0) +
Transp3 ABC-CBA	TRANSP3		*Branch nonlinear*		
Transp4 ABC-ACB	TRANSP4		R(i) Type 9 9	NLINRES	R(i)
ABC Reference	ABC	ABC	R(i) Type 92	NLRES92	R(i)
DEF Reference	DEF	DEF	R(t) Type 97	NLINR_T	R(t)
Lines/Cables Lumped			R(t) Type 91	NLRES91	R(t)
RLC Pi-eq. 1.. 1..3 phase	LINEPI_1.. LINEPI_3		L(i) Type 98	NLININD	
RLC Pi-eq. 1.. 3 ph seq.	LINEPI3S		L(i) Type 93	NLIND93	
RLC Pi-eq. 1.. 3x1 ph. cable	PI_CAB3S		L(i) Type 96	NLIND96	
RL Coupled 51.. 1..3 phase	LINERL_1.. LINERL_3		L(i) Hevia 98->96	HEVIA98	H
RL Coupled 51.. 3 ph. seq.	LINESY_3		MOV Type 92	MOV	MOV
RL Coupled 51.. 6 phase	LINERL_6		MOV Type 3-ph	MOV_3	MOV
RL Coupled 51.. 6 ph. seq.	LINESY_6		R(TACS) Type 91	TACSRES	T
Lines/Cables Distributed			L(i) type 98, init	NLIN98_I	I λ
Transposed lines 1..3 phase	LINEZT_1.. LINEZT_3		L(i) type 96, init	NLIN96_I	I λ
Transposed lines 6 phase	LINEZT6N (all 6 transp.)	LINE Z-T	L(i) type 93, init	NLIN93_I	I λ
Transposed lines 6 phase mutual	LINEZT_6 (0-seq. coupl)	LINE Z-MT	Switches		
Transposed lines 9 phase	LINEZT_9 (all 9 transp.)	LINE Z-T	Switch time controlled	TSWITCH	
Untransp. lines 2,3 phase	LINEZU_2 LINEZU_3		Switch time 3 ph.	SWIT_3XT	
Lines/Cables LCC	PI, Bergeron, JMarti, Noda, Semlyen	Overhead, single core, Enclosing pipe	Switch voltage contr.	SWITCHVC	+ Vf -

Phase number set internally	LCC_1.. LCC_21	
Lines/Cables Read PCH file	Read pch file create lib file	Identifies nodes and length
Phase number set from PCH file	LCC_L1..9 LCC_N1..9	
Sources		
DC type 11	DC1PH	
Tamp type 12	RAMP	
Slope ramp type 13	SLOPE_RA	
AC type 14	AC1PH	
Surge type 15	SURGE	
Heidler type 15	HEIDLER	
Standler type 15	STANDLER	
Cigrè type 16	CIGRE	
TACS source	TACSSOUR	
Empirical type 1	SOUR_1	
AC 3-ph type 14	AC3PH	
AC ungrounded	AC1PHUG	
DC ungrounded	DC1PHUG	
Transformers		
Ideal 1 phase	TRAFO_I	
Ideal 3 phase	TRAFO_I3	
Saturable 1 ph.	TRAFO_S	
Saturable 3 ph.	SATTRAFO Y/D/A/Z	
# Sat. Y/Y 3-leg	TRAYYH_3	
BCTRAN	BCTRAN3	
Hybrid model	XFMR	

Diode (type 11)	DIODE	
Valve (type 11)	SW_VALVE	
Triac (type 12)	TRIAC	
TACS switch (type 13)	SW_TACS	
Measuring	SWMEAS	
Statistic switch	SW_STAT	
Systematic switch	SW_SYST	
Machines		
SM 59 No/8 contrl.	SM59_NC SM59_FC	
IM 56	IM56A	
UM 1 Synchronous	UM_1	
UM 3 Induction	UM_3	
UM 4 Induction	UM_4	
UM 6 Single phase	UM_6	
UM 8 DC	UM_8	
MODELS		
Default	MODELDEF	
Files (sup/mod)	In-memory editable	
Type 94 1..3 phase	TYPE94_1.. TYPE94_3	
TACS		
Coupling to circuit	EMTP_OUT	
Sources DC - 11	DC_01	
Sources AC - 14	AC_01	
Sources Pulse - 23	PULSE_03	
Sources Ramp - 24	RAMP_04	

TACS *Devices*			Transfer funct. General	TRANSF	
Freq sensor	DEVICE50		Transfer funct. Integral	INTEGRAL	
Relay switch	DEVICE51		Transfer funct. Derivative	DERIV	
Level switch	DEVICE52		Transfer funct. Low pass	LO_PASS	
Trans delay	DEVICE53		Transfer funct. High pass	HI_PASS	
Pulse delay	DEVICE54		TACS Initial condition	INIT_T	
Digitizer	DEVICE55		*TACS* *Fortran* *statements*		
User def nonlin	DEVICE56		General	FORTRAN1	
Multi switch	DEVICE57		Math x-y	DIFF2	
Cont integ	DEVICE58		Math x+y	SUM2	
Simple deriv	DEVICE59		Math x*K	MULTK	
Input IF	DEVICE60		Math x*y	MULT2	
Signal select	DEVICE61		Math x/y	DIV2	
Sample track	DEVICE62		Math |x|	ABS	
Inst min/max	DEVICE63		Math -x	NEG	
Min/max track	DEVICE64		Math sqrt(t)	SQRT	
Acc count	DEVICE65		Math exp(x)	EXP	
Rms meter	DEVICE66		Math log(x)	LOG	
TACS *Fortran* *statements*			Math log10(x)	LOG10	
Trigonom sin	SIN		Math rad(x)	RAD	
Trigonom cos	COS		Math deg(x)	DEG	
Trigonom tan	TAN		Math rnd(x)	RND	
Trigonom cotan	COTAN		*TACS* *Fortran* *statements*		

Trigonom asin	ASIN	asin	Logic NOT	NOT	
Trigonom acos	ACOS	acos	Logic AND	AND	
Trigonom atan	ATAN	atan	Logic OR	OR	
Trigonom sinh	SINH	sinh	Logic NAND	NAND	
Trigonom cosh	COSH	cosh	Logic NOR	NOR	
Trigonom tanh	TANH	tanh	Logic >	GT	x x>y y
Frequency Component			Logic >=	GE	x x≥y y
HFS Source	HFS_SOUR	HFS	Logic =?	EQ	x=y
Cigre load 1 ph, 3 ph	CIGRE_1 CIGRE_3	CIGRE LOAD CIGRE LOAD	*User Specified*		
Linear RLC	RLC_F	RLC	Library (default without parameter)	LIB	LIB
Kizilcay F-Dependent	KFD	F(s|z)	Files…	In-memory editable	HVDC LINE J.MARTI