ESCHER : Eindhoven SCHematic EditoR reference manual

Research Report

Eindhoven
University of Technology
Netherlands

Department of Electrical Engineering

ESCHER:
Eindhoven SCHematic EditoR reference manual

by
A. Lodder
M.T. van Stiphout
J.T.J. van Eijndhoven

EUT Report 86-E-157
ISBN 90-6144-157-9
ISSN 0167-9708

February 1986

Eindhoven University of Technology Research Reports

EINDHOVEN UNIVERSITY OF TECHNOLOGY

Department of Electrical Engineering

Eindhoven The Netherlands

ESCHER: Eindhoven SCHematic EditoR reference manual

by

A. Lodder

M.T. van Stiphout

J.T.J. van Eijndhoven

EUT Report 86-E-157

ISBN 90-6144-157-9

ISSN 0167-9708

Coden: TEUEDE

Eindhoven

February 1986

CIP-GEGEVENS KONIKLIJKE BIBLIOTHEEK, DEN HAAG

Lodder, A.

ESCHER: Eindhoven SCHematic EditoR reference manual / by A. Lodder, M.T. van Stiphout and J.T.J. van Eijndhoven. - Eindhoven: University of Technology. - (Eindhoven University of Technology research reports / Department of Electrical Engineering, ISSN 0167-9708; 86-E-157)
Met reg.
ISBN 90-6144-157-9
SISO 664.2 UDC 621.382:681.3.06 UGI 650
Trefw.: elektronische schakelingen; computer aided design.

ESCHER

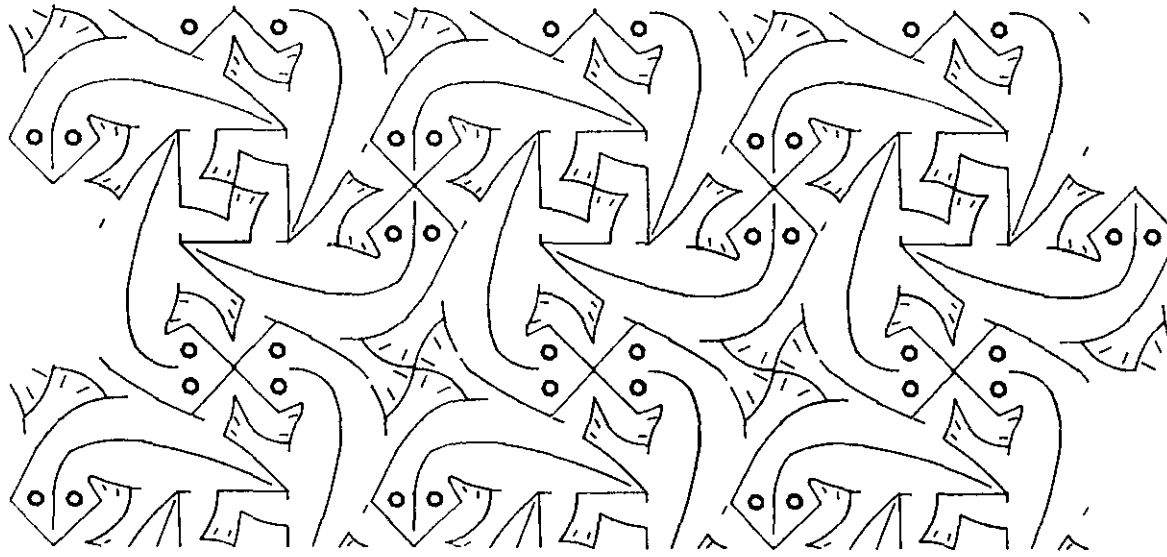
Eindhoven SCHematic EditoR

reference manual

ESCHER Reference manual

by

A. Lodder (ICD)
M.T. van Stiphout (Eindhoven University of Technology)
J.T.J. van Eijndhoven (Eindhoven University of Technology)



This tribute to M. C. Escher is designed with *ESCHER* and plotted by the *ESCHER* plotprogram.
It consists of several abutted instances of which some are rotated and/or mirrored.

ABSTRACT

This paper is a users guide for a schematics editor program, developed in the context of the ICD project. The purpose of the program is defining electrical circuits, by interactively drawing a hierarchically structured network. The output is in accordance with the ICD network standard, interfacing with several simulation programs and layout generation tools. In order to cope with the hierarchy, for each module (template) a contents is defined, as well as a representation. The contents section consists of instances of other modules, connections, and expressions for actual parameters of the instances. The representation consists of a symbol, terminals, and a formal parameter declaration. Furthermore the schematic editor is capable of handling busses in a very flexible way, handling mixed analog and digital circuits, and features a validaty check and a multilevel 'undo' command. The program is intended for a UNIX/C environment, and has 'lint' approval for portability.

The ICD project is a cooperation between the three Dutch universities, a Dutch software company (ICS), British Telecom, and German workstation manufacturer (PCS). The goal of the project is to develop an open system of CAD software tools, for the design of large scale integrated circuits. As project MR-09 it is supported by the Commission of the EEC, under the 3744/81 Microelectronics program. Under the NELSIS program, the Dutch partners were additionally supported by the Ministry of Economics Affairs of the Netherlands.

keywords: electrical networks, schematic entry, computer aided design.

For more information contact:

J.T.J. van Eijndhoven
Dept. of Electrical Engineering
Eindhoven University of Technology
postbox 513
5600 MB Eindhoven
The Netherlands

or:

P. van Es
ICD
postbox 3132
7500 DC Enschede
The Netherlands

Lodder, A. and M.T. van Stiphout, J.T.J. van Eijndhoven
ESCHER: Eindhoven SCHematic EditoR reference manual.
Department of Electrical Engineering, Eindhoven University of Technology (The Netherlands), 1986.
EUT Report 86-E-157

CONTENTS

NAME

escher - interactive graphic editor for network database entry.

SYNOPSIS

escher [-a<number>][-u<number>][-w][-l][-p<string>][-m]
[-r<string>][-R<string>][-e][-t<string>][-s<number>]
[-n][-d<number>][-q<number>]

DESCRIPTION

ESCHER recognises the following options:

-a Autosave. Every <number> minutes an autosave is performed. Default no autosave.

-u Depth of the undo stack. Number is the <number> of commands that can be undone. Default is 10.

-w Allows wires to run along the sides of instances. Default no allowance.

-l Line edit menu invisible. With this option the line edit menu will not be shown during line edit. Default visible.

-p Plot options. The string will be passed to the plot program when called.

-m Mirror_x mirror_y interchange. Default the mirror_x command will be a mirror against a vertical line and mirror_y against a horizontal line. With this options set the meaning will be interchanged.

-r <string> will be used by the library handler for soft replacement. See for the syntax and semantics the ESCHER reference manual.

-R <string> will be used by the library handler for hard replacement. See for the syntax and semantics the ESCHER reference manual.

-e Default the command 'change name' will give the name of the instance or terminal as default for the line edit. This option will result in an empty default.

-t string will be used as the default basename for terminals used by the add and copy commands. Default is "term".

-s number is the maze on the grid to which all input is rounded. Only valid in the circuit area. Default maze is 4. May be set to 1, 2 or 4 only.

-n names of instances and terminals of instances will not be drawn if they become too big compared to the size of the instance. With this option the drawing of these names is forced.

-d Error messages are after 10 seconds by default replaced by the name of the current template. The number following this option will become the maximum display time of the errors. Must be positive or 0.

-q Number represents the amount of seconds the error messages will be visible at least. The default is 2 seconds. Number must be positive or 0.

The options are read with the function getopt(3). See the manual page for syntax of the options. An unknown option or an option requiring an argument but has none or an option with an incorrect syntax for number or resulting in a negative number will give an error message on stderr and cause an immediate termination of the program.

ESCHER uses the following environment variables:

PLOT_SHELL
the command that will be executed when using the plot command. This string is concatenated with the string following the -p option and with the model pathname of the current model. One of the plot command flags "-c" or "-r" is also inserted when active in the circuit or representation area respectively.

USER_LIB
path name of the user library. Default is the current working directory. See the ESCHER reference manual for details.

LIB_PATH
library path name of the user library. See the ESCHER reference manual for details.

GRAPHIC_DEVICE
path name of the graphics device.

FILES
See ESCHER reference manual.

CONTRIBUTED BY
Arie Lodder and Mart van Stiphout.

STATUS
release 1.0

NAME

escherp - escher plot program.

SYNOPSIS

escherp [-r][-c][-v<real>][-a<number>][-n<name>][-b][-f] <template pathname>

DESCRIPTION

escherp recognises the following options:

-r plot the representation of the template.

-c plot the circuit of the template. If both the *-c* and the *-r* are omitted then *-c* is default.

-v <real> is the plot pen velocity ranging from 0.38 to 38.1. Highest value is default.

-a paper format. <number> = 3 (A3) or 4 (A4). Last is default.

-n<name>
draw name above the plot. If <name> = "0" the template name is used.

-b draw name under the plot. Works only in combination with the *-n* option.

-k does not draw a frame around the drawing. Default is a frame.

-f terminal names and instance names are forced to be drawn independent of instance size.

The options are read with the function getopt(3). See the manual page for syntax of the options. An unknown option or an option requiring an argument but has none or an option with an incorrect syntax for number or resulting in a negative number will give an error message on stderr and cause an immediate stop of the program.

escherp can be used as stand alone program but may be called from ESCHER with the plot command. See the ESCHER reference manual.

escherp writes errormessage to stdout to let other programs read from a pipe. If no errors are found it gives: Escherp: no errors found.

FILES

escherp spools output to the file with name 'plotfile' in the current directory.

How this manual is organised.

This manual contains 2 parts:

- Description of the features. Not everything can be explained at once so some browsing may be necessary.

- Description of commands. May be used as reference manual.

The reader of this manual is expected to be familiar with the concepts of network description like: templates or models, instances or subsystems, terminals, wires, busses, formal and actual parameters, abutment and nets.

1. Features description of ESCHER.

1.1 Introduction.

The Eindhoven SCHematic EditoR is an interactive graphical editor suitable for entering or changing information in the ICD network database. Other programs like simulators and automatic placers may use this information as an input.

1.2 Network database.

The network database format is described in the ICD paper:

- Proposal a for Network Description Format in the ICD-system. *

by G.L.J.M. Janssen (Eindhoven University of Technology) and A.C. de Graaf (Delft University of Technology). *ESCHER* uses the ICD implementation of interface with this network database described in the manual pages of lNDI. An introduction may be found in INTRO(lNDI). In an appendix a resume of the fields in the network database used by *ESCHER* is given.

1.3 Templates.

A basic unit of information is the template consisting of

- Circuit, being the inner side of the template build from:
  - Instances with:
    - Subsystem terminals.
      Width, type, kind and attribute is inherited from its representation.
    - Actual parameters.
      Number, names and default values extracted from the formal parameter of its template. Each parameters has a value which could be an expression.
    - Position.
    - Size. Dependent on the size of the surrounding box of its representation.
    - Orientation. Combination of rotation and mirroring.
    - Unique instance name.
  - System terminals:
    - Width.

---

* Chapter 1.3 in: The Integrated Circuit Design Book. Ed. by P. Dewilde. Delft University Press, 1986.

- Type. Signal or electrical. This information may be used by programs reading the network database. See for the semantics the corresponding program. If is not intented to be used set it to signal.

- Kind.
  Inherited from its representation equivalent. Is input, output, input/output or unused. This information may be used by programs reading the network database. See for the semantics the corresponding program. If is not intented to be used set it to input/output.

- Attribute.
  Inherited from its representation equivalent. Could be pull_up, pull_down, tristate or other. This information may be used by programs reading the network database. See for the semantics the corresponding program. If is not intented to be used set it to other.

- Unique position.

- Unique name.

- Wires with:

  - Type.

  - Lower bound.

  - Upper bound.

  - Start and end position.

  A wire must be horizontal or vertical.

- Representation, being the outside of the template:

  - Surrounding box.
    Defines the size of an instance of the template.

  - Terminals with:

    - Type.

    - Width.

    - Kind.

- Attribute.

- Unique name.

- Unique position on the surrounding box.

- Symbol, containing:

  - Lines.
    defined by start and endpoint.

  - Circles. having a centre and a radius.

  - Arcs. Defined by the user by 3 points, internal having a centre, radius, start and end angle.

- Formal parameters defined by:

  - Name.

  - Kind.
    Real, integer or boolean. This information may be used by programs reading the network database. See for the semantics the corresponding program.

  - Range. This information may be used by programs reading the network database. See for the semantics the corresponding program.

  - Default value. This information may be used by programs reading the network database. See for the semantics the corresponding program.

1.4 Hierarchy.

Although the editor is meant to deal with a hierarchical description *of a network the editor will not work hierarchical.* Templates are changed one on a time. By selecting a certain template as root a complete network is defined. This is done by programs reading the network database, not by *ESCHER*.

1.5 Change template.

To end an edit of a template and start the edit of another one a change template command is provided. Change of the current template involves an automatic save of the graphical information of the template on disk and will establish, if it is consistent, a complete network description of the template in the network database. The dump of the graphical information will be used when re-editing the model.

Fast change of the current template is provided by selecting an instance of which the template will become object of the editor. This is called level down.

With every change the name of the old template is push on a template stack. It is possible to pop names of this stack, known as level up. With these features the user is capable to design top down and bottom up as well. Remark that the result of the change template command is pushed on the template stack thus the level up command is not necessarily the reverse of the last issued level down command.

ESCHER's internal memory contains all the graphical information of all the templates used in the session. The template that is edited is called the current template. If there are no error messages present in the errorline this line will show:

- Current template: <template pathname>

Some commands like abort and an error in the change template command will result in an absence of a current template. This is properly indicated in the error line when issuing another command than change template or level up by:

- ERROR: No current template!!

1.6 **Libraries.**

Templates are determined by their name. As indicated in the description of the network database implementation this name is equivalent with a directory on disk. The UNIX pathname of this directory is called the template pathname. Multiple template directories collected in a single directory is called a library. *ESCHER* has mechanisms to supply a flexible and consistent use of these pathnames and libraries.

First of all when entering a model name of a template which will become editor object the directories in the user supplied library path environment variable LIB_PATH are searched for this name. The format of this variable is:

- <library pathname>[:<library pathname>]*

If it is not found then a new template is created in the directory given in the environment variable USER_LIB which must be an UNIX pathname.

If the add instance command is used the name of a template has to be given which will be the origin of an instance. In this case the same library path is searched.

When starting to edit a template it is possible to replace all instances of a model in the circuit by instances of another model. This is done by the *ESCHER* program options

- *-R* <library pathname>:<template name>
  meaning hard replacement of the template with the template of the same name from the library.

- *-r* <library pathname>:<template name>
  meaning soft replacement of the template with the template of the same name of the library.

Hard replacement will cause a deletion of the old instance if the new one doesn't fit. Every time it occurs it gives the error message:

- WARNING: Instance(s) updated or deleted.

Soft replacement will cause to flag an error and a reject of the edit of the template.

- ERROR: Instance <name> does not fit.

Hard replacement is also performed if the template has instances of which the instance time is older than the last modification time of the representation of the instance. If the new one doesn't fit the old one will be removed. See the chapter about instance consistency.

Fitting is rather intelligent: All changes including adding and deleting terminals or formal parameters, moving terminals or changing their names, enlarging the size of the surrounding box will cause a replacement if the new instance will fit in the circuit. A change of a terminal connected to a wire may cause a deletion of the wire. Actual parameters will be deleted if a formal was deleted or a new one will be added if a new formal was formed. Fitting is hard in case of abutment.

In the same way it is possible to replace all the models from a library by the models of another library with the same name. The template name in the '-r/-R' options is simply replaced by another library path name.

Example of how to move models around on your disk: Suppose that template T has instance of template called U. Both are in directory xxx so the pathnames of the model directories are xxx/T and xxx/U. To place both models in the directory yyy simply copy them with a save command of *ESCHER* or with a UNIX command to yyy. Insert the yyy-pathname before the xxx-pathname in the LIB_PATH environment variable. Then next time you start *ESCHER* give the option: -ryyy:U. Start an edit of the T template and save it immediately under the pathname it is found by *ESCHER*: yyy/T. Every reference to U in T will now have also the yyy library name in it.

If LIB_PATH is empty the error line will show when starting the program:

- WARNING: Library path empty

And if the syntax is not correct:

- ERROR: Library path incorrect

If the USER_LIB is empty the current directory acts as user library.

Remark that the user library is not automaticly searched at the add instance command. It is recommended to include the user library in the library path at a user selected place.

1.7 **Save.**

The user may save the graphical information of the template on disk. This can be done with the same pathname and model name it is called or with a new name and pathname.

If the network information is consistent also the network database is updated with the new information. If it is not the current network files of the model are cleared thus providing always consistent and actual network information.

With the *-a* option the user may set an autosave at certain time intervals. The save is done in the directory with the pathname of the current template. The errorline will show:

- WARNING: Performing autosave

1.8 **Editing.**

Editing is adding or deleting instances, actual parameters of these instances, wires and terminals of the circuit and symbol elements, terminals, surrounding box and formal parameters of the representation. Not every edit possibility goes with every kind of object, in detail:

	add	delete	change	move	copy	rotate	mirror
Instances	X	X	X	X	X	X	X
Wires	X	X					
Terminals	X	X	X	X	X		
Actual parameters			X				
Formal parameters	X	X	X				
Surrounding box	X		X				
Symbol elements	X	X		X	X		

Add and copy of instances and terminals will automaticly generate a unique name for them.

Change of terminals and instances will only involve the name of it.

Copy of instance is also a copy of its actual parameters.

A manipulation command is one of the commands add, delete, copy or move. An item is one of instance, wire, terminal, line, circle, or arc. The user may select one of the items as current item (see status menu description) and a manipulation command will work with the item that is current at command selection time.

Possible items while editing the circuit are instance, wire or terminal. In the representation are terminal, line, circle or arc the selectable items.

Trying to issue a copy or move wire will show:

- Not implemented use 'add bus'

1.9 Command parsing.

A command consists of the selection of the command in a menu and giving 1, 2 or 3 coordinates as parameters. Of course this number is command dependent. After executing the command most commands will stay active so only the coordinates have to be entered. Manipulation commands work on the current item.

Items stay active until an other is selected. Commands needing no arguments like zoom, window world, selecting an item and toggle grid may be combined with the normal command input without interrupting it.

Selecting a command instead of a coordinate or changing items while putting in the coordinates of a manipulation command will result in a reject of the old input. If the input was a new command this will become active.

Active command and current item is properly indicated in red in the menu and status area.

The input of the first command is echoed with a crosshair, of the second with a rubber line to the first and of the third with the same echo to the second. The echo for the second coordinate of the window box and edit box command is a rubber box.

1.10 Menu's.

The commands are gathered in the following menus:

- Main menu. In the menu area. For the commands:
  - Exit.
  - Verify.
  - Save.
  - Change template.
  - View.
  - Level down. Not in representation.
  - Level up.
  - Edit name.
  - Edit parameters.
  - Abort.
  - Plot.
  - Toggle area.
  - Shell escape.
  - Edit menu.
- Edit menu. In the menu area. For the commands:
  - Edit box. Not in the circuit

- Add current item.
- Delete current item.
- Move current item.
- Copy current item.
- Split.
- Rotate instance. Not in representation.
- Mirror x instance. Not in representation.
- Mirror y instance. Not in representation.
- Info.
- Window box.
- Toggle area.
- Undo.
- Main menu.

- Status menu. In the status area. For the commands:
  - Toggle grid.
  - Zoom in.
  - Zoom out.
  - Window world.
  - Change item to instance.
  - Change item to wire.
  - Change item to terminal.
  - Change item to line.
  - Change item to circle.
  - Change item to arc.
  - Change type.

  - Change kind.
  - Change attribute.
  - Change offset.
  - Change bus width.

- Context menu. In the context area. For the window center command.

1.11 Connections of terminals and wires.

A wire is connected to a terminal if it ends or start on a terminal. Wires are connected if they share an end or start point.

Terminals may be multiple. It is called that they have a width. The width of a terminal is determined by the current bus width, seen in the status menu, when adding a terminal to the representation or circuit. A single terminal of a multiple one is denoted by a number, called its range number, running from 0 to width - 1.

Wires may also be multiple having a lower bound, being the value of the current offset when adding wires, and an upper bound, found by the value of the current offset add to the current bus width with 1 subtracted. A single wire is referenced by the wire range number running from lower bound to upper bound.

The connection of a multiple terminal with a wire is defined by the offset of this connection: a number, positive or negative, which must be added to the range number of the terminal to find the range number of the connected wire.

Making connections between terminals and wires can be done with the add wire and the add, move or copy terminal commands. There are 3 possibilities:

- The wire and the terminal are of the same width: the offset is derived by *ESCHER* and is equal to the lower bound of the wire.
- The wire is smaller than the terminal width: The user is prompted for an offset. The direction of the wire, seen from the terminal, is indicated in the prompt. An errormessage is given if the offset is illegal, that is that some wires of the bus are not connected:
  - ERROR: High range of wire <direction> not connected
  - ERROR: Offset greater than low range of wire <direction>

- Wire width is greater than terminal width giving:

    - ERROR: <direction> wire width greater than terminal width.

Adding, moving or copying instances may also result in a terminal-wire connection. The offset is always made equal to the lower bound of the wire. This is called automatic offset generation.

Of multiple wires sharing an start or end point only the wires with the same range number are connected.

Crossed wires are not connected.

When adding a wire ending or starting from the middle of an existing one the latter is divided in 2 wires.

When deleting a wire leaving 2 wires connected in the same direction these 2 are merged to 1.

1.12 **Nets.**

A net is the collection terminals, connected by wires or abutment. A net has a net indicator equal to the terminal indicator of one, not predictable which, terminals of the net. The format of the terminal indicator is:

- <origin name> <terminal name> <range number>

The origin name is the name of the instance if it is a subsystem terminal else it is empty. The range number is omitted if it is 0.

Terminal and wire indicators are important for the verify information.

Remark that a wire, part of a multiple wire, is referenced by a terminal indicator and thus may show, by non-zero offset, a different range number in its indicator than the wire range number.

1.13 **Net input and output behaviour.**

ESCHER extracts from the kinds and attributes of the terminals of a net its output and input behaviour. The result is included in the network database. The procedure to determine the behaviour is given below:

The net is examined sequentially on its subsystem terminals. If the are input or inout the net input behaviour is changed according the following matrix which denotes the function:

- New behaviour = f(old behaviour, attribute of next terminal)

If the behaviour was unknown it is inherited from the first subsystem terminal encountered.

old behaviour:	pull_up	pull_down	tristate	other
next terminal:	-	-	-	-
pull_up	-	other	pull_up	-
pull_down	other	-	pull_down	-
tristate	-	-	-	-
other	other	other	other	-

In this table '-' means no change.

The outputs are processed in the same way according to the matrix:

old behaviour:	pull_up	pull_down	tristate	other
next terminal:				
pull_up	-	other(W)	pull_up(W)	(W)
pull_down	other(W)	-	pull_down(W)	(W)
tristate	(W)	(W)	-	-
other	other(W)	other(W)	other(W)	(W)

(W) means no change and verify information will show a warning.

If both the input and output behaviour can be determined and output is pull_up and the input is not pulldown an error occurs in the verify info. Also if the output is pull_down and the input is not pull_up.

After this the system terminals are checked. Errors are given if:

- Terminal is unused although connected.
- Terminal is input and attributes conflict with input behaviour if known.
- Terminal is output and attributes conflict with output behaviour if known.
- Terminal is inout and attributes conflict with in- or output behaviour if known.
- Terminal is input and output behaviour is known.

- Terminal is output and output behaviour is not known.

Further errors if:

- net consists (not containing a single terminal only) and input behaviour is unknown and there is no output system terminal belonging to the net.

- net consists (not containing a single terminal only) and output behaviour is unknown and there is no input system terminal belonging to the net.

A warning if:

- if a sub_terminal of kind input is unconnected and it has a tristate or other attribute.

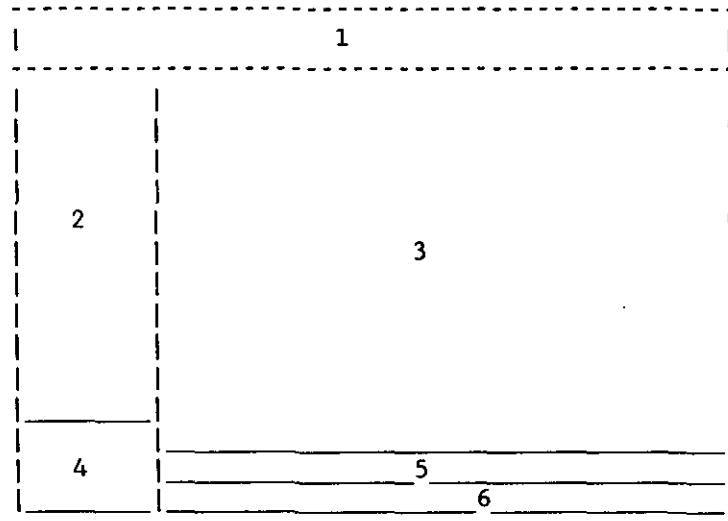
Above mentioned errors and warnings are only visible when using the verify command and will not be given while making connections.

An error in the net will result in a clear of the files of the network database when trying to save the model on disk. See the save command.

Remark: If all the terminals of the circuit are input/output and have an attribute of other none of these warnings or errors will occur.

1.14 **Screen.**

The screen is divided in several areas as shown below.



With:

- 1: Status area for the status menu.

- 2: Menu area for main and edit menu.

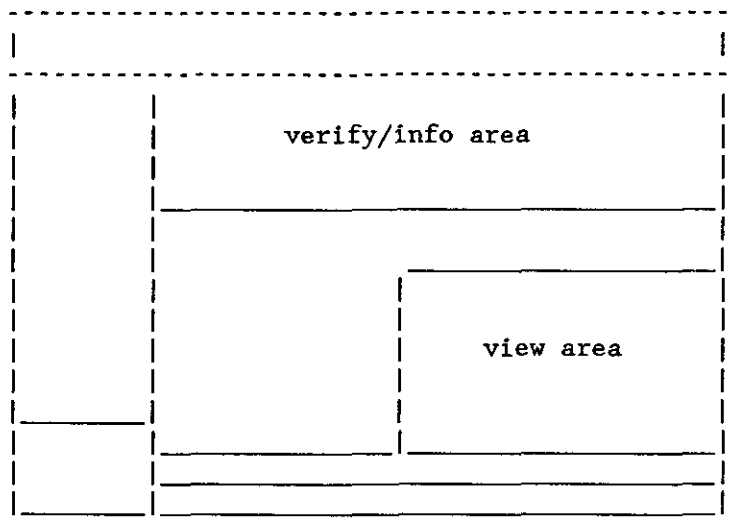
- 3: Work area for actual edit of circuit and representation. Depending on this function this area is called the circuit or representation area. Use the toggle area command to move from one area to the other.

- 4: Context area. For the window centre command.

- 5: Error line.

- 6: Prompt edit line.

The screen may contain several areas in the work area for special functions:



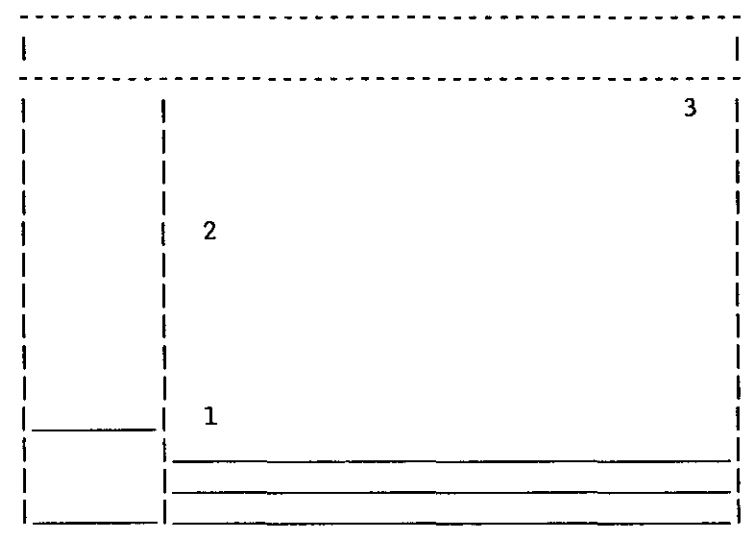
The verify/info area contains 11 lines of text. If there is more information as a result of the info or verify commands the last line will show: --More-- else it will be empty. See the description of the commands of how to show the rest of the information.

All of these areas show text and drawings in white on a black background.

The info/verify area is also used for textedit of formal and actual parameters.

The existence of information in the verify/info or view area does not prohibit the edit of the current template. Editing is done as if these areas were not present. See the description of the proper commands to clear these area's.

Different kinds of textedit use different places of the line edit menu:



- 1: While text editing for a prompt edit.
- 2: While editing formal parameters.
- 3: While editing actual parameters.

The line edit menu has white text on a black background. It will disappear at the end of the edit. The *-l* option will cause not to show this menu at all.

1.15 **Abutment.**

Contiguous instances are only permitted if they can be abutted, that is: share terminals. If so we call equivalent terminals abutted terminals. In case of possible abutment proper matching of terminals is checked:

- The number of terminals at the contiguous sides has to be equivalent.
- The type needs to be the same.
- Terminals need to have equal width.
- Terminals may not have a connection.
- Terminals may not be both output terminals.
- Terminals may not be both input terminals.

- Unused terminals may only be abutted to unused ones.

When abutted the commands move instance, copy instance, rotate instance, mirror_x instance and mirror_y instance will operate on the set of abutted instances as if it were a unit.

The size of a collection of abutted instances is equal to the smallest box that can contain all the instances. This size is taken in account to check whether moved abutted instances will conflict with existing terminals or wires.

Also if new instances are added to an abutted set overlap or contiguousness is checked with this surrounding box. Therefor, in a few situations, *ESCHER* might, although it could be correct, refuse to make the abutment, thus making the order of abutment critical.

Abutment can be undone with the undo, delete instance and the split command.

1.16 Formal parameters.

A template might have formal parameters having the features:

- Kind. Giving the type of the formal, could be real, integer or boolean,

- Range. Format: <lowerbound>, <upperbound>. The value of the actual parameter of an instance of this template must be between upper and lower bound.

- Default. Being the value of the corresponding actual when creating an instance. The default must be in the given range.

The default and the range must be integer with an integer formal. Booleans do not have ranges nor default.

The default and or the range do not have to be specified. If the string for kind is empty a real is assumed.

Integer and reals have the same format as real and integers as part of an actual parameter expression. Also the in the appendix mentioned multipliers may be used.

The user may give these parameters an actual value if the template is called, each instance his own actual value.

1.17 Actual parameters.

The actual parameter value may be a complex expression consisting of real, integer and boolean values, operators, function calls and variables, the last being the names of formal parameters of the template. See the Appendix for a complete BNF notation of the expressions.

The edit of the parameters is done with text edit in the info/verify area.

ESCHER will assign the default value of the formal if a new instance is made.

ESCHER will not evaluate the expression nor check whether this value lies within the range of the formal parameter.

1.18 Verify.

ESCHER has the possibilities to determine the consistency of the network information. When using this command the correctness is determined and showed to the user in the verify/info area. It may contain errors and warnings. If the verify information of the template contains errors the network database is not updated with a save command but give:

- WARNING: Error in network; network files cleared.

Verify information consists of:

- Checking networks:
  - ERROR: net <terminal indicator> short cut
  - ERROR: net <terminal indicator> dangling end
  - WARNING: unconnected wires
    Wires detected without terminal connection
- Checking input/output behaviour of networks:
  - ERROR net <terminal indicator> <net input behaviour> with <net output behaviour>
    The input behaviour conflicts with its output behaviour.
  - WARNING net <terminal indicator> terminal <terminal indicator> <attribute> with <attribute> net
  - ERROR: net <terminal indicator> terminal <terminal indicator> unused terminal
    The terminal is unused although connected.

- ERROR: net <terminal indicator> terminal <terminal indicator> attribute conflicts with net
  The attribute of the terminal is not compatible with the behaviour of the net. Input terminals are checked against input behaviour and output terminals are checked against output behaviour. Input/output terminals get both checks.

- ERROR: net <terminal indicator> terminal <terminal indicator> input with output net
  Terminal is input and output behaviour of the net is known.

- ERROR: net <terminal indicator> terminal <terminal indicator> output with undetermined net
  Terminal is output and output behaviour is not known.

- When checking terminals:

  - ERROR: terminal <terminal indicator> type incompatible with representation.
    The type of the indicated terminal is not equal to the type of its name equivalent in the representation.

  - ERROR: terminal <terminal indicator> width incompatible with representation.
    The width of the indicated terminal is not equal to the width of its name equivalent in the representation.

- When checking formals:

  - ERROR: formal parameter <parameter name> invalid kind
    Kind given is nor 'R', 'r' or empty meaning real or not 'I' or 'i' indicating integer or 'B' or 'b' for boolean.

  - ERROR: formal parameter <parameter name> low range greater than high range

  - ERROR: formal parameter <parameter name> range: real expected

  - ERROR: formal parameter <parameter name> range: integer expected

  - ERROR: formal parameter <parameter name> range: comma expected

  - ERROR: formal parameter <parameter name> range: unexpected character

  - ERROR: formal parameter <parameter name> default: bool expected

- ERROR: formal parameter <parameter name> default: real expected
- ERROR: formal parameter <parameter name> default: integer expected
- ERROR: formal parameter <parameter name> default not in range
- ERROR: formal parameter <parameter name> default: unexpected character

- Checking actuals:
  - ERROR <parameter indicator> expression expected
    actual parameter does not have a value
  - ERROR <parameter indicator> identifier expected
  - ERROR <parameter indicator> ')' expected
  - ERROR <parameter indicator> '(' expected
  - ERROR <parameter indicator> ',' expected
  - ERROR <parameter indicator> unexpected character
  - ERROR <parameter indicator> unknown identifier
    the given identifier can not be recognised as the name of a formal parameter or as the name of a terminal of the current template.

- Checking areas:
  - WARNING: Representation is empty
  - WARNING: Circuit is empty

The messages on checks of parameters are accompanied by a position. This is the number of the character in the expression. When using the line edit menu this number is equal to the cursor position given in this menu. This may not be true when an actual parameter expression is divided over 2 or more lines.

Disk access and reading from disk again from graphical information may change the net indicator: it could be the terminal indicator of another terminal from the same net.

The dangling end error message may result in some, not correct, error messages of short cut in other nets. (This net might be the same net but it is named as a new net because the occurrence of a dangling end

will cause *ESCHER* to stop with the check of that net) Remark that the correct message of a dangling end will follow the messages on short cuts.

1.19 **View.**

The circuit of a template will show only the representation of the template of an instance in the circuit area. With the view command it is possible to give a fast overview of its inside. No editing of this circuit will be allowed.

While editing the representation of a template the view command will give a view of the circuit of the template.

1.20 **Undo.**

The result of every successful manipulation, edit, window management and toggle area command is kept on the undo stack. By giving the undo command the undo information on top of the stack will be undone.

The undo stack is cleared at each interaction, also implicit, with the disk, such as all change template commands, autosave and abort,

The undo stack has a user selectable, with the -u option, depth. The default depth is 10 commands. Recommended is either a small integer or a very large one, 100000, effectively giving an unlimited depth.

1.21 **Info.**

With the info command information of instances is obtained. This information consists of its name, the name and attributes of terminals and the values of the actual parameters.

While editing the representation it will show the formal parameters and the names and attributes of the terminals.

If the template has no terminals, actuals or formals this is properly indicated.

1.22 **Grid.**

To fasten input and improve the quality of drawing all input is converted to points on a grid. Size of the grid is called grid distance. In the circuit area editing is default done with a grid of 4 * 4 grid distances. This is called a maze of 4. To lay wires between existing ones the user may set the maze to 2 or 1 with the *-s* option so defining a grid of 2 * 2 or 1 * 1 respectively.

In the representation there are 2 grids: a fine dotted and a 10 times as large solid one. With the item on line, circle or arc the user is working on the fine grid, for the surrounding box or the position of

terminals only the solid grid is available.

When adding an instance of a template the solid grid is mapped to the 4 * 4 grid of the circuit thus ensuring that terminals will always be on this grid. If necessary the commands add instance, move instance and copy instance will imply a slight correction of instance positions to give above mentioned results.

The user may select a visible or invisible grid using the toggle grid command. When zooming out the grid may disappear when becoming to fine. In the representation the fine grid is disappearing first.

1.23 **Window handler.**

ESCHER provides fast and flexible window management functions including zoom in and zoom out with a factor of 0.7 and a function, window world command, which will adjust the window in such a way that the total circuit or representation will be visible.

The screen contains a window context area in which all instances are drawn as solid boxes and the current window as a solid box in a different colour. Selecting a coordinate in this context area will cause a window centre function.

All these input will not interrupt the currently active edit command. The locator echo position is adjusted as well and clipped against the new window if necessary.

Another window function is window box which needs 2 coordinates as the opposite corners of a box which will be the new window.

Using this feature is especially useful when adding long wires. First zoom in at the start of the desired wire. Then give an add command with wire as current item and give the first coordinate. Then give a window centre command in the context area and give then the second input.

1.24 **Split.**

A split function is provided to split the circuit or representation by moving all the instances, terminals, wires and symbol elements over a certain distance to the left, right, up or down so giving room for more of these items. Wires across this split line will be automaticly lengthened.

1.25 **Line edit.**

Alphanumeric input is provided by means of a line editor. A default text may be edited using control characters to perform special functions like:

- Control V - toggle insert/overwrite.
- Control G - delete character.
- Control H - delete previous character.
- Control Y - delete line.
- Control K - delete rest of line.
- Control U - undo last input.
- Control A - move cursor to the left.
- Control D - move cursor to the right.

Any printable character is inserted or placed depending on the insert/overwrite mode.

A textedit is accompanied by a line edit menu in which a BEEP will show when trying to move the cursor across the line bounderies or when entering an unrecognised control character. With the *-l* option the user may make the menu invisible.

A return is interpreted as stop edit. Remark that the place of the cursor is not significant and so, because most of the alpha-numeric input is accompanied with a default text, a simple return is enough to enter the default value.

Actual and formal parameter edit is done by using textedit in the info/verify area. Promptedit is done in the text area. See screen description for the meaning of area's.

1.26 Instance consistency.

It is not permitted to work with inconsistent instances, that are instances of which the representation of the template is altered after the instance is made. *ESCHER* keeps record of the instance times and representation modification times and compares them when a template is entered for an edit. In case instance time is oldest the old instance is replaced by a new instance of the same template. Replacement is hard so it may be deleted if the new one does not fit. Entering a template with inconsistent instance will give an errorline of:

- WARNING: Instance(s) updated or deleted.

Symbol elements may be edited without changing the modification time.

There is no undo of the modification time if it concerns terminals. So changing a terminal and undoing the change will make the instance inconsistent, but this will only result in an automatic update, no

deletion. Undo of edit box command restores the modification time.

1.27 **Errors.**

The error line on the screen is dedicated to show error messages. An error will stay on screen for at least 2 seconds. If within this time a new error occurs the editor will pause for a short time. This may be changed with the *-q* option.

Errors will not be shown longer than 10 seconds. This time is also controlled by an option: *-d*. After this period it is replaced by the pathname of the current template.

There are 4 kinds of messages:

- ERROR. This is followed by an error description. The command is not executed.
- WARNING. The command is executed but input or result is slightly modified.
- SYSTEM ERROR. The system has returned an error. The system error message is also displayed. See errno(1). Common system errors are:
  - <pathname> No such file or directory.
    Some file or part of a pathname can not be found.
  - <pathname> Permission denied.
    The user has no permission to read or write the file.
- FATAL. A signal is caught. The kind of signal is mentioned. See signal(2).

A error message without keyword is a simple message not accompanied by an *ESCHER* action.

1.28 **Drawings.**

The drawing of the circuit is according to the following rules:

- Of an instance the symbol elements are drawn according to orientation of the instance. The mirror and rotate commands change the orientation of the instance. If no symbol elements are known the surrounding box is drawn. The instance name is shown in the middle of the instance. If it is too long it will not be drawn except if the user has given the *-n* option. Drawing colour is yellow.
- Subsystem terminals are drawn in a type dependent colour: blue for signal and green for electrical. The name and the width,

unless equal to 0 and surrounded by square brackets '[]', are placed in the terminals neighbourhood and always in the instance. They are not shown if the text becomes to big compared to the instance. This may be overruled with the *-n* option. Names are always clipped against the surrounding box. An offset is shown if the terminal is connected and the offset differs from 0.

The shape of the terminals is equal to that of terminals of the representation.

- System terminals are drawn in a type dependent colour. Name and width, if not 0, is mentioned. An offset is shown if the terminal is connected and the offset differs from 0. These terminals are square shaped.

- Wires are coloured in a type dependent way. Upper and lower bound are mentioned somewhere in the middle of the wire, format: <lower>:<higher> except if both bounds are 0. If the lower bound is equal to the higher bound the bound is only mentioned once without ':'.

- To show difference between 2 crossed wires and 4 connected ones in the last case a hollow square is drawn around the point of connection.

- Terminals of the representation have a colour that is determined by their type. The shape is dependent of the kind:

    - Input: a solid triangle pointing inwards the surrounding box.

    - Output: a solid triangle pointing outwards the surrounding box.

    - Input/Output: a solid square rotated 45 degrees.

    - Unused: a hollow square rotated 45 degrees.

- Symbol elements are drawn yellow.

- Surrounding box is green dotted.

- The context area shows the window and all instances while editing a circuit, or the surrounding box and the window in the representation area. Instance or surrounding box is a solid yellow box, the window is a solid green surface. If there is no current template the context area is empty.

All information is echoed on the screen except the attributes of the terminals and all the parameter information. Use the info command to show these.

The text has a size independent of the size of the current window. If the text of instance names and subsystem terminals becomes too large compared to the size of the instance they will not be drawn. The *-n* option overrides this.

System terminals have default attributes 'other' and kind 'input/output'. They inherit the attribute and kind from their name equivalent in the representation at verify or save time.

In most cases the screen is carefully updated if some object is deleted. To fasten drawing this is not always so: the deletion of a system terminal may cause a gap in the name of another system terminal. Use zoom in and zoom out function to get an implicit redraw if necessary.

1.29 Internal memory.

ESCHER uses its program heap to contain the graphical information of all the templates used in the session. A change to a template that was used before does not require disk access. *ESCHER* also allocates 2Kbyte for emergency cases. If the program runs out of memory it will perform the following actions:

- Give the message:
    - SYSTEM ERROR: <procedure name> Not enough memory
- Deallocate the emergency part.
- Calling the undo handler to undo the command under execution.
- Calling save for an autosave indicated with:
    - WARNING: Performing autosave.
- Leaving the editor without clearing the display.

In most of the cases this will result in an uncorrupted template, but there is no guarantee. Ask your system administrator for the maximum heap size of your programs. The size of a template normally varies from 10Kbyte to 40Kbyte, and is very close to the size of the graphical data.

1.30 Plot.

A drawing of the circuit or representation of the template can be made by *ESCHER*. It is in fact a call to a total independent program so drawings may also be made without the use of the editor. When called from *ESCHER* error messages will appear on the screen.

The exact shell command that is executed is represented by a string composed of the environment variable PLOTSHELL a flag '-c' or '-r', depending on circuit or representation area respectively, and the path name of the current template. The string given with the *-p* option is inserted. The user may edit the result.

A pipe is opened to the output of the shell command. See popen(2). The output is line by line displayed in the errorline. Every shell command might be directly executed by *ESCHER* in this way. Removing models can be done by giving the plot command and with the line edit command control-K clear the resulting line. Then type:

- rm_model <template pathname>

An example of the rm_model script:

```
for i in $*
do
        rm -r $i && echo "model $i deleted"
done
```

This script deletes complete directories, so beware!.

1.31 Environment.

The program understands the use of several specific UNIX environment variables. For example GRAPHICS_DEVICE is used to give the name of the graphical device, LIB_PATH contains the library path. New templates are gathered in the USER_LIB. *ESCHER* finds the plot program or plot shell script by means of PLOT_SHELL.

1.32 Portability.

ESCHER is written in C and has a LINT approval on portability. It uses the standard UNIX memory allocation functions. So it will run on every UNIX system. The use of the line editor which only uses functions from the graphics routines also improves portability. All drawings are clipped by *ESCHER* against the current window so the program is not dependent on clipping of the graphics software used. The network database is designed to be portable to other systems. A transfer of graphical information will work if the destination machine has pointers implemented as 32 bit words. Enumeration types in C have to be integers of 32 bit and starting from 0.

2. Commands description.

2.1 Introduction.

In this section each command is described in detail as follows:

- After 'M' the menu the command belongs to is mentioned. Some commands may be abbreviated or replaced by icons. More information was given in the description of the menu's.

- The 'A' stands for the area, circuit or representation, in which the command can be used. The commands in the main and edit menu are visible only while editing in the corresponding area.

- The 'D' is used for description of the command including the necessary input, coordinates or text, and the result if no errors are found. Description, input and result may be area dependent.

- After 'E' a list of possible error or warning messages is given if necessary accompanied by an explanation.

- The status of the command parser, idle, between or same, is given with the 'S'. Idle means no active command so the next input can only be the selection of one. Between does not interrupt current input of coordinates and same means that the command is still active after execution.

The commands are divided in groups:

- Manipulation.

- Edit.

- Item.

- Window.

- Attributes.

- Templates.

- Info.

- Menu.

- Grid.

- Area.

- Shell.

- Undo.

The manipulation commands are divided according to the current item. So the description of the 'move' command with the item 'line' can be found under 'move line'.

2.2 **Manipulation commands.**

2.2.1 *Add instance.*

M Edit.

A Circuit.

D The input consists of 2 parts: first a coordinate indicating the position of the new instance and second a name of the desired template. The library path is searched for the template and if found an instance is added to the circuit.

The middle of the surrounding box will agree with the coordinate input. A fine adjustment shall be made if the terminal positions do not correspond with the 4 by 4 grid.

A unique name for the instance is derived from the template name followed by an '_' and a number;

If possible the instance will be abutted to other instances.

Also an immediate connection to wires will be established. The user is not prompted for an offset because this will automaticly be set to the lower bound of the wire.

E

- ERROR: Incorrect template name
- ERROR: Can't instantiate current template
- ERROR: Can't find template <name>
- ERROR: Representation of <name> is empty
  The template of the desired instance does not have a surrounding box.
- ERROR: Can't place instance on wire
- ERROR: Terminal in new area
  An existing terminal conflicts with the new instance.
- ERROR: Instance overlap

- ERROR: Contiguous instances only allowed with abutted terminals
- ERROR: Number of terminals to abut doesn't match
- ERROR: Can't abut terminals of different type
- ERROR: Terminal width to small to establish connection
- ERROR: Can't connect unused terminal to wire
- ERROR: Can't abut terminals with connection
- ERROR: Can't abut output terminals
- ERROR: Can't abut input terminals
- ERROR: Can only abut unused terminals to unused terminals
- ERROR: Can't abut due to unmatching terminal positions
  One or more terminals does not have an equivalent to abut.
- WARNING: Library name discarded

S Same.

2.2.2 *Add wire.*

M Edit.

A Circuit.

D The input consist of 2 coordinates defining the start and endpoint of the wire. If the endpoint is not equal to a terminal position the second input is used as the first input for a new wire. If the 2 points are the same no wire is added but the user can enter a new first input.

The inputs may be adjusted to get a horizontal or vertical line. The conversion is always the one with the smallest angle involved.

When starting from or ending on a terminal of which no automatic offset can be derived the user is prompted for one.

Wires may not run along the sides of instances. With the -w option the user may change this.

If the new wire lengthens an existing one these 2 wires are merged to a single wire.

E

- ERROR: Position already occupied
- ERROR: Incompatible type
- ERROR: Terminal unused
- ERROR: Terminal already connected by abutment
- ERROR: Offset must be an integer
- ERROR: Range overflow
- ERROR: Offset too large
- ERROR: Offset too small
- ERROR: Wire crosses instance
- ERROR: Wire crosses terminal or node
- ERROR: Illegal wire
  New wire runs along existing one.
- WARNING: Last point adjusted
  ESCHER allows horizontal or vertical wires only.

S Same.

2.2.3 *Add terminal.*

M Edit.

A Circuit and representation.

D Circuit
The input coordinate is the position where a new terminal will be added to the circuit.

A unique name for it is derived consisting of the string "term" followed by a number. The string "term" may be changed with the *-t* option.

The terminal might be immediately connected to a wire, this also implies that it allowed on a wire, dividing it in 2 parts. If the offset can not automaticly be derived the user is prompted for it, for each wire once. In the prompt the direction of the wire is mentioned.

The new terminal may be on the crossing of 2 wires thus creating 4 new ones.

D Representation
The input coordinate is rounded to the nearest point on the solid grid and then used as the position of the new terminal on the surrounding box.

A unique name for it is derived consisting of the string "term" followed by a number. The string "term" may be changed with the *-t* option.

E Circuit

- ERROR: Terminal in instance
- ERROR: Duplicate terminal position
- ERROR: Offset must be an integer
- ERROR: Incompatible type
  No connection because terminal and wire have different types.
- ERROR: Up wire width greater than terminal width
- ERROR: Offset greater than low range of wire up
- ERROR: High range of wire up not connected
- ERROR: Down wire width greater than terminal width
- ERROR: Offset greater than low range of wire down
- ERROR: High range of wire down not connected
- ERROR: Left wire width greater than terminal width
- ERROR: Offset greater than low range of wire left
- ERROR: High range of wire left not connected
- ERROR: Right wire width greater than terminal width
- ERROR: Offset greater than low range of wire right
- ERROR: High range of wire right not connected

E Representation

- ERROR: No surrounding box

- ERROR: Terminal not on surrounding box
  Terminal is not allowed on corner of box either.

- ERROR: Duplicate terminal position

S Same.

2.2.4 *Add line.*

M Edit.

A Representation.

D The 2 input coordinates are the definitions of the start and endpoint of a new line. This line is added to the symbol.

If known the last input is used as the first input of the next add line command. This will not be true if an error occurs or the first input is the same as the last input. In the latter case no line is added but above mentioned mechanism is overruled.

E

- ERROR: Line outside surrounding box

S Same.

2.2.5 *Add circle.*

M Edit.

A Representation.

D The 2 inputs define a circle to be added to the symbol. The first input denotes the centre, the second is a circumference point.

E

- ERROR: Centre point equals circumference point

- ERROR: Circle outside surrounding box

S Same.

2.2.6 *Add arc.*

M Edit.

A Representation.

D Input is a collection of 3 coordinates, of which the last 2 are echoed with a rubber line to its predecessor. The arc through the input points will be added to the symbol definition.

Of the 3 possibilities of drawing an arc ending on 2 given points and going through another given point the one is selected that will end on the first and the last input and goes through the second.

E

- ERROR: Arc outside surrounding box

- ERROR: Points of arc coincide

- ERROR: Points of arc on straight line

S Same.

2.2.7 *Delete instance.*

M Edit.

A Circuit.

D The input coordinate corresponds to an instance which will be removed from the circuit.

Abutted instances will not be handled as a single item. To delete all of the abutted instances repeat the command as often as necessary.

E

- ERROR: No instance found

S Same.

2.2.8 *Delete wire.*

M Edit.

A Circuit.

D The wire indicated by the input is deleted.

If the input is on more then one wire the shortest one is chosen. In case of equal length the one which will be deleted is not predictable.

Deleting a wire of a set of 3 connected wires will merge the 2 remaining ones to 1 wire if possible.

E

- ERROR: No wire found

S Same.

2.2.9 *Delete terminal.*

M Edit.

A Circuit and representation.

D The input denotes the terminal to be removed. If, in the circuit area, the terminal is connected with more then 1 wire the wires will remain connected. In the representation the input is converted to the nearest point on the solid grid.

If the terminal was on 2 wires of which one can be seen as the extension of the other the 2 wires are combined to 1 line.

E Circuit

- ERROR: No terminal found

E Representation

- ERROR: No surrounding box

- ERROR: No terminal found

S Same.

2.2.10 *Delete line.*

M Edit.

A Representation.

D The line which can be found within 1 fine grid distance from the input coordinate is removed from the symbol.

In case more then 1 line can be found a random selection is made.

E

- ERROR: No line found

S Same.

2.2.11 *Delete circle.*

M Edit.

A Representation.

D The circle which can be found within 1 fine grid distance from the input coordinate is removed from the symbol.

In case more then 1 circle can be found a random selection is made.

E

- ERROR: No circle found

S Same.

2.2.12 *Delete arc.*

M Edit.

A Representation.

D The arc which can be found within 1 fine grid distance from the input coordinate is removed from the symbol.

In case more then 1 arc can be found a random selection is made.

E

- ERROR: No arc found

S Same.

2.2.13 *Move instance.*

M Edit.

A Circuit.

D Two input coordinates define a vector over which the instance, or abutted instances, indicated by the first input will be replaced.

To ensure that the terminals will stay on the 4 * 4 grid while working with a maze of 2 or 1 the vector may be adjusted.

If possible the instance, or abutted instances, will be abutted to other instances.

If terminals will be placed on ends of existing wires and if an automatic offset can be generated this connection will be made.

E

- ERROR: No instance found
- ERROR: Can't move connected instance
- ERROR: Can't place instance on wire
- ERROR: Terminal in new area
  An existing terminal conflicts with the instance.
- ERROR: Instance overlap
- ERROR: Can't connect unused terminal to wire
- ERROR: Contiguous instances only allowed with abutted terminals
- ERROR: Number of terminals to abut doesn't match
- ERROR: Can't abut terminals of different type
- ERROR: Terminal width to small to establish connection
- ERROR: Can't abut terminals with connection
- ERROR: Can't abut output terminals
- ERROR: Can't abut input terminals
- ERROR: Can only abut unused terminals to unused terminals
- ERROR: Can't abut due to unmatching terminal positions
  One or more terminals does not have an equivalent to abut.

S Same.

2.2.14 *Move terminal.*

M Edit.

A Circuit and representation.

D Circuit
Two input coordinates define a vector over which the terminal indicated by the first input will be moved.

If the terminal will be moved to a point on 1 or more wires the connections are established. If the offset is not implicit the user is prompted for it. In the prompt the direction of the wire relative to the new terminal position is mentioned.

If the old position of the terminal was on 2 wires of which one can be seen as the extension of the other the 2 wires are combined to 1 line.

D Representation
Two input coordinates are converted to points on the solid grid and then used to define a vector over which the terminal indicated by the first input will be moved.

E Circuit

- ERROR: No terminal found
- ERROR: Destination equals origin
- ERROR: Terminal in instance
- ERROR: Duplicate terminal position
- ERROR: Can't place unused terminal on wire
- ERROR: Incompatible type
  Conflict of type of terminal and wire.
- ERROR: Offset must be an integer
- ERROR: Up wire width greater than terminal width
- ERROR: Offset greater than low range of wire up
- ERROR: High range of wire up not connected
- ERROR: Down wire width greater than terminal width
- ERROR: Offset greater than low range of wire down
- ERROR: High range of wire down not connected
- ERROR: Left wire width greater than terminal width
- ERROR: Offset greater than low range of wire left
- ERROR: High range of wire left not connected
- ERROR: Right wire width greater than terminal width

- ERROR: Offset greater than low range of wire right

- ERROR: High range of wire right not connected

E Representation

- ERROR: No terminal found

- ERROR: New terminal position not on surrounding box

- ERROR: New point on corner

- ERROR: Target equals origin

- ERROR: New terminal position already occupied

S Same.

2.2.15 *Move line.*

M Edit.

A Representation.

D The two inputs are used to define a vector over which the line found within less than 1 fine grid distance of the first input is translated.

If more then 1 line is found within the given distance of the first input the one chosen will be unpredictable.

E

- ERROR: No line found

- ERROR: New line outside box

S Same.

2.2.16 *Move circle.*

M Edit.

A Representation.

D The two inputs are used to define a vector over which the circle found within less than 1 fine grid distance of the first input is translated.

If more then 1 circle is found within the given distance of the first input the one chosen will be unpredictable.

E

- ERROR: No circle found

- ERROR: New circle outside box

S Same.

2.2.17 *Move arc.*

M Edit.

A Representation.

D The two inputs are used to define a vector over which the arc found within less than 1 fine grid distance of the first input is translated.

If more then 1 arc is found within the given distance of the first input the one chosen will be unpredictable.

E

- ERROR: No arc found

- ERROR: New arc outside box

S Same.

2.2.18 *Copy instance.*

M Edit.

A Circuit.

D The first input indicates the instance, or a set of abutted instances, that will be copied to the location of the second input.

For each instance involved an unique name is derived from the template name of the instance a '_' and a number.

If possible the new instance, or instances are abutted to existing instances.

If terminals of the instances are placed on ends of wires and if the offset for this connection can be derived this connection will also be made.

The difference between the 2 inputs is used as a vector over which a translation of the instances is made. While working with

a maze of 1 or 2 a slight correction can be made to ensure that the instance terminals will lie on the 4 * 4 grid.

E

- ERROR: No instance found
- ERROR: Can't place instance on wire
- ERROR: Terminal in new area
  An existing terminal conflicts with the new instance.
- ERROR: Instance overlap
- ERROR: Can't connect unused terminal to wire
- ERROR: Contiguous instances only allowed with abutted terminals
- ERROR: Number of terminals to abut doesn't match
- ERROR: Can't abut terminals of different type
- ERROR: Terminal width to small to establish connection
- ERROR: Can't abut terminals with connection
- ERROR: Can't abut output terminals
- ERROR: Can't abut input terminals
- ERROR: Can only abut unused terminals to unused terminals
- ERROR: Can't abut due to unmatching terminal positions
  One or more terminals does not have an equivalent to abut.

S Same.

2.2.19 *Copy terminal.*

M Edit.

A Circuit and representation.

D Circuit.
The terminal indicated by the first input is copied to the second input.

If the new terminal is placed on 1 or more wires the connections will be established. If necessary the user is prompted for an offset. In the prompt the direction of the wire relative to the

new terminal is mentioned.

D Representation.
The terminal indicated by the first input is copied to the second input. The inputs are first converted to the nearest point on the solid grid.

E Circuit

- ERROR: No terminal found
- ERROR: Destination equals origin
- ERROR: Terminal in instance
- ERROR: Duplicate terminal position
- ERROR: Can't place unused terminal on wire
- ERROR: Incompatible type
- ERROR: Offset must be an integer
- ERROR: Up wire width greater than terminal width
- ERROR: Offset greater than low range of wire up
- ERROR: High range of wire up not connected
- ERROR: Down wire width greater than terminal width
- ERROR: Offset greater than low range of wire down
- ERROR: High range of wire down not connected
- ERROR: Left wire width greater than terminal width
- ERROR: Offset greater than low range of wire left
- ERROR: High range of wire left not connected
- ERROR: Right wire width greater than terminal width
- ERROR: Offset greater than low range of wire right
- ERROR: High range of wire right not connected

E Representation

- ERROR: No terminal found

- ERROR: New terminal position not on surrounding box
- ERROR: New point on corner
- ERROR: Target equals origin
- ERROR: New terminal position already occupied

S Same.

2.2.20 *Copy line.*

M Edit.

A Representation.

D The line which can be found within 1 fine grid distance of the input is copied to the position indicated by the second input.

If more than one line is found the one chosen is not predictable.

The difference between the 2 inputs is used as a vector over which a translation is made. This guarantees that the length and direction of the line is not changed and that the endpoints of the new line will be on the fine grid.

E

- ERROR: No line found
- ERROR: New line outside box

S Same.

2.2.21 *Copy circle.*

M Edit.

A Representation.

D The circle which can be found within 1 fine grid distance of the input is copied to the spot indicated by the second input.

If more than one circle is found the one chosen is not predictable.

The difference between the 2 inputs is used as a vector over which a translation is made. This guarantees that the radius of the circle is not changed and that the centre of the new circle will be on the fine grid.

E

- ERROR: No circle found

- ERROR: New circle outside box

S Same.

2.2.22 *Copy arc.*

M Edit.

A Representation.

D The arc which can be found within 1 fine grid distance of the input is copied to the spot indicated by the second input.

If more than one arc is found the one chosen is not predictable.

The difference between the 2 inputs is used as a vector over which a translation is made. This guarantees that the endpoints of the new arc will be on the fine grid.

E

- ERROR: No arc found

- ERROR: New arc outside box

S Same.

2.3 Edit commands.

2.3.1 *Mirror_x instance.*

M Edit.

A Circuit.

D The input denotes an instance or a set of abutted instances that will be mirrored over a vertical axis through the centre of the instance.

In case the instance is abutted all the instances involved are mirrored. The centre of the rectangle that contains all the instances is used to determine the vertical axis.

With the *-m* option the meaning of the mirror_x and mirror_y command may be interchanged.

E

- ERROR: No instance found
- ERROR: Can't mirror/rotate connected instance
- ERROR: Can't place instance on wire
- ERROR: Possible instance touch or overlap
- ERROR: Wire or terminal in new area
  An existing terminal conflicts with the new position of the instance.

S Same.

2.3.2 *Mirror_y instance.*

M Edit.

A Circuit.

D The input denotes an instance or a set of abutted instances that will be mirrored over a horizontal axis through the centre of the instance.

In case the instance is abutted all the instances involved are mirrored. The centre of the rectangle that contains all the instances is used to determine the horizontal axis.

With the *-m* option the meaning of the mirror_x and mirror_y command may be interchanged.

E

- ERROR: No instance found
- ERROR: Can't mirror/rotate connected instance
- ERROR: Can't place instance on wire
- ERROR: Possible instance touch or overlap
- ERROR: Wire or terminal in new area
  An existing terminal conflicts with the new position of the instance.

S Same.

2.3.3 *Rotate instance.*

M Edit.

A Circuit.

D The input denotes an instance or a set of abutted instances that will be rotated over the centre of the instance. The rotation is 90 degrees counter clockwise.

In case the instance is abutted all the instances involved are rotated. The centre of the rectangle that contains all the instances is used to determine the centre.

E

- ERROR: No instance found
- ERROR: Can't mirror/rotate connected instance
- ERROR: Can't place instance on wire
- ERROR: Possible instance touch or overlap
- ERROR: *Wire or terminal in new area*
  An existing terminal conflicts with the new position of the instance.

S Same.

2.3.4 *Split.*

M Edit.

A Circuit and representation.

D Two coordinate inputs define the split vector. This description assumes that the second input lies to the right and above the first input. Other possibilities of the relative positions will be handled similar.

The split command is sequentially executed in 2 parts: a horizontal and a vertical split. Only the horizontal part is described.

The horizontal split defines a vertical split line through the first input and a horizontal split vector being the horizontal projection of the split vector.

In the circuit area are instances, terminals and wires to the right of split line translated over the horizontal split vector.

Wires crossing the split line are lengthened with the length of the horizontal split vector.

If there are abutted terminals on the splitline the abutment will be replaced by a wire with length equal to the horizontal split vector. The wire has the same type and width as the terminals.

In the circuit area the split line is not allowed to cross instances.

In the representation area the terminals, end points of lines, arc and circles to the right of the split line are translated over the horizontal split vector. If a surrounding box is defined it is adjusted so the newly placed symbol elements will be in the new box and the terminals will stay on its edges.

To determine whether a circle or arc is to the right of the split line the centre of the circle, in case of an arc the circle it is part of, is used.

E Circuit

- ERROR: Split points coincide
- ERROR: Split line crosses instance
- WARNING: Circuit of <name> is empty

E Representation

- ERROR: First split input outside surrounding box
- ERROR: Split points coincide
- WARNING: Representation of <name> is empty

S Same, idle on warning.

2.3.5 *Edit name.*

M Main.

A Circuit and representation.

D The input indicates a terminal, in the circuit area also an instance, of which the name can be changed using text edit.

The old name is used as default. The user may change this with the *-e* option resulting in a text edit with no default.

E *Circuit*

- ERROR: No instance or terminal found
- ERROR: Empty terminal name not allowed
- ERROR: Duplicate terminal name
- ERROR: Empty instance name not allowed
- ERROR: Duplicate instance name

E Representation

- ERROR: No terminal found
- ERROR: Empty terminal name not allowed
- ERROR: Duplicate terminal name

S Same.

2.3.6 *Edit parameters*.

M Main.

A Circuit and representation.

D Circuit
The input is used to determine an instance of which the actual parameters have to be edited. The edit is a text edit for which the info/verify area is used. This area is cleared if necessary. All the actual parameters are handled sequentially as follows: The user is prompted to enter the value of the parameter. The prompt is the name of the parameter followed by an '='. As default is the last known value given and if not known the *default value of the formal parameter*. As result, the user can give an expression consisting of constants, function calls, operators and formal parameter names. The syntax corresponds with electrical conventions for multipliers like 10k, 100n and 1M. See the appendix.

If the expression will not fit on 1 line the user may continue on the next line by giving a '\' as last character on the line. The expression may be spread out over at most 6 lines.

If the expression contains more than 1 line it is concatenated after the edit. While editing again the so formed expression *acts as default and* will be automaticly divided in more lines each ending with a '\'. There is no certainty that this division equals the original one. In rare cases is it imaginable that

characters may be lost by this line splitting.

If the last parameter is handled the info/verify area will be cleared.

D Representation
In the representation this command means an edit, add, delete or change, of the formal parameters of the template.

For the edit the info/verify area is used and first cleared if necessary.

The user is sequentially prompted for the name, kind, range and default value of a parameter. The edit of the kind is accompanied in the error line with

- Kind is i(nteger), b(ool) or (default) r(eal)

giving the 3 possibilities for the kind. Upper case characters may be used. Default is 'R'.

The format of the range is also seen in the error line:

- Format: number, number

After entering the name of the parameter the editor checks the existence of it. If so the user is asked to delete it or not. Default is no deletion. If the user wishes not to delete, the change of the parameter is done with known information as default.

Entering an empty name will stop the edit and clear the info/verify area.

E Circuit

- ERROR: No instance found

- ERROR: Template of this instance has no formals

- WARNING: Template has no instances

S In the circuit area same and in the representation idle.

2.3.7 *Edit box.*

M Edit.

A Representation.

D The 2 coordinate inputs, the last under rubber box echo, are converted to the nearest point on the solid grid and then used as the new surrounding box.

Existing terminals which can not be placed on the new box will be deleted.

E

- ERROR: Box too small
  The 2 inputs equalise after conversion to the solid grid.

- ERROR: Symbol element outside surrounding box

- WARNING: Terminal(s) deleted

S Same.

2.4 **Item commands.**

2.4.1 *Change item to instance.*

M Status.

A Circuit.

D Further manipulation will be done with instance as current item.

S Between.

2.4.2 *Change item to wire.*

M Status.

A Circuit.

D Further manipulation will be done with wire as current item.

S Between.

2.4.3 *Change item to terminal.*

M Status.

A Circuit and representation.

D Further manipulation will be done with terminal as current item.

S Between.

2.4.4 *Change item to line.*

M Status.

A Representation.

D Further manipulation will be done with line as current item.

S Between.

2.4.5 *Change item to circle.*

M Status.

A Representation.

D Further manipulation will be done with circle as current item.

S Between.

2.4.6 *Change item to arc.*

M Status.

A Representation.

D Further manipulation will be done with arc as current item.

S Between.

2.5 **Window commands.**

2.5.1 *Zoom in.*

M Status.

A Circuit and representation.

D The size of the current window is multiplied by 0.7. A redraw is made with the new window size.

S Between.

2.5.2 *Zoom out.*

M Status.

A Circuit and representation.

D The size of the current window is divided by 0.7. A redraw is made with the new window size.

S Between.

2.5.3 *Window world.*

M Status.

A Circuit and representation.

D The size of the current window is adjusted so that the complete circuit or representation will become visible. If the current area is empty a default window size is used. A redraw is made with the new window size.

E

- WARNING: Circuit of <name> is empty

- WARNING: Representation of <name> is empty

S Between.

2.5.4 *Window box.*

M Edit.

A Circuit and representation.

D Two coordinates, first with crosshair and second with rubber box echo, define the left under and right upper corner of the desired window. A best fit to the work area defines the new window with which a redraw is made.

E

- ERROR: Box becomes too small
  First input equalises second.

S Same.

2.5.5 *Window centre.*

M Context.

A Circuit and representation.

D A coordinate selected in the context area will be interpreted as the centre of the new window. The size of the window is not changed. A redraw is made with the new window.

S Between.

2.6 **Attributes commands.**

2.6.1 *Change type.*

M Status.

A Circuit and representation.

D Selecting this particular part of the status menu toggles the type of new added terminals and wires between SIG (signal) and EL (electrical).

The type of existing terminals can not be changed. Use delete and add terminal to do so.

S Between.

2.6.2 *Change kind.*

M Status.

A Circuit and representation.

D This status menu part controls the kind of new terminals. Selecting this command will change it from IO (input/output) to IN (input) and from IN to OUT (output), further to U (unused) and back to IO.

The kind of a terminal is only active when adding one to the representation. Circuit terminals will inherit the kind of its representation equivalent during network database update time. If there is no name equivalent in the representation the default kind will be input/output.

The kind of existing terminals can not be changed. Use delete and add terminal to do so.

S Between.

2.6.3 *Change attribute.*

M Status.

A Circuit and representation.

D This status menu part is used to select the attribute of new terminals. Chosing this command will change it from OTR (other) to TRI (tristate) and from TRI to PU (pull up), further to PD (pull down) and back to OTR.

The *attribute of a terminal is only* active when adding one to the representation. Circuit terminals will inherit the attribute of its representation equivalent during network database update time. If there is no name equivalent in the representation the default attribute will be other.

The attribute of existing terminals can not be changed. Use delete and add terminal to do so.

S Between.

2.6.4 *Change offset.*

M Status.

A Circuit and representation.

D After selecting this command the status menu is replaced by an array of numbers in the range of 0 to 16 followed by the character 'o' (other) of which one may be chosen. Further adding of wires will be done with the new offset.

If the user wants a number greater than 16 he may use the 'o'. He is then prompted to enter a number. The maximum offset is 99.

E

- ERROR: Incorrect number

- WARNING: Number set to 99

S Between.

2.6.5 *Change bus width.*

M Status.

A Circuit and representation.

D After selecting this command the status menu is replaced by an array of numbers in the range of 0 to 16 followed by the character 'o' (other) of which one may be chosen. New wires and terminals will have a width equal to bus width.

If the user wants a number greater than 16 he may use the 'o'. He is then prompted to enter a number. The minimum bus width is 1 and maximum bus width is 99.

E

- ERROR: Incorrect number

- WARNING: Number set to 99

- WARNING: Bus set to 1
  The user has tried to set the bus width to 0.

S Between.

2.7 Template commands.

2.7.1 *Change template.*

M Main.

A Circuit and representation.

D The user is prompted for a template name which will become the new current template. The library path is searched for the new model. If it can not be found it is assumed that it belongs in the user library. The user is informed about this in the error line:

- New template <template pathname>

If there is a old template the information of it is saved on disk using an autosave. Autosave is performed before the new template prompt.

The new template is, if it is an existing one, checked on consistency of its instances.

The edit of the new template starts in the circuit area.

The name of the old template is pushed on the template stack so it may be used by the level up command.

E

- ERROR: Incorrect template name

- ERROR: Instance does not fit
  A soft replacement is performed with negative result. The edit of the new template is rejected.

- WARNING: Given library name discarded

- WARNING: Instance(s) updated or deleted
  A hard replacement is performed with negative result or the representation of template(s) or instance(s) is modified after the instance time of some instance(s) of the new

template.

- WARNING: Error in network; network files cleared

- WARNING: Circuit of <name> is empty

S Idle.

2.7.2 *Level up.*

M Main.

A *Circuit and representation.*

D The model name on the top of the template stack will become the new current template.

If there is a current template the information of it is saved on disk using an autosave.

The new template is, if it is an existing one, checked on consistency of its instances.

The edit of the new template starts in the circuit area.

E

- ERROR: Stack empty

- ERROR: Instance does not fit
  A soft replacement is performed with negative result. The edit of the new template is rejected.

- WARNING: Instance(s) updated or deleted
  A hard replacement is performed with negative result or the representation of template(s) or instance(s) is modified *after the instance time of some* instance(s) of the new template.

- WARNING: Error in network; network files cleared

- WARNING: Circuit of <name> is empty

S Idle.

2.7.3 *Level down.*

M Main.

A Circuit.

D the input coordinate is used to search for an instance of which the template will become the new current template.

If there is a current template the information of it is saved on disk using an autosave.

The new template is checked on consistency of its instances.

The edit of the new template starts in the circuit area.

The name of the old template is pushed on the template stack so it may be used by the level up command.

E

- ERROR: No instances present
- ERROR: No instance found
- ERROR: Instance does not fit
  A soft replacement is tried with negative effect. The edit of the new template is rejected.
- WARNING: Instance(s) updated or deleted
  A hard replacement is performed with negative result or the representation of template(s) or instance(s) is modified after the instance time of some instance(s) of the new template.
- WARNING: Error in network; network files cleared
- WARNING: Circuit of <name> is empty

S Same if there are instances in the new circuit area else idle.

2.7.4 *Save*.

M Main.

A Circuit and representation.

D The graphical information of the current template is saved on the disk. The user is prompted for a template pathname. Default is the pathname known by *ESCHER*.

Saving a template implies a verify. If successful the network database files are updated too else they are cleared to ensure consistency of the network data base files with the graphical information.

E

- ERROR: Incorrect path

- WARNING: Error in network; network files cleared

S Idle.

2.7.5 *Abort*.

M Main.

A Circuit and representation.

D Edit of the current template is stopped. The disk is not updated.

After the command there is no current template so the change template or level up command is needed.

S Idle.

2.7.6 *Exit*.

M Main.

A Circuit and representation.

D The *ESCHER* program stops and control is returned to the shell.

The current template is not saved.

2.8 **Info commands**.

2.8.1 *Info*.

M Edit.

A Circuit and representation.

D Circuit.
If the command is selected and the info/verify area contains info information and there is more information this will be displayed else the info/verify area will be cleared. The user may give a coordinate input referring to an instance of which the info information is to be displayed. If there is information of the selected instance present the next part of it is shown. In case of no information or the selected instance differs from the previous or there is no more to display the information of the instance is extracted and the next part is shown.

New information will erase existing verify information.

Repeated selection of the info command will result in a clear of the info/verify area if information is exhausted.

D Representation.
If the command is selected and the info/verify area contains information and there is more information this will be displayed else the info/verify area will be cleared.

If there is no information the representation info will be extracted and the first part displayed.

New information will erase existing verify information.

Repeated selection of the info command will result in a remove of the info/verify area if information is exhausted.

E

- ERROR: No instance found

- WARNING: Template has no instances

S Same in circuit area, idle in representation.

2.8.2 *Verify.*

M Main.

A Circuit and representation.

D If verify information exists and there is more to display this will be displayed else the info/verify area is cleared. If there is no verify information this is extracted from the template and the first part of it displayed.

Existing info information will always be destroyed.

Repeated use of this command will result in an clear of the info/verify area.

E

- Verify: no errors found

S Idle.

2.8.3 *View.*

M Main.

A Circuit and representation.

D Circuit.
The user input is used to search for an instance of which the circuit is to appear in the view area. The view area is cleared on the selection of this command.

To clear the view area issue this command but instead of a coordinate input another command.

D Representation
If a the view area is clear the circuit of the current template is shown else the view area is cleared.

E

- ERROR: Template has no instances

- ERROR: No instance found

- WARNING: Circuit of <name> is empty

S Same in circuit area, idle in representation.

2.9 **Menu commands.**

2.9.1 *Main menu.*

M Edit.

A Circuit and representation.

D The main menu will become the new menu.

S Idle.

2.9.2 *Edit menu.*

M Main.

A Circuit and representation.

D The edit menu will become the new menu.

S Idle.

2.10 **Grid commands.**

2.10.1 *Toggle grid.*

M Status.

A Circuit and representation.

D If the grid was visible it will become invisible else it will become visible. If it really will be shown depends on the size of the current window.

S Between.

2.11 **Area commands.**

2.11.1 *Toggle area.*

M Main and edit.

A Circuit and representation.

D This command toggles the current area between circuit and representation. Depending of the new current area the circuit or representation will be redrawn. This commands implies an window world command with a default window size if the new area is empty. Going to the circuit area will cause the current item to be the instance. Arriving in the representation area it will be the terminal.

E

- WARNING: Circuit of <name> is empty
- WARNING: Representation of <name> is empty

S Idle.

2.12 **Shell commands.**

2.12.1 *Shell escape.*

M Main.

A Circuit and representation.

D An escape to the shell is made. On return the graphics device is re-initialised and the screen redrawn.

The process used as shell is determined by the environment variable SHELL.

S Idle.

2.12.2 *Plot.*

M Main.

A Circuit and representation.

D The current template is saved on disk using autosave. A string is composed of the PLOT_SHELL environment variable and the template pathname. Depending on the area a "-c" (circuit) or "-r" (representation) is inserted. The resulting string is offered for a line edit. The probably edited string is used as a shell command to which a pipe is opened in reading I/O mode (see popen(3S)). The output of this pipe is displayed in the error area, line by line. The interval of the line display is equal to the display error time.

E

- ERROR: Plot command environment variable undefined
- WARNING: Performing autosave
- WARNING: Error in network; network files cleared

S Idle.

2.13 **Undo commands.**

2.13.1 *Undo.*

M Edit.

A Circuit and representation.

D The command on the top of the undo stack is undone. The work area is redrawn if the command did not imply an edit of a parameter.

E

- WARNING: Undo of actual parameter edit
- WARNING: Undo of formal parameter edit
- WARNING: Empty undo stack

S Idle.

Appendix: Network database as used by ESCHER.

In the network database each model has his own directory. Each program working on the network database is invited to have his own files in the model directory.

In this appendix only the fields *ESCHER* uses are described. See for the general format the paper mentioned in the manual.

Interface routines are found in lNDI.

ESCHER uses the following files:

- *escher_data*.
  A dump of graphical information of the template. In binary format so unreadable.

- *call*.
  Each record denotes the properties of an instance with successive elements:

  - String: instance name.

  - Number: not used, always zero.

  - String: pathname of the model directory of the template of the instance.

  - String: concatenation of the actual parameters of the instance separated with a ';'. Each parameter has a format of: <name> = <expression>. Could be empty.

  - String: not used, always empty.

- *connection*.
  Each record denotes 2 connected terminals.

  - String: net description consisting of a netname, being 'net' followed by a positive 4 digit integer, a number representing the input behaviour and a number denoting the output behaviour. Coding of these numbers:

    - 0: other

    - 1: tristate

    - 2: pull down

    - 3: pull up

        - 4: unknown

    - String: origin name of the first terminal. Empty for system terminals, instance name for subsystem terminals.

    - String: first terminal name.

    - Number: range number of first terminal.

    - String: origin name of the second terminal. Empty for system terminals, instance name for subsystem terminals.

    - String: second terminal name.

    - Number: range number of second terminal.

    - Number: not used, always 0.

    - Number: not used, always 0.

    - String: not used, always empty.

  All terminals with the same net are connected to each other. In case the records have the same netnumber the first terminal is always the same.

- *interface*.
  Giving the properties of terminals and formal parameters. If it is a terminal:

    - String: terminal name

    - Number: always 0. Meaning this is a terminal.

    - Number:

        - 0: Electrical type.

        - 1: Signal type.

    - Number: Always zero.

    - Number: terminal range number. Remark that every terminal of a multiple terminal has his own record.

    - String: Terminal attribute and kind, a concatenation of numbers separated by spaces. Code of the numbers:

        - 1: input

- 2: output
- 3: input/output
- 4: unused
- 5: pull_up
- 6: pull_down
- 7: tristate
- 8: other

If it is a parameter:

- String: name of the formal.
- Number: Always 1 denoting a formal parameter.
- Number: kind:
  - 0: integer.
  - 1: float.
  - 2: bool.
- Number: low bound of range
- Number: high bound of range.
- Number: Existing default:
  - 0: no default.
  - 1: default specified.
- Number: default value, not existent if default is not specified.
- String: Not used, always 0.

IF there is no range specified both numbers will be 0.

A save command, including the change template commands implying a save, update always the *escher_data* file. When entering the template again for a re-edit the data of this file is read into internal memory.

The network files, *call*, *interface* and *connection*, are written if the network is consistent, that is the verify commands will only show warnings no errors, else they are cleared, that is they do not contain data. The files are not destroyed.

Appendix: BNF notation of the syntax of actual parameters.

Actual parameters must have the following syntax:

```
<ACTUAL PARAMETER> ::= <identifier> "=" <expression>

<identifier> ::= (<letter> | "_") (<letter> | <digit> | "_" | ".")

<expression> ::= [<unary_operator>] <term> (<operator> ["!"] <term>)

<term> ::=   "(" <expression> ")"
           | <identifier>    must be the name of a formal parameter
           | <value>
           | "SIN" "(" <expression> ")"
           | "COS" "(" <expression> ")"
           | "TAN" "(" <expression> ")"
           | "ARCSIN" "(" <expression> ")"
           | "ARCCOS" "(" <expression> ")"
           | "ARCTAN" "(" <expression> ")"
           | "EXP" "(" <expression> ")"
           | "LN" "(" <expression> ")"
           | "LOG" "(" <expression> ")"
           | "ABS" "(" <expression> ")"
           | "ROUND" "(" <expression> ")"
           | "COUNT" "(" <expression> ")"
           | "MAX" "(" <expression_list> ")"
           | "MIN" "(" <expression_list> ")"
           | "FANOUT" "(" <identifier> { "," <expression> } ")"
                  identifier must be the name of a template terminal
                  expression must be range number of that terminal

<expression_list> := <expression> "," <expression> ["," <expression>]+

<unary_operator> ::=   "+"
                     | "-"
                     | "!"

<operator> ::=   "+"
               | "-"
               | "/"
               | "*"
               | "%"
               | "**"
               | ">"
               | "<"
               | "=="
               | "<="
               | ">="
               | "||"
               | "&&"
               | "!="
```

```
<value> ::=   <integer>
            | <real> [<multiplier>]

<multiplier> ::=   "k"          multiply factor is 1.0e+3
                 | "m"                             1.0e-3
                 | "M"                             1.0e+6
                 | "G"                             1.0e+9
                 | "u"                             1.0e-6
                 | "p"                             1.0e-12
                 | "n"                             1.0e-9
                 | "f"                             1.0e-15

<integer> ::=  {<digit>}+

<real> ::=   {<digit>}* "." {<digit>}+
           | {<digit>}* "." {<digit>}+ "E" <integer>

sign       ::=   "+"
               | "-"
```

During verify time the syntax is checked only, not evaluated. The parameters are written in the network database as a string. Semantics and evaluation of the expression has to be done by programs reading the network database.

The above mentioned constraints on identifiers are checked and errors found are reported.

Eindhoven University of Technology Research Reports
Department of Electrical Engineering

ISSN 0167-9708
Coden: TEUEDE

(138) Nicola, V.F.
A SINGLE SERVER QUEUE WITH MIXED TYPES OF INTERRUPTIONS: Application to the modelling of checkpointing and recovery in a transactional system.
EUT Report 83-E-138. 1983. ISBN 90-6144-138-2

(139) Arts, J.G.A. and W.F.H. Merck
TWO-DIMENSIONAL MHD BOUNDARY LAYERS IN ARGON-CESIUM PLASMAS.
EUT Report 83-E-139. 1983. ISBN 90-6144-139-0

(140) Willems, F.M.J.
COMPUTATION OF THE WYNER-ZIV RATE-DISTORTION FUNCTION.
EUT Report 83-E-140. 1983. ISBN 90-6144-140-4

(141) Heuvel, W.M.C. van den and J.E. Daalder, M.J.M. Boone, L.A.H. Wilmes
INTERRUPTION OF A DRY-TYPE TRANSFORMER IN NO-LOAD BY A VACUUM CIRCUIT-BREAKER.
EUT Report 83-E-141. 1983. ISBN 90-6144-141-2

(142) Fronczak, J.
DATA COMMUNICATIONS IN THE MOBILE RADIO CHANNEL.
EUT Report 83-E-142. 1983. ISBN 90-6144-142-0

(143) Stevens, M.P.J. en M.P.M. van Loon
EEN MULTIFUNCTIONELE I/O-BOUWSTEEN.
EUT Report 84-E-143. 1984. ISBN 90-6144-143-9

(144) Dijk, J. and A.P. Verlijsdonk, J.C. Arnbak
DIGITAL TRANSMISSION EXPERIMENTS WITH THE ORBITAL TEST SATELLITE.
EUT Report 84-E-144. 1984. ISBN 90-6144-144-7

(145) Weert, M.J.M. van
MINIMALISATIE VAN PROGRAMMABLE LOGIC ARRAYS.
EUT Report 84-E-145. 1984. ISBN 90-6144-145-5

(146) Jochems, J.C. en P.M.C.M. van den Eijnden
TOESTAND-TOEWIJZING IN SEQUENTIËLE CIRCUITS.
EUT Report 85-E-146. 1985. ISBN 90-6144-146-3

(147) Rozendaal, L.T. en M.P.J. Stevens, P.M.C.M. van den Eijnden
DE REALISATIE VAN EEN MULTIFUNCTIONELE I/O-CONTROLLER MET BEHULP VAN EEN GATE-ARRAY.
EUT Report 85-E-147. 1985. ISBN 90-6144-147-1

(148) Eijnden, P.M.C.M. van den
A COURSE ON FIELD PROGRAMMABLE LOGIC.
EUT Report 85-E-148. 1985. ISBN 90-6144-148-X

(149) Beeckman, P.A.
MILLIMETER-WAVE ANTENNA MEASUREMENTS WITH THE HP8510 NETWORK ANALYZER.
EUT Report 85-E-149. 1985. ISBN 90-6144-149-8

(150) Meer, A.C.P. van
EXAMENRESULTATEN IN CONTEXT MBA.
EUT Report 85-E-150. 1985. ISBN 90-6144-150-1

(151) Ramakrishnan, S. and W.M.C. van den Heuvel
SHORT-CIRCUIT CURRENT INTERRUPTION IN A LOW-VOLTAGE FUSE WITH ABLATING WALLS.
EUT Report 85-E-151. 1985. ISBN 90-6144-151-X

(152) Stefanov, B. and L. Zarkova, A. Veefkind
DEVIATION FROM LOCAL THERMODYNAMIC EQUILIBRIUM IN A CESIUM-SEEDED ARGON PLASMA.
EUT Report 85-E-152. 1985. ISBN 90-6144-152-8

(153) Hof, P.M.J. Van den and P.H.M. Janssen
SOME ASYMPTOTIC PROPERTIES OF MULTIVARIABLE MODELS IDENTIFIED BY EQUATION ERROR TECHNIQUES.
EUT Report 85-E-153. 1985. ISBN 90-6144-153-6

(154) Geerlings, J.H.T.
LIMIT CYCLES IN DIGITAL FILTERS: A bibliography 1975-1984.
EUT Report 85-E-154. 1985. ISBN 90-6144-154-4

(155) Groot, J.F.G. de
THE INFLUENCE OF A HIGH-INDEX MICRO-LENS IN A LASER-TAPER COUPLING.
EUT Report 85-E-155. 1985. ISBN 90-6144-155-2

(156) Amelsfort, A.M.J. van and Th. Scharten
A THEORETICAL STUDY OF THE ELECTROMAGNETIC FIELD IN A LIMB, EXCITED BY ARTIFICIAL SOURCES.
EUT Report 86-E-156. 1986. ISBN 90-6144-156-0

(157) Lodder, A. and M.T. van Stiphout, J.T.J. van Eijndhoven
ESCHER: Eindhoven SCHematic EditoR reference manual.
EUT Report 86-E-157. 1986. ISBN 90-6144-157-9

(158) Arnbak, J.C.
DEVELOPMENT OF TRANSMISSION FACILITIES FOR ELECTRONIC MEDIA IN THE NETHERLANDS.
EUT Report 86-E-158. 1986. ISBN 90-6144-158-7