SIMetrix

FILE FORMAT MANUAL

VERSION 8.3

MAY 2019

SIMetrix File Format Manual

Trademarks:
PSpice is a trademark of Cadence Design Systems Inc.

SIMetrix Technologies Ltd.,
78 Chapel Street,
Thatcham,
Berkshire
RG18 4QN
United Kingdom

Tel:	+44 1635 866395
Fax:	+44 1635 868322
Email:	support@simetrix.co.uk
Web:	http://www.simetrix.co.uk

Contents

Chapter 1

Introduction

1.1 Overview

From version 8, the default file format used with SIMetrix is an ASCII based format. Previous to this, the default file format was a proprietary binary format. The ASCII format has existed within SIMetrix since version 4.9, where there has been the optional ability to save schematics to ASCII along with the ability to read ASCII file format schematics. The file format produced in version 8 is compatible with earlier versions, for schematic features available in those versions. Likewise, schematics in the ASCII format along with similarly aged binary schematics are compatible with version 8.

1.2 Styles Used in this Document

1. All definitions are in a Sans-Serif Font and are indented.
2. Elements that are in a **Bold Sans-Serif** Font represent keywords
3. Elements that are in an *Italic Sans-Serif Font* represent variables
4. Constant values are represented by a Plain Sans-Serif Font.
5. Elements enclosed in square brackets - [ ] - are optional
6. The pipe symbol - | - indicates either-or choices

1.3 Changes from Previous Versions

Changes in Revision 8

Introduced at version 8 of SIMetrix.

- Changed wire definition from primitive object **Wire** to compound object form. The compound form now contains an **Attributes** property with wire position information. All other wire properties now exist as self contained **Property** elements.
- Added style library.
- Added annotation objects for lines, shapes and images.
- Added FilledPoly to symbol definition.
- Added Image to symbol definition.

- Added TitleBlockAnnotation to contain title block information.
- Added **–SXVERSION1.5** and **–EndSXVERSION1.5** to define blocks that are ignored by the file reader in version 8 and beyond.
- Added **binarch** and **system** attribute to **LicenceInfo**

Changes in Revision 7

Introduced at version 7.2 of SIMetrix.

- Added **window** property to schematic.

Changes in Revision 6

Introduced at version 6.00d of SIMetrix.

- **product** attribute added to **LicenceInfo** primitive object.

Changes in Revision 5

Introduced at version 5.50c of SIMetrix.

- **rotated** attribute added to Pin object

Changes in Revision 4

Introduced at version 5.50 of SIMetrix.

- Added **LicenceInfo** primitive object. This provides information about the license in use by the user who saved the schematic. The information is used by our technical support staff to aid problem diagnosis. It has no other purpose and can be ignored by readers.

Changes in Revision 3

Applies to SIMetrix version 5.3 and later.

- Addition of **snapgrid** attribute to ViewObject object.

Changes in Revision 2

- Schematic added SimulatorMode primitive object.
- **Revision** value added to header. This replaces the minor format version.

Chapter 2

Basic Schematic Concepts

2.1 Schematic Elements

Graphically, all SIMetrix schematics contain three types of object. These are:

1. Instances. These reference a symbol that defines its graphical representation and electrical terminations (known as 'pins'). Instances have properties that are used to define its electrical and other characteristics. When an instance is first placed on a schematic, its properties are copied from those defined in the symbol. These properties can subsequently be edited unless they are defined as 'protected' in the symbol.
2. Wires. Wires define and graphically represent the interconnection between instance pins.
3. Annotations. Annotations are graphical elements that do not make up part of the simulated circuit but provide mechanisms for labelling or improving readability of a schematic.

In addition, each schematic has a 'text window' or more commonly the 'F11 window'. This is a simple text edit box in which simulator constructs may be entered. This does not appear as a graphical element, but its contents are stored in the schematic file.

2.2 Symbols and Symbol Libraries

The symbol used by an instance is referenced by its name and is usually obtained from a symbol library. (The exception is for component symbols used in hierarchies). The symbol library can be local or global. The local symbol library is stored within the schematic itself and would usually have definitions for all symbols used within the schematic. The global symbol library consists of a number of files that the user can install and are managed by the Symbol Library Manager.

When a schematic is opened, a situation can arise where a symbol is defined in both the local and global libraries. Indeed this would usually be the case, although the two definitions will usually - but not always - be identical. The procedure used by SIMetrix to resolve the symbol is as follows:

1. Search in global library.
2. If present in global library, examine the 'track' flag. This flag is set if the user checks the "All references to symbol automatically updated" box when saving the symbol in the graphical symbol editor.
3. If track flag is set then use the global symbol.
4. If track flag is not set or the symbol is not found in the global library, search the local library.

5. Use local symbol if present. If not present and a global symbol was found in step 1, then use global symbol. Otherwise raise error condition.

2.3 Coordinate System

All SIMetrix schematics are internally represented in what is known as the “Sheet coordinate system”. The basic unit in this system is the “Sheet unit”. The system uses integer arithmetic throughout and so floating point values are not recognised or output.

Schematics are drawn on a grid with 120 sheet units per grid unit. Currently all instance pins and wire terminations always lie on a grid square if the schematic was created competely within the SIMetrix environment. That is their location in sheet units is always a multiple of 120. However, schematics imported from PSpice “Schematics” may have off-grid elements, so this cannot be assumed.

The origin of the sheet coordinate system is abritrary. That is there is no fixed relationship between the point 0,0 in sheet units and any particular point on the viewed schematic. SIMetrix does not require the user to specify a worksheet as a reference for the design. A worksheet can be placed, but this is just a symbol with a special ‘protected’ attribute that prevents it from being moved.

2.4 Connectivity

The SIMetrix schematic editor never stores connectivity information. SIMetrix generates connectivity information from a topological analysis of the the wires and instance pins and this process is performed as the netlist is generated. All net names are assigned during this process.

Because of this, it is not necessary to save connectivity information. However, SIMetrix does store the names of nets on wires and instance pins but this information is only used by SIMetrix itself for the purposes of cross-probing. As this information is only updated when a netlist is generated, it cannot be guaranteed to be accurate unless a netlist is created immediately before the schematic is saved.

If a reader is using the ASCII file to create another file for import into an alien schematic editor and that schematic editor requires connectivity to be presented, then it is essential that the netlist generator is run before saving the file. If this is done then the netnames object stored in each instance can be used to identify connectivity between instance pins.

2.5 Styles and Style Libraries

Added in version 8, styles define how the instances and annotations appear on the schematic in terms of their colour, line types and font settings. Styles are held within a Style Library, which is conceptually similar to the Symbol Library. Style libraries exist both globally within SIMetrix and locally within a particular schematic, where by default the local style library has precedence over the global library. When a style is requested for use, the local style library is inspected first and if the style does not exist in that library, then the global library is inspected. If neither library has the style, a default style is applied.

To ensure required styles are transported with a schematic, all styles used within the schematic are stored to the schematics local style library, which is then saved with the schematic. When the schematic is opened, by default the local style library will be loaded along with the schematic. In cases where a style defined in a schematic is different to an already defined style in the users global style, the user will be given an option to either use or ignore the incoming local style.

Chapter 3

Basic ASCII File Structure

Currently there are two types of file: one for schematics and one for symbol libraries. However, both follow the same basic structure, as follows:

1. The format consists of one or more objects.
2. There are two types of object namely ‘primitive’ and ‘compound’.
3. Primitive objects consist of a single line in the form: *ObjectType name-value-pairs*

   where *ObjectType* is a single keyword to identify the object type
   name-value-pairs is one or more constructs of the form

   name=value

   where *name* is the name of some attribute and *value* is its value. Note that name value pairs may be in any order.
   value may be quoted if it contains special characters. See below.

   Note that there is no limit to the line length of a primitive object. In the case of the “TextWindow” on page 23 object, the line length could be considerable. File readers should not make any assumption about the maximum length.
4. Compound objects consist of a sequence of one or more objects in the form:

   .ObjectType
   Objects
   .End*ObjectType*

   Objects is a sequence of one or more objects. These may be primitive or compound. Note that the object type is prefixed with a period (‘.’). The object definition is terminated with the object name prefixed with **.End**.
5. The first object in all files is always a “Header” on page 7.

An important feature of this structure is that it allows for object types that have not yet been defined. A file reader should not raise an error if such an object is encountered but skip to the end of the object. It may be appropriate to raise a warning condition in this situation. The reader should also not raise an error if an unrecognised attribute type in a primitive object is detected.

If a new object type is defined, the revision number will be increased. (The revision number is is defined in the header but is otherwise ignored). The format version will only be changed if some aspect of the basic structure is altered in a manner that is not forward compatible.

In most cases the order in which the objects are placed is not important. Where a particular order is required, this will be noted in the relevant section describing the object type. However, SIMetrix will always output objects in the order defined in this document.

3.1 Quoted Values

As mentioned previously, the values in name=value pairs may be quoted to allow for special characters.

If value contains spaces, new lines, tab characters, double quotes (") or backslash (\ ) then it must be quoted using the double quotation character: ". These special characters are represented as described below:

Tab	\t
Newline	\n
Double quote (")	\"
Back slash (\ )	\\

A back slash followed by any character other than those listed above will be ignored.

3.2 Comments

SIMetrix will not currently output any comment text, but the following object names have been reserved for use as comments. These should be ignored without warnings being issued if they are encountered.

Primitive - i.e. single line comment	**Rem** *comment-text*
Compound - i.e. multi line comments	**.Rem** *Comment lines* **.EndRem**

Additionally, the following tags were introduced in version 8 to mark out sections of the schematic file that should not be read in version 8. The **–SXVERSION1.5** tag denotes the start of a block that will be ignored by the file reader from version 8 onwards, with **–EndSXVERSION1.5** ending the block.

3.3 File Types

Schematic Files

Schematics files actually define two elements and together the pair is known as a 'Component'. The two elements are the schematic itself and a symbol to represent it in a hierarchy. Top level and single sheet schematics do not have a symbol, so for these the symbol entry is empty. Schematic files have the following format:

- "Header" on page 7
- "ComponentObject" on page 7

Symbol Library Files

These contain a collection of symbols and may be used to define all or part of a global library. Format is:

- "Header" on page 7
- "SymbolLibrary" on page 16

Chapter 4

Object Types

4.1 Header

This is of the following form:

SIMetrixFile type=*type* **format**=*version* [**revision**=*revision*]

where:

type is always schematic. A reader should reject the file if any other value is encountered.

version is currently 1.0. A reader should reject the file if this value is anything else.

revision should be ignored by file readers. SIMetrix versions 5.2c and earlier do not write a revision value. From 5.2d a revision number of 2 or greater will be written. Each time a new object or primitive attribute is added, the revision number will be increased.

4.2 Component Object

Compound object of the form:

.Component

[SymbolDefinition]

[StyleDefinition]

[SchematicDefinition]

.EndComponent

Chapter 5

Symbol Definition

.Symbol

SymbolAttributes

[Segments]

[ArcSegments]

[FilledPoly]

[Image]

[Pins]

[Properties]

.EndSymbol

The Symbol Attributes must be the first line. The remaining lines may be in any order but the order of the property lines may affect how they are displayed.

5.1 Symbol Attributes

Primitive object of the form:

Attributes name=*name* [**description**=*description*] [**catalog**=*catalog*] [**track**=1]

name	symbol name. This is a unique name to identify the symbol. It may consist of alpha-numeric characters only. This is known as the 'internal name' in the user interface.
description	Description of symbol. This is how the symbol's name is usually presented to the user.
catalog	Semicolon delimited list of names. This defines where the symbol is placed in the Library Manager and Symbol Brower tree list hierarchy.
track=1	This is only operative for symbols defined within a symbol library file. It is set if the symbol is saved with "All references to symbol automatically updated" box checked. It causes the symbol in the global library to take precedence over a symbol of the same name in the schematic's local library. Otherwise the local symbol takes precedence.

5.2 Segments

Primitive object. Defines a straight line segment of a symbol. Single line object of the form:

Segment x1=*x1* **y1**=*y1* **x2**=*x2* **y2**=*y2*

x1	x-coordinate of start of segment
y1	y-coordinate of start of segment
x2	x-coordinate of end of segment
y2	y-coordinate of end of segment

The following are optional arguments and may reduce performance:

styleNormal	Override of the normal style for this particular segment.
styleSelected	Override of the selected style for this particular segment.

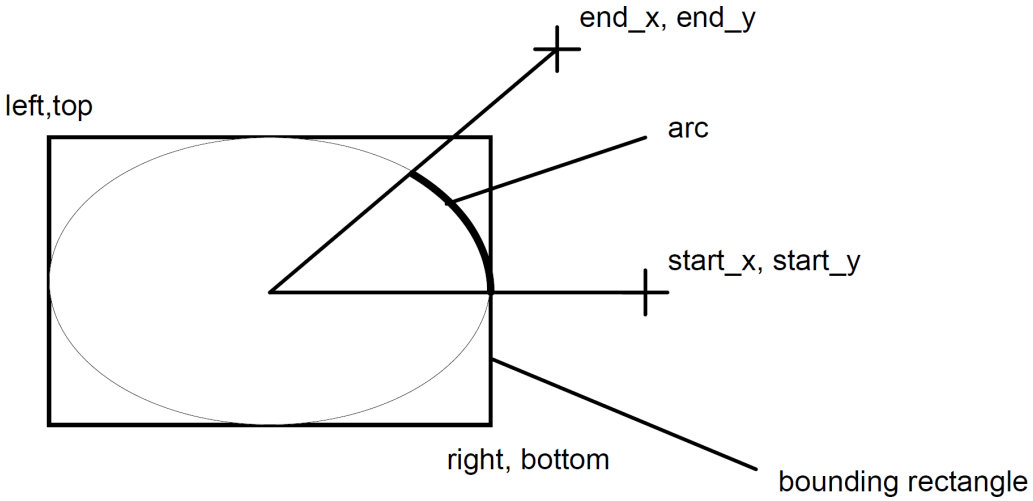
5.3 Arc Segments

Primitive object. These define arcs, circles and ellipses and are of the form:

Arc l=*left* **t**=*top* **r**=*right* **b**=*bottom* **x1**=*x1* **y1**=*y1* **x2**=*x2* **y2**=*y2*

left	x-coordinate of left edge of bounding rectangle
top	y-coordinate of top edge of bounding rectangle
right	x-coordinate of right edge of bounding rectangle
bottom	y-coordinate of bottom edge of bounding rectangle
x1	see diagram
y1	see diagram
x2	see diagram
y2	see diagram

The arc forms part of an ellipse whose bound is described by the bounding rectangle defined by left, top, right and bottom values. This is shown in the following diagram:



The values x1, y1, x2 and y2 define two lines drawn from the centre of the ellipse which specify the start and end of the arc. The intersection of the line drawn from the centre to x1, y1 with the arc defines the start point while the intersection of the line drawn from the centre to x2, y2 defines the end point. The arc is drawn anticlockwise from the start to the end. If x1=x2 and y1=y2, a full ellipse is drawn.

The following are optional arguments and may reduce performance:

styleNormal	Override of the normal style for this particular arc.
styleSelected	Override of the selected style for this particular arc.

5.4 FilledPoly

Primitive object. Defines an enclosing area to fill.

FilledPoly x=*X* **y** = *Y*

X	An array of x coordinates separated by commas, each corresponding to the matching y element in *Y*.
Y	An array of y coordinates separated by commans.

5.5 Image

Primitive object. Defines an image placed within a symbol.

Image l=*left* **t**=*top* **r**=*right* **b**=*bottom* **rot**=*rotate* **mir**=*mirror* **data**=*image*

l	Left coordinate of the image.
t	Top coordinate of the image.
r	Right coordinate of the image.
b	Bottom coordinate of the image.
rot	Rotate flag for the image. Default=0.

mir	Mirror flag for the image. Default=0.
data	Base64 representation of the image to display.

5.6 Pins

Primitive object. Defines a symbol's pin. A pin is a point on the symbol where an electrical connection to a wire or another pin may be made. Pins are defined thus:

Pin name=*name* **order**=*order* **x**=*x* **y**=*y*

[**visible**=1 **xlabel**=*xlabel* **ylabel**=*ylabel* **align**=*align* **rotated**=*rotated*] |

[**visible**=0] [**prefix**=*prefix*] [**suffix**=*suffix*]

name	Pin's name. This may not contain spaces, tabs or new lines.
order	Pin order used to define SPICE netlist position. Pin order values must start at 1 and be contiguous and unique. So if there are 5 pins, the pin order values 1, 2, 3, 4 and 5 must all be present.
x	x-coordinate
y	y-coordinate
visible	If equal to 0, specifies that the pin's name will not be visible on the schematic. Otherwise the pin name will be visible and its location will be defined by xlabel, ylabel and align. Default=1.
xlabel	x-coordinate of pin's label relative to the symbol origin. Ignored if visible=0.
ylabel	y-coordinate of pin's label relative to the symbol origin. Ignored if visible=0.
align	Defines pin label's alignment, i.e what point of the text does xlabel and ylabel define. Ignored if visible=0. May be one of the following. 'Base' means the baseline of the font. LeftTop CentreTop RightTop LeftBase CentreBase RightBase
rotated	If equal to 1, pin name will be rotated by 90 degrees.
prefix	Defines special text that prefixes the pin name in the SPICE netlist. This is used for some SIMetrix digital devices.
suffix	Defines special text that appends the pin name in the SPICE netlist. This is used for some SIMetrix digital devices.

5.7 Properties

Primitive object. Properties define the characteristics of a symbol and are also used for display purposes. All properties have a name, value and a number of attributes. Properties are defined as follows:

Property name=*name* [**value**=*value*] [**autopos**=1|0] [**normal**=*normpos*]

[**rotated**=*rotpos*] [**align**=*align*] [**x**=*x*] [**y**=*y*] [**protected**=0|1] [**visible**=1|0] [**linear**=0|1]

[**selectable**=0|1] [**fixedrotated**=0|1] [**displayName**=0|1] [**resolveSymbolic**=0|1]

name	Property name. May not contain spaces tabs or new lines and must be unique within a symbol.
value	Property's value. May be any text string. Default is an empty string.
autopos	Defines whether property is displayed using the auto-positioning algorithm. The auto-positioning algorithm chooses a location for the text of the property (i.e. its value) according to the value of the **normal** and **rotated** attributes (see below). The default value is 1, i.e. autopos is enabled by default. If autopos is not enabled, the property's location is defined by the **align**, **x** and **y** values.
normal=*normpos*	Can be one of left, right, top or bottom. Defines where the property text is located if **autopos**=1 and if the rotation of the instance is 0 or 180 degrees. The property text will be placed along the specified edge of the symbol's border in a manner that will not overlap with other autopositioned properties. Ignored if **autopos**=0, default=left.
rotated=*rotpos*	Can be one of left, right, top or bottom. Defines where the property text is located if **autopos**=1 and if the rotation of the instance is 90 or 270 degrees. The property text will be placed along the specified edge of the symbol's border in a manner that will not overlap with other auto-positioned properties. Ignored if **autopos**=0. Default=left.
x	x-coordinate of property text relative to origin if **autopos**=0. Ignored if autopos=1. Default=0.
y	y-coordinate of property text relative to origin if **autopos**=0. Ignored if autopos=1. Default=0.
align	Defines the property text alignment, i.e. what part of the text do the x and y values define. Ignored if **autopos**=1. May be one of the following. 'Base' means the baseline of the font. LeftTop CentreTop RightTop LeftBase CentreBase RightBase Default=LeftTop.
protected	Defines whether the property may be edited on an instance of the symbol within the schematic editor. If **protected**=0 then the property may be edited. Otherwise it may not be edited. If a property is protected, the only method of editing it is to edit the symbol itself. Default=0.
visible	Defines whether the property is visible on the schematic. If 0 the property is not visible otherwise it is visible. Note that other attributes that control how a property is displayed are still read in and stored even if **visible**=0. Such attributes will become effective if the user changes the visible status of the property. Default=1.
linear	Defines how the property's font size alters with schematic zoom. If **linear**=0 then the font size is varied in a non-linear fashion in order to maximise readability and aesthetics. Otherwise the font size varies linearly with zoom magnification. Default=0.
selectable	Defines whether the property may be selected individually. Default=0.
fixedrotated	If non-zero and autopos=0, the text of the property will be displayed vertically. Default=0.

displayName If 0, only the property's value will be displayed on the schematic. Otherwise its name will be displayed as well in the form *name*=*value*. Default=0.

Chapter 6

Style Definition

Compound object of the form:

.StyleLibrary

[Style]

.EndStyleLibrary

The Style Library is optional as system styles or default styles can be used if the local style definition. In some cases the user may opt to ignore styles loaded in with a schematic.

Each style defined within the schematic's style library will be added to a style library that is local to that particular schematic. When styles are requested, SIMetrix will first look in the local style file, then the global style file then the use the default style, in cases where a style is not found.

6.1 Style

Primitive object. This object defines a single style that will be added to the local style library.

Style name=*name* **lineColour**=*line-colour* **lineThickness**=*thickness* **penstyle**=*penstyle*

fontFamily=*font-name* **fontSize**=*point-size* **fontColour**=*font-colour* **fontItalics**=*italics*

fontBold=*bold* **fontOverline**=*overline* **fontUnderline**=*underline*

propertyStyle=*property*

name	The name of the style.
line-colour	The colour to apply to the style as an integer value RGB colour definition.
thickness	The thickness of the line. Default is 0, which means hairline and will generally be a single pixel wide.
penstyle	The style of pen to use when drawing lines. The options are: Solid, Dash, Dot, Dashdot, Dashdotdot. The default is Solid.
font-name	The name or family of fonts to use for any text that is displayed. Default is 'arial'.
point-size	The font size.
font-colour	The colour of the font as an integer value RGB colour definition.
italics	Flag to turn italics text on. Default is 0.
bold	Flag to turn bold text on. Default is 0.
overline	Flag to turn overline text on. Default is 0.

underline	Flag to turn underline text on. Default is 0.
property	Flag to state whether this style should be considered for use as a property override style. Default is 0.

Chapter 7

SchematicDefinition

Compound object of the form:

.Schematic

[SymbolLibrary]

[ViewObject]

[Instances]

[Wires]

[ImageAnnotation]

[LineAnnotation]

[ShapeAnnotation]

[TextAnnotation]

[TitleBlockAnnotation]

[TextWindow]

[SimulatorMode]

[LicenseInfo]

.EndSchematic

The above shows the format and order that SIMetrix will write out the ASCII file. However, only the location of the SymbolLibrary object is important - i.e. it must be first. SIMetrix can successfully read in an ASCII format file with the remaining objects in any order.

7.1 SymbolLibrary

Compound object of the form:

.SymbolLibrary

[Symbols]

.EndSymbolLibrary

Symbols consists of zero or more "Symbol objects" on page 8.

7.2 ViewObject

Primitive object. A reader can ignore this object if desired. The x,y location defines a reference point that SIMetrix uses to locate the schematic within the viewable area. The zoom value defines the initial magnification set when the user first opens the schematic.

View x=*x* **y**=*y* **zoom**=*magnification* **snapgrid**=*snapgrid*

x, *y* Sheet coordinate location of the top left corner of the viewable window.

magnification Index value representing zoom level. The index values are defined in the following table:

Index	Pixels per visible grid	Sheet units per pixel	Magnification
2	120	1	10
3	60	2	5
4	40	3	10/3
5	30	4	5/2
6	24	5	2
7	20	6	5/3
8	15	8	4/4
9	12	10	1
10	10	12	5/6
11	16	15	2/3
12	12	20	1/2
13	10	24	5/12
14	20	30	1/3
15	15	40	1/4
16	12	50	1/5
17	10	60	1/6
18	16	75	2/15
19	12	100	1/10
20	10	120	1/12

Note that indexes 0 and 1 are not used with schematics although they are used internally by the symbol editor.

snapgrid Revision 3 and later. Grid that wire ends and instance pins snap to in the editing environment. If omitted, defaults to 120.

7.3 Instances

Compound object. Zero or more Instance objects as defined below:

.Instance

Instance Attributes

[Properties]

[NetNames]

.EndInstance

The Instance Attributes object must be the first object. The remaining objects may be in any order.

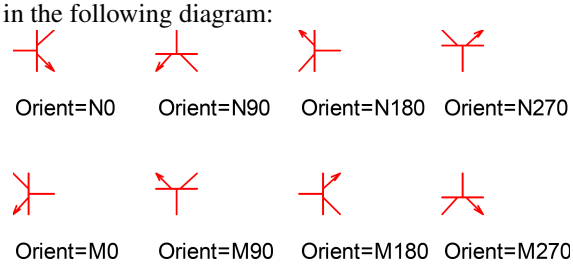
Instance Attributes

Primitive object of the form:

Attributes **type**=symbol|component **name**=*symbol-name*|**path**=*component-path*

[**selected**=0|1] [**protected**=0|1] [**x**=*x*] [**y**=*y*] [**orient**=*orient*]

type	Value is either symbol or component. Specifies whether the instance is a regular symbol or a hierarchical component. Symbol: Search symbol library for symbol specified by the **name** attribute. The symbol will always be present in the local symbol library. Component: Symbol must be read from the component file specified by the **path** attribute.
name	Name of symbol to use for instance when type=symbol.
component	File system path name of file containing symbol for hierarchical component.
selected	If non-zero, the instance will be selected when the schematic is first opened. Otherwise it will not be selected. Default = 0.
protected	If non-zero, the instance will be protected. That is the user will not be able to move it or edit it in any way until it is unprotected. This attribute is used for symbols that are used to decorate the schematic such as worksheets and title boxes.
x	x-coordinate of instance. Default=0.
y	y-coordinate of instance. Default=0.
orient	orientation of instance. This is one of the following codes: N0, N90, N180, N270, M0, M90, M180, M270 N0 means the instance should be displayed how the symbol is defined without rotation or mirroring. The remaining orient values are described in the following diagram:



Properties

Zero or more properties in the identical format described "Symbol Properties" on page 11. Note that only unprotected properties will be listed with the instance definition. Protected properties are read from the symbol definition.

NetNames

This is a list of names assigned to the pins of the instance. These will only be present if a netlist has been generated for the schematic. SIMetrix itself only uses these names for the purposes of cross-probing and are not used to define connectivity. They cannot be relied upon to accurately define connectivity unless a netlist is created for the schematic immediately before the ASCII schematic file is generated. For more information See "Connectivity" on page 4.

Netnames [**pin1**=*name1* [**pin2**=*name2* ...]]

name1, *name2* etc.	Name of net connected to pin 1, 2 etc. The number (1, 2, etc) refers to the pin order value. (See "Pins" on page 11).

7.4 Wires

Wires define the interconnections between instance pins. Note that the locations of junction markers are not stored in the schematic file. Junction markers are small solid squares that denote connection points where more than two wire terminations meet.

In version 8, wires were changed from a primitive object to a compound object and started to include properties. For backwards compatibility, the old primitive forms are included in schematics generated in version 8, but are not read back in for version 8 files.

The *netname* and *branch* attributes in the legacy primitive object are now properties in the compound object.

.Wire

[Attributes]

[Properties]

.EndWire

Attributes

Primitive object. Contains the start and stop position for the wire.

Wire x1=*x1* **y1**=*y1* **x2**=*x2* **y2**=*y2*

x1, *y1*, *x2*, *y2*	Coordinates of wire ends.

Properties

Contains properties for the *netname* and *branch*. Follows the format set out in "Symbol Properties" on page 11.

Legacy Definition

Pre-version 8 wire definition, as a primitive object.

Wire x1=*x1* **y1**=*y1* **x2**=*x2* **y2**=*y2* [**net**=*netname*] [**branch**=*branch*]

x1, *y1*, *x2*, *y2*	Coordinates of wire ends.

netname	Name of net assigned by netlister. This is used by SIMetrix for the purposes of cross-probing and is created when the netlist generator is run. This attribute may be empty or omitted.
branch	Branch formula used to calculate wire current for cross probing. This is derived by netlist generator and may be empty or missing if no netlist has been run.

7.5 ImageAnnotation

Compound object. Defines an image that is placed on the schematic.

.ImageAnnotation

[Attributes]

[Properties]

.EndImageAnnotation

Attributes

Primitive object.

Image l=*left* **t**=*top* **r**=*right* **b**=*bottom* **rot**=*rotate* **mir**=*mirror* **data**=*image*

l	Left coordinate of the image.
t	Top coordinate of the image.
r	Right coordinate of the image.
b	Bottom coordinate of the image.
rot	Rotate flag for the image. Default=0.
mir	Mirror flag for the image. Default=0.
data	Base64 representation of the image to display.

Properties

Follows the format set out in "Symbol Properties" on page 11.

7.6 LineAnnotation

Compound object. Defines a line that is placed on the schematic. Two types of line annotation exist, lines and arrows.

.LineAnnotation

[Attributes]

[Properties]

.EndLineAnnotation

Attributes

Primitive object.

Attributes x1=*x1* **y1**=*y1* **x2**=*x2* **y2**=*y2* **type**=*type*

x1, *y1*, *x2*, *y2*	Coordinates of line ends.
type	Type of line annotation. Either 'line' or 'arrow'. If arrow, the arrow head points to the values given by *x2* and *y2*.

Properties

Follows the format set out in "Symbol Properties" on page 11.

7.7 ShapeAnnotation

Compound object. Defines a shape that is placed on the schematic. Several different shapes exist, these are: rectangle, rhombus, rounded rectangle, ellipse, octagon and triangle.

.ShapeAnnotation

[Attributes]

[Properties]

.EndShapeAnnotation

Attributes

Primitive object.

Image l=*left* **t**=*top* **r**=*right* **b**=*bottom* **shape**=*shape* **orient**=*orientation* **x**=*x* **y**=*y*

l	Left coordinate of the image.
t	Top coordinate of the image.
r	Right coordinate of the image.
b	Bottom coordinate of the image.
shape	The type of shape. Options are: 'rect', 'rhombus', 'roundedrect', 'ellipse', 'octagon', 'triangle'.
orient	The orientation of the shape. See "Instance Attributes" on page 18 for a description of the *orient* attribute.
x, *y*	The x and y location of the shape.

Properties

Follows the format set out in "Symbol Properties" on page 11.

7.8 TextAnnotation

Compound object. Defines a text annotation that is placed on the schematic.

.TextAnnotation

[Attributes]

[TextInfo]

[Properties]

.EndTextAnnotation

Attributes

Primitive object, defines the position and orientation of the text.

Attributes x=*x* **y**=*y* **orient**=*orient*

x, *y*	Coordinates of the text. This is the top left most point.
orient	Orientation of the text. See "Instance Attributes" on page 18 for description of the *orient* attribute.

TextInfo

Primitive object.

TextInfo text=*text* **font**=*font* **size**=*size* **italics**=*italics* **bold**=*bold*

text	Coordinates of the text. This is the top left most point.
font	Font name to use.
size	Size of the font.
italics	Whether make the font italics, values are strings of either: true or false.
bold	Whether to make the font bold, values are strings of either: true of false.

Properties

Follows the format set out in "Symbol Properties" on page 11.

7.9 TitleBlockAnnotation

Compound object. Defines a text annototation that is placed on the schematic.

.TitleBlockAnnotation

[Attributes]

[TextBlockInfo]

[Properties]

.EndTitleBlockAnnotation

Attributes

Primitive object, defines the position and orientation of the title block.

Attributes x=*x* **y**=*y* **orient**=*orient*

x, *y*	Coordinates of the title block. This is the top left most point.
orient	Not used.

TextBlockInfo

Primitive object, defines the content of the title block.

TextBlockInfo company=*company* **title**=*title* **author**=*author* **notes**=*notes*

layout=*layout* **image**=*image* **date**=*date* **version**=*version*

company	Coordinates of the title block. This is the top left most point.
title	Text to appear in the title box.
author	Text to appear in the author box.
notes	Text to appear in the notes box.
layout	Layout flag, either 'horiztonal' or 'vertical'.
image	Base64 representation of the image to appear in the image box.
date	Optional date field. If not set, the value will automatically be taken from the file date.
version	Optional version field. If not set, the value will automatically be taken from file version.

Properties

Follows the format set out in "Symbol Properties" on page 11.

7.10 TextWindow

Primitive object. This object defines the contents of the schematic's "F11 Window".

Text value=*text*

text	The text that appears in the schematic's F11 window.

7.11 SimulatorMode

Primitive object. This object defines the schematic's initial simulator mode. This object was added at revision 2.

SimulatorMode value=SIMPLIS | SIMetrix

7.12 LicenseInfo

Primitive object. This object supplies support information about the user and system. This is not required but we use this information to help process technical support requests.

LicenseInfo version=*version* **feats**=*feature_string* **user**=*licensed_user*

serial=*serial_number* **code**=*license_code* **product**=*product*

version	Version of program
feature_string	Feature names separated by pipe (‘|’) symbols
licensed_user	Licensed user name
serial_number	Serial number of license. This usually starts ‘SX’
license_code	License validation code
product	Product name
binarch	x86 or x64 - indicates whether using a 32 bit version or 64 bit version os the program
system	System information in form osclass/major.minor.sp/arch/memory/physical/processors/logical/hash Where

osclass	Class of operating system. Currently always WINNT
major	OS major version
minor	OS minor version
sp	OS service pack
arch	OS architecture - currently x86 or x64
memory	installed memory in bytes
physical	Number of physical cores
processors	Number of processor chips
logical	Number of logical cores
hash	A code unique for each user/system combination

SIMetrix 8.0 User Manual