



# TECHNICAL REPORT

EIA-EDIF-AG-1™

Issued 1989-07  
Stabilized 2015-07

Using EDIF 200 for Schematic Transfer

## NOTICE

This document has been taken directly from the original TechAmerica document and contains only minor editorial and format changes required to bring it into conformance with the publishing requirements of SAE Technical Standards. The release of this document is intended to replace the original with the SAE International document. Any numbers established by the original document remain unchanged.

The original document was adopted as an SAE publication under the provisions of the SAE Technical Standards Board (TSB) Rules and Regulations (TSB 001) pertaining to accelerated adoption of specifications and standards. TSB rules provide for (a) the publication of portions of unrevised specifications and standards without consensus voting at the SAE committee level, and (b) the use of the existing specification or standard format.

SAE Technical Standards Board Rules provide that: "This report is published by SAE to advance the state of technical and engineering sciences. The use of this report is entirely voluntary, and its applicability and suitability for any particular use, including any patent infringement arising therefrom, is the sole responsibility of the user."

SAE reviews each technical report at least every five years at which time it may be revised, reaffirmed, stabilized, or cancelled. SAE invites your written comments and suggestions.

Copyright © 2016 SAE International

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior written permission of SAE.

**TO PLACE A DOCUMENT ORDER:** Tel: 877-606-7323 (inside USA and Canada)  
Tel: +1 724-776-4970 (outside USA)  
Fax: 724-776-0790  
Email: CustomerService@sae.org  
<http://www.sae.org>

**SAE WEB ADDRESS:**

Full document details: [SAE EIAEDIFAG1](#)

SAE values your input. To provide feedback on this Technical Report, please visit <http://www.sae.org/technical/standards/EIAEDIFAG1>

Documents

Document Sharing Hub

# EIA PUBLICATION

## APPLICATIONS GUIDE

---

### USING EDIF 2 0 0 For Schematic Transfer

---

EDIF Schematic Technical Subcommittee

---

ELECTRONIC INDUSTRIES ASSOCIATION  
ENGINEERING DEPARTMENT



JULY 1989

## NOTICE

EIA Engineering Standards and Publications are designed to serve the public interest through eliminating misunderstandings between manufacturers and purchasers, facilitating interchangeability and improvement of products, and assisting the purchaser in selecting and obtaining with minimum delay the proper product for his particular need. Existence of such Standards and Publication shall not in any respect preclude any member or non-member of EIA from manufacturing or selling products not conforming to such Standards and Publications, nor shall the existence of such Standards and Publications preclude their voluntary use by those other than EIA members, whether the standard is to be used either domestically or internationally.

Recommended Standards and Publications are adopted by EIA without regard to whether or not their adoption may involve patents on articles, materials, or processes. By such action, EIA/TIA does not assume any liability to any patent owner, nor does it assume any obligation whatsoever to parties adopting the Recommended Standard or Publication.

EDIF/P-1 is an EIA Publication, produced by the EDIF Schematic Technical Subcommittee. Technical Publications are distinguished from EIA Recommended Standards or Interim Standards, in that they contain a compilation of engineering data or information useful to the technical community, and represent approaches to good engineering practices that are suggested by the formulating committee.

This Publication is not intended to preclude or discourage other approaches that similarly represent good engineering practice, or that may be acceptable to, or have been accepted by, appropriate bodies. Parties who wish to bring other approaches to the attention of the formulating committee to be considered for inclusion in future revisions of this Publication are encouraged to do so. It is the intention of the formulating committee to revise and update this Publication from time to time as may be occasioned by changes in technology, industry practice, or government regulations, or for other appropriate reasons.

COPYRIGHT 1989

Published by

**ELECTRONIC INDUSTRIES ASSOCIATION**

Engineering Department

2001 Pennsylvania Ave., N.W.

Washington, D.C. 20006

(Temporary Headquarters 1722 Eye St., N.W. Washington, D.C. 20006)

PRICE: \$15.00

Printed in U.S.A.

All rights reserved

## CAUTION

The recommendations contained in this document are intended to provide temporary conventions for certain EDIF applications. These recommendations are only applicable to EDIF Version 2 0 0. They are not an amendment to the official EDIF standard, nor should they should be expected to be relevant to subsequent versions. Efforts will be made to incorporate these recommendations into the next formal version of EDIF.

This document has been reviewed by the EDIF Technical Committee for content and consistency. It is considered to be of value to EDIF users, subject to the above caution and the mutual agreement of the parties involved in a data transfer.

## ACKNOWLEDGMENTS

Support for production of this document was furnished in part by ACM SIGDA.

## About This Document

This document offers some conventions for the use of EDIF in transfer applications involving schematic designs. It discusses the use of EDIF 2 0 0 constructs that apply to the SCHEMATIC view.

This document does not attempt to cover the use of EDIF for the transfer of all design aspects. It is intended to be a starting point for implementing EDIF schematic translators. It is not an EDIF user manual. It is not an amendment to the *EDIF 2 0 0 Recommended Standard ANSI/EIA-548*, which contains the official definition of the format and should always be taken as the authoritative source of EDIF information.

This document assumes you are familiar with Schematic capture systems but not familiar with EDIF. You will need to refer to EIA-548 for detailed definitions of some of the constructs discussed below.

## Document Organization

Section 1, **Overview**, provides a general description of what the EDIF SCHEMATIC view is, what it is used for, how it differs from other EDIF views, and the EDIF levels relevant to its application.

Section 2, **The EDIF File**, introduces the format and constructs of an EDIF file at the top level.

Section 3, **The Cell Library**, introduces the two types of cell libraries and their technology information.

Section 4, **Cell and View**, describes EDIF *cell* and *view* constructs.

Section 5, **Symbol Representation**, describes the constructs used to create a symbol and represent the interface of a cell view.

Section 6, **Schematic Representation**, describes the constructs used to implement the schematic sheets within the contents of a cell view.

Appendix A, **Recommendations**, is a list of Schematic Technical Subcommittee recommendations found in this document.

Appendix B, **Complete Schematic Example**, includes a complete schematic example and pictures of a hierarchical design.

## Notational Conventions

All constructs are *italicized*.

All symbolic constants are in SMALL CAPS.

All examples are in **bold**.

All Schematic Technical Subcommittee recommendations are marked with \* and enumerated in Appendix A.

Most of the constructs described in this document are presented in the following format:

---

```
(edifedif
  edifFileNameDef
  edifVersion
  edifLevel
  keywordMap
  { < status > |
  external |
  library |
  design |
  comment |
  userData }
)edif
```

---

This format is the one described in the *EDIF 2 0 0* and shows those constructs that are typically used in the Schematic view. Refer to the *EDIF 2 0 0 Recommended Standard ANSI/EIA-548* for a description of this notation.

## Related Publications

The following EDIF publications contain important information related to this document:

*EDIF Version 2 0 0 Recommended Standard ANSI/EIA-548-1988*, Electronic Industries Association, 1988. This publication is the authoritative reference on the use and format of all EDIF constructs.

*Introduction To EDIF*, Electronic Industries Association, 1989. This publication is an overview of basic EDIF concepts.

*EDIF Connectivity*, Electronic Industries Association, to be published. This publication is an overview of EDIF connectivity concepts.



## TABLE OF CONTENTS

## Section 1

|                                                                 |     |
|-----------------------------------------------------------------|-----|
| Overview .....                                                  | 1-1 |
| What is the Schematic View? .....                               | 1-1 |
| What is the Schematic View Used For? .....                      | 1-1 |
| How Does the Schematic View Differ From Other EDIF Views? ..... | 1-2 |
| EDIF Levels Relevant to the Schematic View .....                | 1-2 |

## Section 2

|                     |     |
|---------------------|-----|
| The EDIF File ..... | 2-1 |
|---------------------|-----|

## Section 3

|                              |     |
|------------------------------|-----|
| The Cell Library .....       | 3-1 |
| Technology Information ..... | 3-3 |
| Number Scaling .....         | 3-3 |
| Display Attributes .....     | 3-4 |

## Section 4

|                     |     |
|---------------------|-----|
| Cell and View ..... | 4-1 |
|---------------------|-----|

## Section 5

|                                |      |
|--------------------------------|------|
| Symbol Representation .....    | 5-1  |
| The Cell Interface .....       | 5-2  |
| The Symbol Body .....          | 5-3  |
| Symbol Ports .....             | 5-5  |
| Symbol Port Bundles .....      | 5-6  |
| Port Implementations .....     | 5-8  |
| The Reference Designator ..... | 5-13 |
| Port Attributes .....          | 5-15 |
| The Port Direction .....       | 5-15 |
| The Pin Number .....           | 5-15 |
| Port Delays .....              | 5-16 |

## TABLE OF CONTENTS

## Section 1

|                                                                 |     |
|-----------------------------------------------------------------|-----|
| Overview .....                                                  | 1-1 |
| What is the Schematic View? .....                               | 1-1 |
| What is the Schematic View Used For? .....                      | 1-1 |
| How Does the Schematic View Differ From Other EDIF Views? ..... | 1-2 |
| EDIF Levels Relevant to the Schematic View .....                | 1-2 |

## Section 2

|                     |     |
|---------------------|-----|
| The EDIF File ..... | 2-1 |
|---------------------|-----|

## Section 3

|                              |     |
|------------------------------|-----|
| The Cell Library .....       | 3-1 |
| Technology Information ..... | 3-3 |
| Number Scaling .....         | 3-3 |
| Display Attributes .....     | 3-4 |

## Section 4

|                     |     |
|---------------------|-----|
| Cell and View ..... | 4-1 |
|---------------------|-----|

## Section 5

|                                |      |
|--------------------------------|------|
| Symbol Representation .....    | 5-1  |
| The Cell Interface .....       | 5-2  |
| The Symbol Body .....          | 5-3  |
| Symbol Ports .....             | 5-5  |
| Symbol Port Bundles .....      | 5-6  |
| Port Implementations .....     | 5-8  |
| The Reference Designator ..... | 5-13 |
| Port Attributes .....          | 5-15 |
| The Port Direction .....       | 5-15 |
| The Pin Number .....           | 5-15 |
| Port Delays .....              | 5-16 |

## TABLE OF CONTENTS [continued]

## Section 5 Symbol Representation [continued]

|                                                      |      |
|------------------------------------------------------|------|
| Fanin and Fanout .....                               | 5-18 |
| Capacitive Load .....                                | 5-19 |
| Displaying Names, Attributes and Properties .....    | 5-20 |
| Displaying Names .....                               | 5-20 |
| Displaying Attribute Values .....                    | 5-20 |
| Displaying Property Values .....                     | 5-21 |
| Symbol Graphics .....                                | 5-23 |
| Comment Text and Graphics .....                      | 5-24 |
| Port Relationships .....                             | 5-27 |
| Permutability .....                                  | 5-27 |
| Connectivity (Joined, WeakJoined and MustJoin) ..... | 5-28 |

## Section 6

|                                |      |
|--------------------------------|------|
| Schematic Representation ..... | 6-1  |
| Schematic Pages .....          | 6-3  |
| Instances .....                | 6-3  |
| Port Instances .....           | 6-7  |
| Nets .....                     | 6-9  |
| Busses .....                   | 6-11 |
| Net Bundles .....              | 6-13 |
| Tie Cells and Subnets .....    | 6-18 |
| Ripper Cells .....             | 6-20 |
| Offpage Connectors .....       | 6-31 |
| Physical Connectors .....      | 6-32 |
| Port Implementations .....     | 6-32 |
| Graphic View .....             | 6-33 |
| Caveat .....                   | 6-33 |

## Appendix A

|                       |     |
|-----------------------|-----|
| Recommendations ..... | A-1 |
|-----------------------|-----|

## Appendix B

|                                  |     |
|----------------------------------|-----|
| Complete Schematic Example ..... | B-1 |
|----------------------------------|-----|

## LIST OF FIGURES

|                                                |      |
|------------------------------------------------|------|
| 5-1. Symbol Representation                     | 5-1  |
| 5-2. Multiple Port Implementations             | 5-10 |
| 5-3. A Single Port Implementation              | 5-12 |
| 5-4. Comment Text and Graphics                 | 5-25 |
| 5-5. Port Permutability                        | 5-28 |
| 6-1. Schematic Representation                  | 6-1  |
| 6-2. Instance Orientation                      | 6-5  |
| 6-3. Net Example                               | 6-10 |
| 6-4. Conceptual: Bus                           | 6-12 |
| 6-5. Simple Bus                                | 6-12 |
| 6-6. Net Bundle Example                        | 6-18 |
| 6-7. Tie Cell and Subnets                      | 6-20 |
| 6-8. One-dimensional Ripper                    | 6-21 |
| 6-9. Conceptual: One/Two-dimensional Ripper    | 6-22 |
| 6-10. Two-dimensional Ripper                   | 6-23 |
| 6-11. Multi-dimensional Ripper                 | 6-24 |
| 6-12. Conceptual: Multi-dimensional Ripper     | 6-25 |
| 6-13. Ripper Port Connections                  | 6-26 |
| 6-14. Bus and Ripper Example                   | 6-30 |
| 6-15. Conceptual: Offpage/Interface Connectors | 6-32 |
| B-1. Cell Tree                                 | B-1  |
| B-2. \$inv Symbol                              | B-2  |
| B-3. \$nand2 Symbol                            | B-2  |
| B-4. Hierarchical Level 3 Symbol               | B-3  |
| B-5. Hierarchical Level 3 Schematic            | B-3  |
| B-6. Hierarchical Level 2 Symbol               | B-4  |
| B-7. Hierarchical Level 2 Schematic            | B-4  |
| B-8. Top Schematic                             | B-5  |

## Overview

### What is the Schematic View?

The SCHEMATIC view is an abstract graphical description of an electronic design. The focus of schematics is on virtual components and the electrical networks which interconnect them. The components used in the schematic are virtual in the sense that they may differ from the physical components they represent: a schematic nand-gate is typically a single logic-gate, rather than one gate in a quad-pack (such as a 74LS00).

The SCHEMATIC view describes logic symbols and the schematics which use such symbols. This is achieved by describing graphic and connectivity information at varying levels of detail and composition. The SCHEMATIC view is semantically a superset of the NETLIST view, since all the connectivity information of a design can be specified using the SCHEMATIC view.

### What is the Schematic View Used For?

The EDIF SCHEMATIC view is used to transfer and archive symbol libraries and block, gate, and transistor level schematics. This information is usually transferred between dissimilar schematic capture systems. The SCHEMATIC view can be specifically used to:

- Transfer and distribute symbol libraries and/or schematics between two or more dissimilar Electronic Design Automation (EDA) systems.
- Transfer symbol libraries from a semiconductor vendor to an EDA system.
- Archive symbol libraries and/or schematics in a vendor independent format.
- Modify and extract information from EDIF files that contain symbol libraries and/or schematics. (For example, a program may be used to obtain a netlist directly from an archived schematic.)
- Transfer symbol libraries and/or schematics from a schematic capture system to a layout system (PCB or IC) that has schematic editing capabilities.

Netlists derived from the SCHEMATIC view can be simulated to verify the intentions of the designer or matched with the physical implementation of the design to verify correct design implementation. Because the SCHEMATIC view is designed to express the logical structure of a design, it is typically the designer's window into the design and consequently a definition of the design itself.

## How Does the Schematic View Differ From Other EDIF Views?

Unlike the NETLIST view, the SCHEMATIC view uses graphics. Unlike the graphics in PCBLAYOUT and MASKLAYOUT views, SCHEMATIC view graphics are abstract: the coordinates found in a SCHEMATIC view are not intended to convey fabrication information.

## EDIF Levels Relevant to the Schematic View

In addition to different views of a design, EDIF supports three different levels of complexity which indicate whether a design contains basic, parameterized, or procedural constructs.

This document only discusses EDIF Level 0. The use of parameterized designs expressed in EDIF Levels 1 and 2, while applicable to the design transfer problem, will not be discussed.

## The EDIF File

An EDIF description, or *edif*, may contain several designs and libraries of cell definitions, all contained within a single logical and physical file. The constructs *edifFileNameDef*, *edifVersion*, *edifLevel* and *keywordMap* contain the information required to process an *edif*.

At the top level, the file has the following format:

---

```

(edif
    edifFileNameDef
    edifVersion
    edifLevel
    keywordMap
    { < status > |
    external |
    library |
    design |
    comment |
    userData }
)

```

---

*edif*

*EdifFileNameDef* is used to identify the entire EDIF file.

*EdifVersion* indicates the version of EDIF used in the file. This document describes EDIF Version 2 0 0 constructs.

*EdifLevel* specifies the highest EDIF Level used in the file. EDIF Level 0 files contain only the basic constructs. EDIF Level 1 files contain constants, parameters, variables and/or expressions in addition to the basic constructs. EDIF Level 2 files contain procedural constructs in addition to constructs allowed in the Level 1 files.

---

```

(edifLevel
    integerToken
)

```

---

*edifLevel*

This document only discusses EDIF Level 0.

*KeywordMap* is used to specify all keyword mappings used in the file. Four levels of mapping may be specified. Keyword Level 0 indicates no mappings; the file is native EDIF. Keyword Level 1 may specify keyword aliases. Keyword Level 2 may specify keyword macros which do not use iterative parameter binding. Finally, keyword Level 3 may specify keyword macros which use iterative parameter binding.

---

```
(keywordMap
  keywordlevel
  { comment }
)
```

*keywordMap**keywordMap*

---

This document only discusses keyword Level 0.

*Status* provides accounting information about the design data in an EDIF file. It may be used to describe the time the data was created, the program and its version that generated the data, and the author and origin of the data. Several *written* statements may be used to record a history of modifications to the data.

---

```
(status
  { written |
    comment |
    userData }
)
```

*status**status*

---

*TimeStamp* appears in the *written* section of *status*. Date and time are expressed as integers, Universal Time Coordinated (UTC).

---

```

(written timeStamp
  { < author > |
    < program > |
    < dataOrigin > |
    property |
    comment |
    userData }
)

```

written

written

---

The following example shows a Version 2 0 0, Level 0 EDIF file named **simpleExample**.

```

(edif simpleExample
  (edifVersion 2 0 0)
  (edifLevel 0)
  (keywordMap (keywordLevel 0))
  (status
    (written
      (timestamp 1988 12 18 11 0 0)
      (author "Schematic Technical Subcommittee")
      (program "Schematic Writer" (Version "2.4"))
      (dataOrigin "Berkeley")))
  (library ...))

```

This example file contains only native keywords (no keyword aliases or macros). The file was created on December 18, 1988 at 11:00:00 UTC using a program called **Schematic Writer, version 2.4**. The data originated at **Berkeley**.

*Status* may be used within *edif*, *library*, *external*, *cell*, *view*, and *design*. When used within *edif*, the information contained in the *status timeStamp* refers to the time the EDIF file was created. When used within the other constructs listed above, *timeStamp* refers to the actual design data.\* This use of *status* is useful for data management, allowing for the proper synchronization of the information contained in multiple EDIF files.

The other subconstructs in the *written* construct, *author*, *program* and *dataOrigin*, are used to define a set of strings which describe the person who wrote the file, the name and version of the program used to create the data, and the location where the data was created and the local version number of the data.

---

\*STSC Recommendation (1)

---

```
(design designNameDef                                     design
  cellRef
  { < status > |
  property
  comment |
  userData }
)
```

---

*design*

*Design* identifies the cell at the top level of the hierarchy of a particular design within a library.

## The Cell Library

An EDIF library is a grouping of *cells* used as design modules. A library contains cell definitions and the information common to all the cells in the library. This information includes the fabrication technology description or electrical characteristics. *Library* is defined in the following format:

---

```
(library libraryNameDef
  edifLevel
  technology
  { < status > |
  cell |
  comment |
  userData }
)
```

*library*

---

*library*

*External* specifies an external library, which is referenced within the EDIF file but has definitions that are external to the EDIF file (for example, graphics and attributes). External libraries allow a user to transfer a set of cells once, such as a library of parts, and afterwards send the schematics that use such cells. This type of transfer eliminates the problem of creating multiple copies of the parts library in the target system.

---

```
(external libraryNameDef
  edifLevel
  technology
  { < status > |
  cell |
  comment |
  userData }
)
```

*external*

---

*external*

The *cells* declared in an *external* must contain declarations for all objects to be referenced by other *views* in the EDIF file. Therefore, the *interface* section of the *view* must contain the constructs that define objects (*port*, *portBundle*).

The following example of an external library, **publicLib**, contains a simple cell description of cell OR. Cell OR contains the declaration of three ports, **i1**, **i2** and **o**.

```
(external publicLib
(edifLevel 0)
(technology
.....
)
(cell OR
(cellType Generic)
(view Graphic_Schematic (viewType Schematic)
(interface
(port i1)
(port i2)
(port o)
)
)
)
)
```

## Technology Information

A *technology* contains the number scaling and default figure display attributes used for every cell in the library.

---

```

(technology numberDefinition
  { figureGroup |
    <simulationInfo> |
    fabricate |
    <physicalDesignRule> |
    comment |
    userData }
)

```

*technology*

---

*technology*

---

*Fabricate* and *physicalDesignRule* are not discussed in this document.

## Number Scaling

*NumberDefinition* is used to collect all scaling information pertinent to a *library*. *Scale* defines the relationship between the EDIF units and System International (SI) units, (i.e. meters and kilograms).

---

```

(numberDefinition
  { scale |
    <gridMap> |
    comment }
)

```

*numberDefinition*

---

*numberDefinition*

---

*GridMap* is not discussed in this document.

Implementors of the SCHEMATIC view can use, but are not restricted to, the units for distance, time, and capacitance. The respective *unitType* constants and SI units for these examples are DISTANCE (meters), TIME (seconds) and CAPACITANCE (farads).

The following example describes the distance scaling as 1 EDIF unit equivalent to  $254 * 10^{-6}$  meters, or one hundredth of an inch.

```
(technology
  (numberDefinition
    (scale 1 (e 254 -6) (unit DISTANCE))
  )
)
```

## Display Attributes

*FigureGroup* is used to define default display attribute values to be used later by figures throughout the library. Example default values are color, line width, text height and fill pattern. Several *figureGroups* may be defined and referenced later, and overridden individually.

---

```
(figureGroup figureGroupNameDef
  { < cornerType > |
    < endType > |
    < pathWidth > |
    < borderWidth > |
    < color > |
    < fillPattern > |
    < borderPattern > |
    < textHeight > |
    < visible > |
    includeFigureGroup |
    property |
    comment |
    userData }
)
figureGroup
```

---

```
figureGroup
```

There is no minimum set of attributes required in a *figureGroup*. This means that an EDIF reader will need to have a set of default values for the attributes listed above. On some systems, *figureGroup* attributes are predefined by the system and cannot be changed as specified in the *figureGroup*.

The following example defines a set of figure group attributes.

```
(technology
  (numberDefinition
    (scale 1 (e 254 -6) (unit Distance))
  )
  (figureGroup default_figure_group
    (pathWidth 0)
    (borderWidth 1)
    (color 100 100 100)
    (fillPattern 1 1 (boolean (boolean (FALSE))))
    (borderPattern 1 1 (boolean (boolean (TRUE))))
    (textHeight 20)
  ))
```

## Cell and View

---

```

(cell cellNameDef
  cellType
  { < status > |
  view |
  < viewMap > |
  property |
  comment |
  userData }
)

```

*cell*

---

*cell*

*Cell* is used to define a cell that may be instantiated later in another cell to build a design hierarchy. It holds all the information that pertains to a specific cell.

Cell instances may occur in any order in the contents of the views of other cells, but only if the referenced *cells* were defined earlier in the file or if they belong to external libraries and have already been declared as such in an *external*. View names must be unique within a cell, even if they are of different view types.

---

```

(view viewNameDef
  viewType
  interface
  { < status > |
  < contents > |
  property |
  comment |
  userData }
)

```

*view*

---

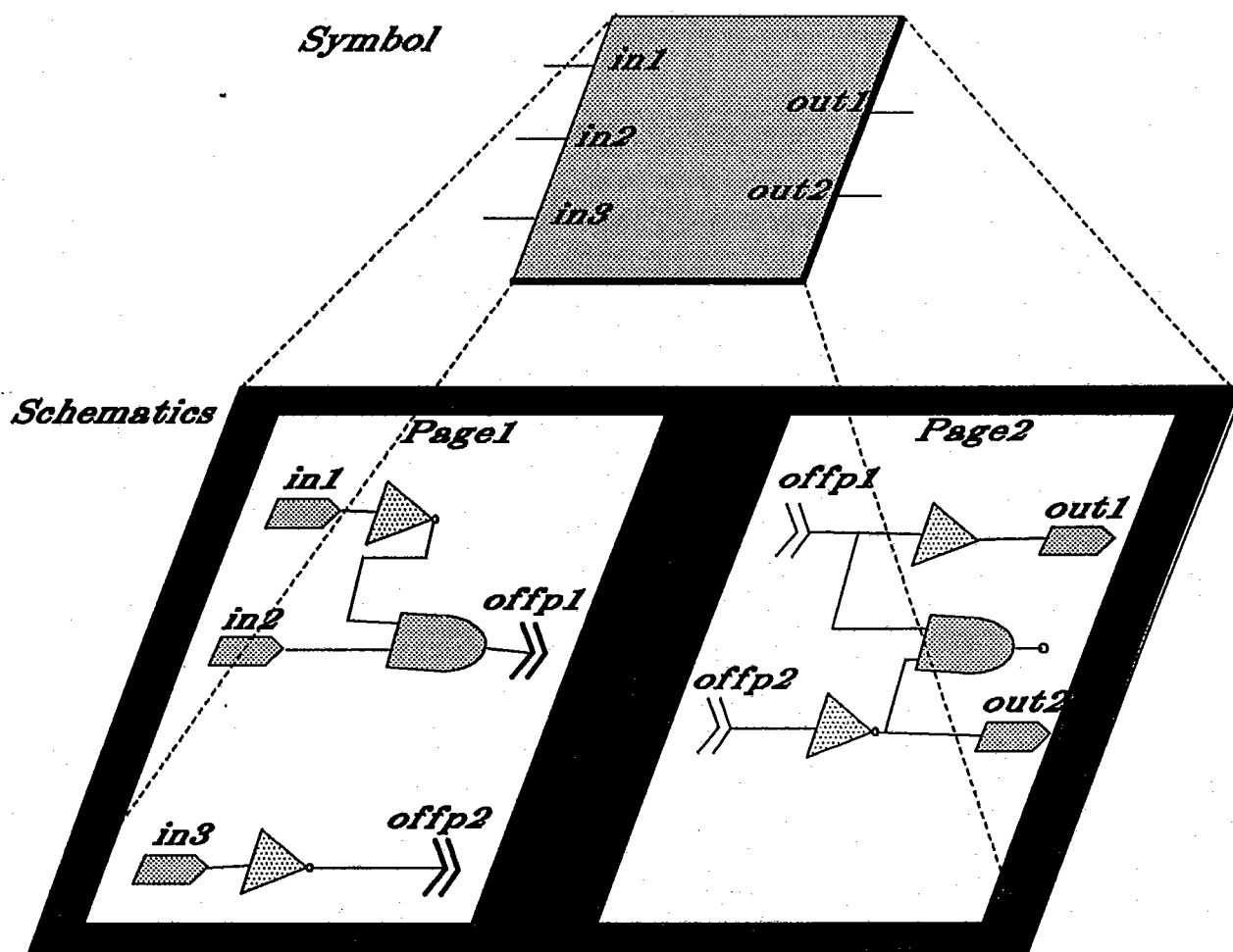
*view*

*View* is used to specify a data representation or perspective of a *cell*. The representation is classified by the *viewType*.

*ViewType* indicates the intended use of the view. It also places restrictions on the constructs appropriate within the contents of such a view. This document describes many of the constructs allowed when a view is classified as *viewType* SCHEMATIC.

## Symbol Representation

In this section we will describe the abstract representation of a cell view. In schematic capture systems, this abstraction is represented by a symbol, and includes the symbol's ports as well as properties and attributes associated with the symbol's ports and body.



• Figure 5-1 Symbol Representation

## The Cell Interface

Symbols are defined within the interface section of the SCHEMATIC view. The interface is used to express information about a cell representation which is provided to an instance. This information includes the port definitions and the symbol graphics which describe the ports (by means of *portImplementation*), and any other graphics that make up a schematic symbol.

---

```
(interface
  { port |
    portBundle |
    permutable |
    < designator > |
    joined |
    weakJoined |
    mustJoin |
    < symbol > |
    property |
    parameter |
    timing |
    simulate |
    < protectionFrame >
    < arrayRelatedInfo >
    comment |
    userData }
)
```

*interface*

---

*interface*

---

*Parameter*, *timing*, and *simulate* are not discussed in this document. *ProtectionFrame* and *arrayRelatedInfo* are not allowed in the SCHEMATIC view.

The constructs contained in the interface can be categorized into five types: constructs that define ports (*port* and *portBundle*), constructs that describe relationships between ports (*permutable*, *joined*, *weakJoined* and *mustJoin*), constructs that are attributes of the cell (*designator*), the *symbol* construct, which is the implementation of the symbol, and the *property* construct, which defines the properties of the cell.

## The Symbol Body

*Symbol* is used to describe the abstract or schematic representation of a cell view. It is commonly referred to as a schematic logic symbol. In a schematic capture system, *symbol* would be used to represent the set of figures that construct the schematic symbol in a schematic view.

Note that a coordinate space exists only within *symbol* and nowhere else in an *interface* of a Schematic view. All coordinates within *symbol* are relative to the symbol's origin.

---

```
(symbol
  { portImplementation |
    figure |
    annotate |
    instance |
    commentGraphics |
    < pageSize > |
    < boundingBox > |
    keywordDisplay |
    parameterDisplay |
    property |
    propertyDisplay |
    comment |
    userData }
)
```

*symbol*

*symbol*

---

Comment graphics and text can be associated with a symbol by using the *commentGraphics* construct. However, the annotative text and graphics in *commentGraphics* are not used as the graphic representation of the symbol. *Annotate* is used to display text that is preserved on instantiation of the symbol. The symbol graphics in schematic views may also be implemented as instances of GRAPHIC views. Additional information on comment text and graphics can be found on page 5-24.

Attributes and properties that are associated with cells are defined in the interface of a view. If such attributes are to be displayed within the coordinate space of the symbol then *keywordDisplay* and *propertyDisplay* are used.

*KeywordDisplay* is used to display such attributes as the reference designator defined by *designator* in the interface. *PropertyDisplay* is used to display user

properties that are defined in the interface. *KeywordDisplay* and *propertyDisplay* are described on page 5–20.

The extent of the symbol page is indicated by the *pageSize* in *symbol*. *PageSize* is useful for indicating a page size for an editor. The extent of the page must cover the design information, but in many systems it will be larger than the design. *PageSize* allows for a margin around graphic data.

---

|                   |                 |
|-------------------|-----------------|
| ( <i>pageSize</i> | <i>pageSize</i> |
| <i>rectangle</i>  |                 |
| )                 |                 |
|                   | <i>pageSize</i> |

---

The *boundingBox* in *symbol* gives the extent of the symbol for use by systems that require this information for display or analysis purposes. The extent of the symbol covers all figures used to implement the symbol, ports and text origins, but does not include the text strings since EDIF cannot at present specify text width

---

|                      |                    |
|----------------------|--------------------|
| ( <i>boundingBox</i> | <i>boundingBox</i> |
| <i>rectangle</i>     |                    |
| )                    |                    |
|                      | <i>boundingBox</i> |

---

## Symbol Ports

*Port* is used to define the ports of the current cell. A port may be a single bit wide or several bits wide (as in an arrayed or bus port). It has a name, defined with an *identifier* or an *array* (for arrayed ports). A port may also have numerous attributes associated with it, as shown below.

---

```
(port portNameDef
  { < direction > |
    < designator> |
    < unused> |
    portDelay |
    < dcFaninLoad > |
    < dcFanoutLoad > |
    < dcMaxFanin > |
    < dcMaxFanout > |
    < acLoad > |
    property |
    comment |
    userData }
)
```

---

*port**port*

## Symbol Port Bundles

*PortBundle* is used to collect ports, arrayed ports and port bundles into a named group. *PortBundle* differs from arrayed ports in that the name of each signal of the port is different, as opposed to being the same name with a different index. In contrast to ports, *portBundles* have no attributes associated with them. They allow ports with different attributes to be collected together and given a single name.

---

```
(portBundle portNameDef
  listOfPorts
  { property |
    comment |
    userData }
)
```

*portBundle*

---

*portBundle*

---

The following is an example of a port bundle **abc** which consists of ports **a**, **b** and **c**.

```
(portBundle abc
  (listOfPorts
    (port a)
    (portBundle b
      (listOfPorts
        (port x)
        (port y)))
    (port (array c 4))))
```

The "size" of a *portBundle* can be determined by adding up the sizes of each nested port and port bundle. In the previous example, port **a** is 1 bit wide. *PortBundle b*, which consists of ports **x** and **y**, is 2 bits wide. Port **c**, an arrayed port, is 4 bits wide. Therefore, the size of *portBundle abc* is 7 bits.

The *portBundle* can be referenced as a whole or as individual elements throughout the EDIF file by using the nested *portRef*. The following examples show the use of *portRef* to reference port members, ports and port bundles.

- (portRef abc) -- this references *portBundle abc*.
- (portRef b (portRef abc)) -- this references *portBundle b* within *portBundle abc*.

- **(portRef x (portRef b (portRef abc)))** -- this references port **x** within *portBundle b*, within *portBundle abc*.
- **(portRef c (portRef abc))** -- this references arrayed port **c**.
- **(portRef (member c 2) (portRef abc))** -- this references the third member of arrayed port **c**, within *portBundle abc*.

## Port Implementations

*PortImplementation*, when used within the *symbol* section of an *interface*, describes the graphics of the ports previously defined with *port*.

---

```
(portImplementation
```

```
  portNameRef | portRef
  { < connectLocation > |
  figure |
  instance |
  commentGraphics |
  keywordDisplay |
  property |
  propertyDisplay |
  comment |
  userData }
)
```

*portImplementation*

---

*portImplementation*

In symbols, *portImplementation* is generally used for each port. However, its use is optional; it may not be appropriate when the ports are "implied" where a default connection is made by the system instead of the user. Implied ports are signals which are not specifically implemented on symbols, but which are connected (defined) throughout the design hierarchy. Power and ground signals are often implemented in such a manner. Implied ports must be defined by *port* and are not implemented within the *symbol* (so that no graphical connections can be made to them). In some cases implied ports could be implemented in a *page* of the *contents*, making the ports explicit.

While power and ground ports are usually implicit within systems, they must be explicitly declared in EDIF. All power and ground signals within a cell must be explicitly connected by means of the *net* construct. A *cell* which bridges the abstraction from transistor level to logic level has power and ground *ports* in its *interface*, *portImplementations* of power and ground in its *contents*, but no *portImplementations* of power and ground in its *symbol*. In this way, instantiators cannot (graphically) connect power and ground signals to the cell's symbol. While it is required to declare the ports, it is not required to implement them graphically.

The *connectLocation* of the port is the area where connections can be made to a port when the port is instantiated with the rest of the symbol in a schematic page. It is recommended that the *connectLocation* always be specified. It should contain at most

one figure, preferably a *dot*, since some systems can only connect to a single point (usually the end point of a port).\*

In some cases, a port may have more than one implementation. It is recommended that this be represented as multiple *portImplementations* instead of multiple *dots* within a single *connectLocation*.\*\* The example below illustrates a multiple *portImplementation*, with two *portImplementations* for port *nq*. It is diagrammed in Figure 5-2.

```
(cell doublepins (cellType Generic)
  (view V1 (viewType Schematic)
    (interface
      (port a)
      (port b)
      (port nq)
      (port q)
      (port y)
      (symbol
        (portImplementation a ...)
        (portImplementation b ...)
        (portImplementation nq ...)
        (portImplementation nq ...)
        (portImplementation q ...)
        (portImplementation y ...)
        ...))))
```

---

\*STSC Recommendation (2)

\*\*STSC Recommendation (3)

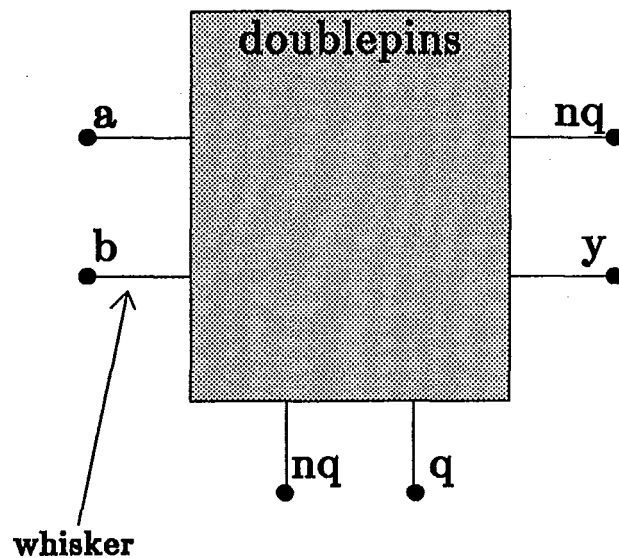


Figure 5-2 Multiple Port Implementations

In some systems the "whiskers" of a symbol are considered the ports. These whiskers should be implemented as figures inside of *portImplementation*. Some systems consider the whiskers of a symbol as part of the symbol body. Whiskers in these systems should be implemented as part of the figures inside of *symbol*. EDIF does not place a restriction on one interpretation as being right and another one wrong; it allows both implementations to be described. It is the responsibility of the schematic reader to handle both situations.

The following example illustrates the use of *portImplementation*.

```
(cell Symbol_1
  (cellType Generic)
  (view Graphic_Schematic (viewType Schematic)
    (interface
      (port P1
        (designator "1")
        (direction Input))
      ...
    (symbol
      (portImplementation
        (name P1
          (display Default_1
            (justify LowerLeft))
```

```
(orientation R0)
(origin (pt 500 0))))
(keywordDisplay designator
(display Default_1
(justify LowerCenter)
(orientation R0)
(origin (pt 0 -750))))
(connectLocation
 (figure Default_1
  (dot (pt 0 0))))
(figure Default_1
 (comment "Graphics for the Pin")
 (path
  (pointList
   (pt 0 0)
   (pt 2500 0))))
...)))))
```

Figure 5-3 illustrates this example, including the name and designator text, and the display of port graphics and the connect location. The coordinates are relative to the symbol origin and not any particular object (such as a pin) in the symbol.

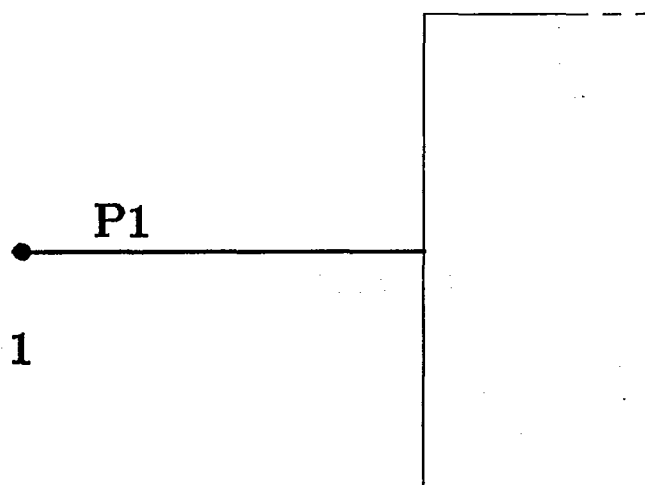


**Figure 5-3 A Single Port Implementation**

## The Reference Designator

A reference designator uniquely identifies an instance so that the instance may be associated with a physical part. The default reference designator of the symbol is defined by the *interface designator*. The reference *designator* may be a null string. Its default location can be specified using a *keywordDisplay* within *symbol*. The *designator* value and location can be overridden inside the *instance* by means of *designator* and *stringDisplay*.

For example, a designator may be initially specified as null along with its default location. When the view is instantiated, the designator is given an actual value.

```
(cell symbol_1 (cellType Generic)
  (view v1 (viewType Schematic)
    (interface
      (port a)
      ...
      (designator "")
      (symbol
        (keywordDisplay designator
          (display default_figure_group
            (origin (pt ...))))
      ...))))

(cell circuit (cellType Generic)
  (view v2 (viewType Schematic)
    (interface
      ...)
    (contents
      (page page1
        (instance Cmp1
          (viewRef v1 (cellRef symbol_1))
          (designator "V12")
          (transform
            ...))))))
```

Here, the designator value has been changed from "" (null) to "V12". The example on the next page uses the same EDIF description as above. *StringDisplay* is used to change the display location to (100, 50), relative to the schematic page origin.