

# INTERNATIONAL STANDARD

**IEC**  
**61691-1**

First edition  
1997-06

---

---

**Design automation –**

**Part 1:  
VHDL language reference manual**

*Automatisation de la conception –*

*Partie 1:  
Manuel de référence du langage VHDL*

IEC 61691-1:1997

<https://standards.iteh.ai/en/standards/iec/14358097-9e2a-4077-9a05-b8ef03056469/iec-61691-1-1997>



Numéro de référence  
Reference number  
IEC 61691-1: 1997 (E)

## Validité de la présente publication

Le contenu technique des publications de la CEI est constamment revu par la CEI afin qu'il reflète l'état actuel de la technique.

Des renseignements relatifs à la date de reconfirmation de la publication sont disponibles auprès du Bureau Central de la CEI.

Les renseignements relatifs à ces révisions, à l'établissement des éditions révisées et aux amendements peuvent être obtenus auprès des Comités nationaux de la CEI et dans les documents ci-dessous:

- **Bulletin de la CEI**
- **Annuaire de la CEI**  
Publié annuellement
- **Catalogue des publications de la CEI**  
Publié annuellement et mis à jour régulièrement

## Terminologie

En ce qui concerne la terminologie générale, le lecteur se reportera à la CEI 60050: *Vocabulaire Electrotechnique International* (VEI), qui se présente sous forme de chapitres séparés traitant chacun d'un sujet défini. Des détails complets sur le VEI peuvent être obtenus sur demande. Voir également le dictionnaire multilingue de la CEI.

Les termes et définitions figurant dans la présente publication ont été soit tirés du VEI, soit spécifiquement approuvés aux fins de cette publication.

## Symboles graphiques et littéraux

Pour les symboles graphiques, les symboles littéraux et les signes d'usage général approuvés par la CEI, le lecteur consultera:

- la CEI 60027: *Symboles littéraux à utiliser en électrotechnique*;
- la CEI 60417: *Symboles graphiques utilisables sur le matériel. Index, relevé et compilation des feuilles individuelles*;
- la CEI 60617: *Symboles graphiques pour schémas*;

et pour les appareils électromédicaux,

- la CEI 60878: *Symboles graphiques pour équipements électriques en pratique médicale*.

Les symboles et signes contenus dans la présente publication ont été soit tirés de la CEI 60027, de la CEI 60417, de la CEI 60617 et/ou de la CEI 60878, soit spécifiquement approuvés aux fins de cette publication.

## Publications de la CEI établies par le même comité d'études

L'attention du lecteur est attirée sur les listes figurant à la fin de cette publication, qui énumèrent les publications de la CEI préparées par le comité d'études qui a établi la présente publication.

## Validity of this publication

The technical content of IEC publications is kept under constant review by the IEC, thus ensuring that the content reflects current technology.

Information relating to the date of the reconfirmation of the publication is available from the IEC Central Office.

Information on the revision work, the issue of revised editions and amendments may be obtained from IEC National Committees and from the following IEC sources:

- **IEC Bulletin**
- **IEC Yearbook**  
Published yearly
- **Catalogue of IEC publications**  
Published yearly with regular updates

## Terminology

For general terminology, readers are referred to IEC 60050: *International Electrotechnical Vocabulary* (IEV), which is issued in the form of separate chapters each dealing with a specific field. Full details of the IEV will be supplied on request. See also the IEC Multilingual Dictionary.

The terms and definitions contained in the present publication have either been taken from the IEV or have been specifically approved for the purpose of this publication.

## Graphical and letter symbols

For graphical symbols, and letter symbols and signs approved by the IEC for general use, readers are referred to publications:

- IEC 60027: *Letter symbols to be used in electrical technology*;
- IEC 60417: *Graphical symbols for use on equipment. Index, survey and compilation of the single sheets*;
- IEC 60617: *Graphical symbols for diagrams*;

and for medical electrical equipment,

- IEC 60878: *Graphical symbols for electromedical equipment in medical practice*.

The symbols and signs contained in the present publication have either been taken from IEC 60027, IEC 60417, IEC 60617 and/or IEC 60878, or have been specifically approved for the purpose of this publication.

## IEC publications prepared by the same technical committee

The attention of readers is drawn to the end pages of this publication which list the IEC publications issued by the technical committee which has prepared the present publication.

# INTERNATIONAL STANDARD

**IEC**  
**61691-1**

First edition  
1997-06

---

---

## Design automation –

### Part 1: VHDL language reference manual

*Automatisation de la conception –*

*Partie 1:  
Manuel de référence du langage VHDL*

© IEC 1997 Droits de reproduction réservés – Copyright - all rights reserved

Aucune partie de cette publication ne peut être reproduite ni utilisée sous quelque forme que ce soit et par aucun procédé, électronique ou mécanique, y compris la photo-copie et les microfilms, sans l'accord écrit de l'éditeur.

No part of this publication may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm, without permission in writing from the publisher.

International Electrotechnical Commission  
Telefax: +41 22 919 0300

3, rue de Varembé Geneva, Switzerland  
e-mail: [inmail@iec.ch](mailto:inmail@iec.ch) IEC web site <http://www.iec.ch>



Commission Electrotechnique Internationale  
International Electrotechnical Commission  
Международная Электротехническая Комиссия

CODE PRIX  
PRICE CODE

**XH**

*Pour prix, voir catalogue en vigueur  
For price, see current catalogue*

## CONTENTS

	Page
FOREWORD.....	7
Clause	
<b>Section 0: General</b> .....	8
0.1 Scope and object.....	8
0.2 Structure and terminology .....	8
0.2.1 Syntactic description.....	9
0.2.2 Semantic description.....	10
0.2.3 Front matter, examples, notes, references, and annexes .....	10
0.2.4 Normative reference.....	11
<b>Section 1: Design entities and configurations</b> .....	11
1.1 Entity declarations .....	11
1.1.1 Entity header .....	12
1.1.1.1 Generics.....	12
1.1.1.2 Ports.....	13
1.1.2 Entity declarative part.....	14
1.1.3 Entity statement part.....	15
1.2 Architecture bodies.....	15
1.2.1 Architecture declarative part.....	16
1.2.2 Architecture statement part.....	16
1.3 Configuration declarations.....	18
1.3.1 Block configuration.....	19
1.3.2 Component configuration .....	21
<b>Section 2: Subprograms and packages</b> .....	23
2.1 Subprogram declarations .....	23
2.1.1 Formal parameters.....	24
2.1.1.1 Constant and variable parameters.....	24
2.1.1.2 Signal parameters .....	25
2.1.1.3 File parameters .....	26
2.2 Subprogram bodies .....	26
2.3 Subprogram overloading .....	29
2.3.1 Operator overloading .....	30
2.3.2 Signatures.....	30
2.4 Resolution functions .....	31
2.5 Package declarations.....	32
2.6 Package bodies .....	33
2.7 Conformance rules.....	34
<b>Section 3: Types</b> .....	35
3.1 Scalar Types .....	36
3.1.1 Enumeration types.....	37
3.1.1.1 Predefined enumeration types .....	38

3.1.2	Integer types.....	38
3.1.2.1	Predefined integer types.....	39
3.1.3	Physical types.....	39
3.1.3.1	Predefined physical types.....	41
3.1.4	Floating point types.....	41
3.1.4.1	Predefined floating point types.....	42
3.2	Composite types.....	42
3.2.1	Array types.....	42
3.2.1.1	Index constraints and discrete ranges.....	44
3.2.1.2	Predefined array types.....	47
3.2.2	Record types.....	47
3.3	Access types.....	48
3.3.1	Incomplete type declarations.....	48
3.3.2	Allocation and deallocation of objects.....	49
3.4	File types.....	50
3.4.1	File operations.....	50
<b>Section 4: Declarations.....</b>		<b>52</b>
4.1	Type declarations.....	53
4.2	Subtype declarations.....	54
4.3	Objects.....	55
4.3.1	Object declarations.....	55
4.3.1.1	Constant declarations.....	56
4.3.1.2	Signal declarations.....	56
4.3.1.3	Variable declarations.....	58
4.3.1.4	File declarations.....	59
4.3.2	Interface declarations.....	60
4.3.2.1	Interface lists.....	62
4.3.2.2	Association lists.....	63
4.3.3	Alias declarations.....	65
4.3.3.1	Object aliases.....	66
4.3.3.2	Nonobject aliases.....	67
4.4	Attribute declarations.....	68
4.5	Component declarations.....	69
4.6	Group template declarations.....	69
4.7	Group declarations.....	70
<b>Section 5: Specifications.....</b>		<b>71</b>
5.1	Attribute specification.....	71
5.2	Configuration specification.....	73
5.2.1	Binding indication.....	74
5.2.1.1	Entity aspect.....	76
5.2.1.2	Generic map and port map aspects.....	77
5.2.2	Default binding indication.....	79
5.3	Disconnection specification.....	80

<b>Section 6: Names</b> .....	82
6.1 Names.....	82
6.2 Simple names.....	84
6.3 Selected names.....	84
6.4 Indexed names.....	87
6.5 Slice names.....	87
6.6 Attribute names.....	88
<b>Section 7: Expressions</b> .....	88
7.1 Expressions.....	88
7.2 Operators.....	90
7.2.1 Logical operators.....	90
7.2.2 Relational operators.....	91
7.2.3 Shift operators.....	92
7.2.4 Adding operators.....	94
7.2.5 Sign operators.....	96
7.2.6 Multiplying operators.....	96
7.2.7 Miscellaneous operators.....	98
7.3 Operands.....	99
7.3.1 Literals.....	99
7.3.2 Aggregates.....	100
7.3.2.1 Record aggregates.....	100
7.3.2.2 Array aggregates.....	101
7.3.3 Function calls.....	102
7.3.4 Qualified expressions.....	102
7.3.5 Type conversions.....	103
7.3.6 Allocators.....	104
7.4 Static expressions.....	105
7.4.1 Locally static primaries.....	105
7.4.2 Globally static primaries.....	106
7.5 Universal expressions.....	107
<b>Section 8: Sequential statements</b> .....	108
8.1 Wait statement.....	109
8.2 Assertion statement.....	111
8.3 Report statement.....	111
8.4 Signal assignment statement.....	112
8.4.1 Updating a projected output waveform.....	114
8.5 Variable assignment statement.....	117
8.5.1 Array variable assignments.....	117
8.6 Procedure call statement.....	118
8.7 If statement.....	118
8.8 Case statement.....	119
8.9 Loop statement.....	120
8.10 Next statement.....	121
8.11 Exit statement.....	121
8.12 Return statement.....	121
8.13 Null statement.....	122

<b>Section 9: Concurrent statements</b> .....	122
9.1 Block statement .....	123
9.2 Process statement.....	124
9.3 Concurrent procedure call statements.....	125
9.4 Concurrent assertion statements .....	126
9.5 Concurrent signal assignment statements .....	127
9.5.1 Conditional signal assignments.....	129
9.5.2 Selected signal assignments .....	130
9.6 Component instantiation statements.....	131
9.6.1 Instantiation of a component.....	132
9.6.2 Instantiation of a design entity .....	134
9.7 Generate statements .....	137
<b>Section 10: Scope and visibility</b> .....	138
10.1 Declarative region.....	138
10.2 Scope of declarations.....	139
10.3 Visibility .....	140
10.4 Use clauses .....	143
10.5 The context of overload resolution .....	144
<b>Section 11: Design units and their analysis</b> .....	145
11.1 Design units .....	145
11.2 Design libraries.....	145
11.3 Context clauses.....	146
11.4 Order of analysis.....	147
<b>Section 12: Elaboration and execution</b> .....	147
12.1 Elaboration of a design hierarchy.....	148
12.2 Elaboration of a block header.....	149
12.2.1 The generic clause.....	149
12.2.2 The generic map aspect.....	150
12.2.3 The port clause .....	150
12.2.4 The port map aspect .....	150
12.3 Elaboration of a declarative part.....	151
12.3.1 Elaboration of a declaration .....	151
12.3.1.1 Subprogram declarations and bodies.....	152
12.3.1.2 Type declarations.....	152
12.3.1.3 Subtype declarations.....	152
12.3.1.4 Object declarations .....	153
12.3.1.5 Alias declarations .....	153
12.3.1.6 Attribute declarations .....	153
12.3.1.7 Component declarations .....	153
12.3.2 Elaboration of a specification.....	153
12.3.2.1 Attribute specifications.....	154
12.3.2.2 Configuration specifications.....	154
12.3.2.3 Disconnection specifications .....	154

12.4	Elaboration of a statement part .....	155
12.4.1	Block statements .....	155
12.4.2	Generate statements .....	155
12.4.3	Component instantiation statements.....	157
12.4.4	Other concurrent statements.....	157
12.5	Dynamic elaboration.....	157
12.6	Execution of a model.....	158
12.6.1	Drivers.....	158
12.6.2	Propagation of signal values .....	159
12.6.3	Updating implicit signals .....	162
12.6.4	The simulation cycle .....	163
<b>Section 13:</b>	<b>Lexical elements.....</b>	<b>164</b>
13.1	Character set .....	164
13.2	Lexical elements, separators, and delimiters .....	166
13.3	Identifiers.....	167
13.3.1	Basic identifiers.....	167
13.3.2	Extended identifiers .....	168
13.4	Abstract literals.....	168
13.4.1	Decimal literals .....	168
13.4.2	Based literals .....	169
13.5	Character literals .....	169
13.6	String literals .....	170
13.7	Bit string literals .....	170
13.8	Comments .....	172
13.9	Reserved words .....	172
13.10	Allowable replacements of characters.....	173
<b>Section 14:</b>	<b>Predefined language environment.....</b>	<b>173</b>
14.1	Predefined attributes.....	173
14.2	Package STANDARD.....	187
14.3	Package TEXTIO.....	194
<b>Annex A –</b>	<b>Syntax summary .....</b>	<b>199</b>
<b>Annex B –</b>	<b>Glossary .....</b>	<b>216</b>
<b>Annex C –</b>	<b>Potentially nonportable constructs .....</b>	<b>233</b>
<b>Annex D –</b>	<b>Changes from IEEE Std 1076-1987 .....</b>	<b>235</b>
<b>Annex E –</b>	<b>Related standards .....</b>	<b>236</b>
<b>Index .....</b>		<b>237</b>



## INTERNATIONAL ELECTROTECHNICAL COMMISSION

## DESIGN AUTOMATION –

## Part 1: VHDL language reference manual

## FOREWORD

- 1) The IEC (International Electrotechnical Commission) is a worldwide organization for standardization comprising all national electrotechnical committees (IEC National Committees). The object of the IEC is to promote international co-operation on all questions concerning standardization in the electrical and electronic fields. To this end and in addition to other activities, the IEC publishes International Standards. Their preparation is entrusted to technical committees; any IEC National Committee interested in the subject dealt with may participate in this preparatory work. International, governmental and non-governmental organizations liaising with the IEC also participate in this preparation. The IEC collaborates closely with the International Organization for Standardization (ISO) in accordance with conditions determined by agreement between the two organizations.
- 2) The formal decisions or agreements of the IEC on technical matters express, as nearly as possible, an international consensus of opinion on the relevant subjects since each technical committee has representation from all interested National Committees.
- 3) The documents produced have the form of recommendations for international use and are published in the form of standards, technical reports or guides and they are accepted by the National Committees in that sense.
- 4) In order to promote international unification, IEC National Committees undertake to apply IEC International Standards transparently to the maximum extent possible in their national and regional standards. Any divergence between the IEC Standard and the corresponding national or regional standard shall be clearly indicated in the latter.
- 5) The IEC provides no marking procedure to indicate its approval and cannot be rendered responsible for any equipment declared to be in conformity with one of its standards.
- 6) Attention is drawn to the possibility that some of the elements of this International Standard may be the subject of patent rights. The IEC shall not be held responsible for identifying any or all such patent rights.

International Standard IEC 61691-1 has been prepared by IEC technical committee 93: Design automation.

This standard is based on IEEE Std 1076: *IEEE Standard VHDL language Reference Manual (1993)*.

IEC 61691 consists of the following parts, under the general title *Design automation*:

- Part 1: 1997, *VHDL language reference manual*
- Part 2, – *Multivalued logic system for VHDL model interoperability*<sup>1)</sup>

*This standard does not follow the rules for the structure of international standards given in Part 3 of the ISO/IEC Directives.*

The text of this standard is based on the following documents:

FDIS	Report on voting
93/42/FDIS	93/45/RVD

Full information on the voting for the approval of this standard can be found in the report on voting indicated in the above table.

Annexes A, B, C, D, and E are for information only.

<sup>1)</sup> To be published

# Design automation – VHDL language reference manual

## Section 0: General

This section describes the purpose and organization of this International Standard, Very High Speed Integrated Circuit (VHSIC) Hardware Description Language (VHDL) language reference manual.

### 0.1 Scope and object

The object of this international standard is to define VHDL accurately. Its primary audiences are the implementors of tools supporting the language and the advanced users of the language. Other users are encouraged to use commercially available books, tutorials, and classes to learn the language in some detail prior to reading this standard. These resources generally focus on how to use the language, rather than how a VHDL-compliant tool is required to behave.

At the time of its publication, this standard was the authoritative definition of VHDL. From time to time, it may become necessary to correct and/or clarify portions of this standard. Such corrections and clarifications may be published in separate standards. These modify this standard at the time of their publication and remain in effect until superseded by subsequent standards, or until the standard is officially revised.

### 0.2 Structure and terminology

This standard is organized into sections, each of which focuses on some particular area of the language. Within each section, individual constructs or concepts are discussed in each clause.

Each clause describing a specific construct begins with an introductory paragraph. Next, the syntax of the construct is described using one or more grammatical “productions.”

A set of paragraphs describing the meaning and restrictions of the construct in narrative form then follow. Unlike many other IEEE standards, which use the verb “shall” to indicate mandatory requirements of the standard and “may” to indicate optional features, the verb “is” is used uniformly throughout this standard. In all cases, “is” is to be interpreted as having mandatory weight.

Additionally, the word “must” is used to indicate mandatory weight. This word is preferred over the more common “shall,” as “must” denotes a different meaning to different readers of this standard.

- a) To the developer of tools that process VHDL, “must” denotes a requirement that the standard imposes. The resulting implementation is required to enforce the requirement and to issue an error if the requirement is not met by some VHDL source text.
- b) To the VHDL model developer, “must” denotes that the characteristics of VHDL are natural consequences of the language definition. The model developer is required to adhere to the constraint implied by the characteristic.
- c) To the VHDL model user, “must” denotes that the characteristics of the models are natural consequences of the language definition. The model user can depend on the characteristics of the model implied by its VHDL source text.

Finally, each clause may end with examples, notes, and references to other pertinent clauses.

### 0.2.1 Syntactic description

The form of a VHDL description is described by means of context-free syntax, using a simple variant of Backus-Naur form; in particular:

- a) Lowercased words in roman font, some containing embedded underlines, are used to denote syntactic categories, for example:

formal\_port\_list

Whenever the name of a syntactic category is used, apart from the syntax rules themselves, spaces take the place of underlines (thus, “formal port list” would appear in the narrative description when referring to the above syntactic category).

- b) Boldface words are used to denote reserved words, for example:

**array**

Reserved words must be used only in those places indicated by the syntax.

- c) A *production* consists of a *left-hand side*, the symbol “::=” (which is read as “can be replaced by”), and a *right-hand side*. The left-hand side of a production is always a syntactic category; the right-hand side is a replacement rule.

The meaning of a production is a textual-replacement rule: any occurrence of the left-hand side may be replaced by an instance of the right-hand side.

- d) A vertical bar separates alternative items on the right-hand side of a production unless it occurs immediately after an opening brace, in which case it stands for itself:

letter\_or\_digit ::= letter | digit

choices ::= choice { | choice }

In the first instance, an occurrence of “letter\_or\_digit” can be replaced by either “letter” or “digit.” In the second case, “choices” can be replaced by a list of “choice,” separated by vertical bars [see item f) for the meaning of braces].

- e) Square brackets enclose optional items on the right-hand side of a production; thus the two following productions are equivalent:

return\_statement ::= **return** [ expression ] ;

return\_statement ::= **return** ; | **return** expression ;

Note, however, that the initial and terminal square brackets in the right-hand side of the production for signatures (in 2.3.2) are part of the syntax of signatures and do not indicate that the entire right-hand side is optional.

- f) Braces enclose a repeated item or items on the right-hand side of a production. The items may appear zero or more times; the repetitions occur from left to right as with an equivalent left-recursive rule. Thus, the following two productions are equivalent:

term ::= factor { multiplying\_operator factor }

term ::= factor | term multiplying\_operator factor

- g) If the name of any syntactic category starts with an italicized part, it is equivalent to the category name without the italicized part. The italicized part is intended to convey some semantic information. For example, *type\_name* and *subtype\_name* are both syntactically equivalent to name alone.
- h) The term *simple\_name* is used for any occurrence of an identifier that already denotes some declared entity.

## 0.2.2 Semantic description

The meaning and restrictions of a particular construct are described with a set of narrative rules immediately following the syntactic productions. In these rules, an italicized term indicates the definition of that term and identifiers appearing entirely in uppercase refer to definitions in package STANDARD (see clause 14.2).

The following terms are used in these semantic descriptions with the following meaning:

**erroneous:** The condition described represents an ill-formed description; however, implementations are not required to detect and report this condition.

Conditions are deemed erroneous only when it is impossible in general to detect the condition during the processing of the language.

**error:** The condition described represents an ill-formed description; implementations are required to detect the condition and report an error to the user of the tool.

**illegal:** A synonym for “error.”

**legal:** The condition described represents a well-formed description.

## 0.2.3 Front matter, examples, notes, references, and annexes

Some clauses of this standard contain examples, notes, and cross-references to other clauses of the standard; these parts always appear at the end of a clause. Examples are meant to illustrate the possible forms of the construct described. Illegal examples are italicized. Notes are meant to emphasize consequences of the rules described in the clause or elsewhere. In order to distinguish notes from the other narrative portions of this standard, notes are in a type smaller than the rest of the text. Cross-references are meant to guide the user to other relevant clauses of the standard. Examples, notes, and cross-references are not part of the definition of the language.

## 0.2.4 Normative reference

The following normative document contains provisions which, through reference in this text, constitute provisions of this International Standard. At the time of publication, the edition indicated was valid. All normative documents are subject to revision, and parties to agreements based on this International Standard are encouraged to investigate the possibility of applying the most recent edition of the normative document indicated below. Members of IEC and ISO maintain registers of currently valid International Standards.

ISO 8859-1: 1987, *Information processing – 8-bit single-byte coded graphic character sets – Part 1: Latin alphabet No. 1*

## Section 1: Design entities and configurations

The design entity is the primary hardware abstraction in VHDL. It represents a portion of a hardware design that has well-defined inputs and outputs and performs a well-defined function. A design entity may represent an entire system, a subsystem, a board, a chip, a macro-cell, a logic gate, or any level of abstraction in between. A configuration can be used to describe how design entities are put together to form a complete design.

A design entity may be described in terms of a hierarchy of blocks, each of which represents a portion of the whole design. The top-level block in such a hierarchy is the design entity itself; such a block is an external block that resides in a library and may be used as a component of other designs. Nested blocks in the hierarchy are internal blocks, defined by block statements (see clause 9.1).

A design entity may also be described in terms of interconnected components. Each component of a design entity may be bound to a lower-level design entity in order to define the structure or behaviour of that component. Successive decomposition of a design entity into components, and binding those components to other design entities that may be decomposed in like manner, results in a hierarchy of design entities representing a complete design. Such a collection of design entities is called a design hierarchy. The bindings necessary to identify a design hierarchy can be specified in a configuration of the top-level entity in the hierarchy.

This section describes the way in which design entities and configurations are defined. A design entity is defined by an entity declaration together with a corresponding architecture body. A configuration is defined by a configuration declaration.

### 1.1 Entity declarations

An entity declaration defines the interface between a given design entity and the environment in which it is used. It may also specify declarations and statements that are part of the design entity. A given entity declaration may be shared by many design entities, each of which has a different architecture. Thus, an entity declaration can potentially represent a class of design entities, each with the same interface.

```
entity_declaration ::=
  entity identifier is
    entity_header
    entity_declarative_part
  [ begin
    entity_statement_part ]
  end [ entity ] [ entity_simple_name ] ;
```

The entity header and entity declarative part consist of declarative items that pertain to each design entity whose interface is defined by the entity declaration. The entity statement part, if present, consists of concurrent statements that are present in each such design entity.

If a simple name appears at the end of an entity declaration, it must repeat the identifier of the entity declaration.

### 1.1.1 Entity header

The entity header declares objects used for communication between a design entity and its environment.

```
entity_header ::=
    [ formal_generic_clause ]
    [ formal_port_clause ]
```

```
generic_clause ::=
    generic ( generic_list ) ;
```

```
port_clause ::=
    port ( port_list ) ;
```

The generic list in the formal generic clause defines generic constants whose values may be determined by the environment. The port list in the formal port clause defines the input and output ports of the design entity.

In certain circumstances, the names of generic constants and ports declared in the entity header become visible outside the design entity (see clauses 10.2 and 10.3).

*Examples:*

—An entity declaration with port declarations only:

```
entity Full_Adder is
    port ( X, Y, Cin: in Bit; Cout, Sum: out Bit) ;
end Full_Adder ;
```

—An entity declaration with generic declarations also:

```
entity AndGate is
    generic
        ( N: Natural := 2);
    port
        ( Inputs: in Bit_Vector (1 to N);
          Result: out Bit);
end entity AndGate ;
```

—An entity declaration with neither:

```
entity TestBench is
end TestBench ;
```

#### 1.1.1.1 Generics

Generics provide a channel for static information to be communicated to a block from its environment. The following applies to both external blocks defined by design entities and to internal blocks defined by block statements.

```
generic_list ::= generic_interface_list
```

The generics of a block are defined by a generic interface list; interface lists are described in 4.3.2.1. Each interface element in such a generic interface list declares a formal generic.