INTERNATIONAL STANDARD

IEC 62016

First edition
2003-12

Core model of the electronics domain

Withdrawn

IECNORM.COM : Click to view the full PDF of IEC 62016:2003

Reference number
IEC 62016:2003(E)

Publication numbering

As from 1 January 1997 all IEC publications are issued with a designation in the 60000 series. For example, IEC 34-1 is now referred to as IEC 60034-1.

Consolidated editions

The IEC is now publishing consolidated versions of its publications. For example, edition numbers 1.0, 1.1 and 1.2 refer, respectively, to the base publication, the base publication incorporating amendment 1 and the base publication incorporating amendments 1 and 2.

Further information on IEC publications

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 this publication, including its validity, is available in the IEC Catalogue of publications (see below) in addition to new editions, amendments and corrigenda. Information on the subjects under consideration and work in progress undertaken by the technical committee which has prepared this publication, as well as the list of publications issued, is also available from the following:

- **IEC Web Site (www.iec.ch)**

- **Catalogue of IEC publications**

  The on-line catalogue on the IEC web site (http://www.iec.ch/searchpub/cur_fut.htm) enables you to search by a variety of criteria including text searches, technical committees and date of publication. On-line information is also available on recently issued publications, withdrawn and replaced publications, as well as corrigenda.

- **IEC Just Published**

  This summary of recently issued publications (http://www.iec.ch/online_news/justpub/jp_entry.htm) is also available by email. Please contact the Customer Service Centre (see below) for further information.

- **Customer Service Centre**

  If you have any questions regarding this publication or need further assistance, please contact the Customer Service Centre:

  Email: custserv@iec.ch
  Tel: +41 22 919 02 11
  Fax: +41 22 919 03 00

IECNORM.COM : Click to view the full PDF of IEC 62016:2003
Withdrawn

INTERNATIONAL STANDARD

IEC 62016

First edition
2003-12

Core model of the electronics domain

Withdrawn
IECNORM.COM : Click to view the full PDF of IEC 62016:2003

© IEC 2003 – Copyright - all rights reserved

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, 3, rue de Varembé, PO Box 131, CH-1211 Geneva 20, Switzerland
Telephone: +41 22 919 02 11 Telefax: +41 22 919 03 00 E-mail: inmail@iec.ch Web: www.iec.ch

Commission Electrotechnique Internationale
International Electrotechnical Commission
Международная Электротехническая Комиссия

PRICE CODE XH

For price, see current catalogue

CONTENTS

Withdrawn
IECNORM.COM : Click to view the full PDF of IEC 62016:2003

IECNORM.COM : Click to view the full PDF of IEC 62016:2003
Withdrawn

Withdrawn
IECNORM.COM : Click to view the full PDF of IEC 62016:2003

INTERNATIONAL ELECTROTECHNICAL COMMISSION

CORE MODEL OF THE ELECTRONICS DOMAIN

FOREWORD

1) The International Electrotechnical Commission (IEC) is a worldwide organization for standardization comprising all national electrotechnical committees (IEC National Committees). The object of 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, IEC publishes International Standards, Technical Specifications, Technical Reports, Publicly Available Specifications (PAS) and Guides (hereafter referred to as "IEC Publication(s)"). 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. 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 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 IEC National Committees.
3) IEC Publications have the form of recommendations for international use and are accepted by IEC National Committees in that sense. While all reasonable efforts are made to ensure that the technical content of IEC Publications is accurate, IEC cannot be held responsible for the way in which they are used or for any misinterpretation by any end user.
4) In order to promote international uniformity, IEC National Committees undertake to apply IEC Publications transparently to the maximum extent possible in their national and regional publications. Any divergence between any IEC Publication and the corresponding national or regional publication shall be clearly indicated in the latter.
5) IEC provides no marking procedure to indicate its approval and cannot be rendered responsible for any equipment declared to be in conformity with an IEC Publication.
6) All users should ensure that they have the latest edition of this publication.
7) No liability shall attach to IEC or its directors, employees, servants or agents including individual experts and members of its technical committees and IEC National Committees for any personal injury, property damage or other damage of any nature whatsoever, whether direct or indirect, or for costs (including legal fees) and expenses arising out of the publication, use of, or reliance upon, this IEC Publication or any other IEC Publications.
8) Attention is drawn to the Normative references cited in this publication. Use of the referenced publications is indispensable for the correct application of this publication.
9) Attention is drawn to the possibility that some of the elements of this IEC Publication may be the subject of patent rights. IEC shall not be held responsible for identifying any or all such patent rights.

IECNORM.COM : Click to view the full PDF of IEC 62016:2003

International Standard IEC 62016 has been prepared by IEC technical committee 93: Design automation.

The document was released by the feeder organization, Government Electronics and Information Technology Association, a sector of the Electronic Industries Alliance (EIA), for committee draft, comment, and review by members of IEC TC93 [1].

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

FDIS	Report on voting
93/172/FDIS	93/176/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.

[1] The EIA feeder organization retains the copyright and intellectual property of this work but provides permission for IEC to reproduce and exploit the information contained herein.

This standard does not follow the rules for the structure of International Standards given in the ISO/IEC Directives, Part 2.

The committee has decided that the contents of this publication will remain unchanged until 2012-07. At this date, the publication will be

- reconfirmed;
- withdrawn;
- replaced by a revised edition, or
- amended.

Withdrawn
IECNORM.COM : Click to view the full PDF of IEC 62016:2003

INTRODUCTION

The Core Model of the electronics domain provides a common basis for design information handled by CAD systems within the electronic domain. It is the purpose of this model to provide a conceptual representation of the electronics domain, so that the compliant CAD systems handle a similar set of concepts, thus making inter-communication, sharing and exchange of design information a much easier task. It is not the purpose of this model to describe implementation details or to provide a data representation of electronics domain information.

The Core Model of the electronics domain, Edition 1.0, is referred to as the “Core Model” throughout this document. The Core Model, in part, has been created by enhancing the industry connectivity consensus model, EDIF CFI DR Alignment Model Version 1.0 (www.edif.org).

The chosen description language for this Core Model is EXPRESS, as defined by ISO 10303-11.

It is necessary to describe the Core Model as an information model in order to provide a formal definition of the design information that shall be be recognized by the compliant CAD systems. The benefits of a formal description derive from its ability to provide an unambiguous representation of concepts, attributes and relationships, and the global rules and constraints that may be applied. By having such a description, it is possible to check the consistency and the correctness of the model as well as to provide a reliable starting-point for further development. It also facilitates the design of correct electronics CAD implementations based on this Core Model, as the actual implementation methods can be checked against the model.

This Core Model includes connectivity, hierarchy and design information for the electronics domain. Future parts of this Core Model standard may be extended to include other categories of information (for example, *cell_representation*, schematic representation, the PCB domain, symbols and display information).

In order to facilitate the creation of other parts of this Core Model standard, some objects have been used in this Core Model to facilitate support for other parts of the electronics domain. There are two types of such objects.

- Entities, such as *cell_representation*, are important concepts that provide support for defining other Core Model parts.
- Constraints: Some of constraints of this Core Model use conditions that are always true. They have been written in this way in order to ensure that they remain valid when the model is extended.

IECNORM.COM: Click to view the full PDF of IEC 62016:2003

CORE MODEL OF THE ELECTRONICS DOMAIN

1 Scope and object

This International Standard provides the semantics definitions for the following categories of information related to electronic circuit designs. Each category of design information is modelled as an EXPRESS schema.

The Core Model consists of 10 schemas. Each of them is presented in this document as a separate chapter. At the beginning of each chapter, a description of the corresponding schema is provided.

- The *hierarchy_model* schema describes the hierarchical information of a cell, i.e. the way a cell may be divided into other cells.

  A circuit may be divided into cells which, in turn, may be further subdivided into other cells, thus creating a hierarchy. The hierarchy information describes the cells, the possible cell representations and their instances.

- The *design_hierarchy_model* schema describes the annotation on an occurrence hierarchy.

  The definition of a design requires that specific representations (views) of design objects in the hierarchy are selected. This unambiguously creates a configured design hierarchy. This concept is similar to the configuration of a design in VHDL and is related to view selection mechanisms of other electronics design domain information models in industry. The design hierarchy identifies top-level design cells and may provide annotated design-specific data into the elaborated hierarchy.

- The *connectivity_view_model* schema describes the connectivity information of a cell.

  This describes the way in which the circuits are connected in order that information or energy may flow from one part of a design or product to another. The Core Model subdivides this information into

  - the *logical_connectivity_model* schema which describes the connectivity for a given level of a hierarchy.

    Logical connectivity information describes the bit level, abstract electrical connectivity for a given level of a hierarchy, in terms of signals and signal groups.

  - the connectivity_structure_model schema which describes the structural connectivity of a connectivity view.

    Structural connectivity information describes the connectivity, for a given level of a hierarchy, from the structural point of view. The structural connectivity is specified in terms of busses, nets and rippers. Such a structural representation is used to provide support for physical implementation and annotation.

- The *library_model* schema describes the technology information contained in a library, as well as the reusable objects and data.

  A library provides a means of grouping cell definitions. A library may be used to group other classes of reusable objects and information as well. Information in a library may be related in terms of technology information.

- The *information_base_model* schema describes the information in an information base.

  The information_base describes the kind of information that can be found directly in an information base.

IECNORM.COM : Click to view the full PDF of IEC 62016:2003

In addition, the following information is also included in the model.

- The *design_management_model* schema provides the design management information.

  This information is needed to trace back to the source or the owner of the data.
- The *documentation_model* schema describes the documentation provided for an object.
- The *support_definition_model* schema contains the definition of some auxiliary entities, types and functions that are used by several schemas.

Names of objects used in this Core Model standard were chosen to be the same as the names of the similar objects and concepts in existing electronics domain information models wherever possible.

2 Reference documents

IEC 61690-1:2000, *Electronic design interchange format (EDIF) – Part 1: Version 300*

IEC 61690-2:2000, *Electronic design interchange format (EDIF) – Part 2: Version 400*

ISO 13030-11:1994, *Industrial automation systems and integration – Product representation and exchange – Part 11: Description methods: The EXPRESS-I language reference manual*

3 General modelling issues

This standard describes the general modelling techniques and conventions used by the Core Model. The most important concept discussed here is that of relationship between entities (ownership and reference). In addition, issues such as uniqueness in aggregates, default values, empty sets and model topology are presented.

3.1 Ownership and reference

An object is said to be the “owner” of another object, if the latter can only exist in the context of the former. From the point of view of the information model, this means that an instance of the “owned” object can exist if, and only if, there is an instance of the owner object. Any given instance of an object must have exactly one owner. Some examples of owner relationships are provided below.

By contrast, if an object references another object, the existence of the latter is not dependent on the existence of the former. In the information model this means that the existence of an instance of the referenced object is not related to the existence of any instance of the object that references it. A given instance may be referenced by any number of other instances unless stated otherwise by a constraint.

Generally speaking, the referencing mechanism provides a way of sharing data whereas the ownership dependence is used to create a scope for the objects and to control their existence.

The difference between the two types of relationship is important because of their effects in an actual implementation. If the implementation provides a static representation of data (such as an EDIF file), an owned object can be textually contained in the definition of the owner, whereas a referenced object is used by the referencing object but may be declared in another context. If the system allows dynamic representation of the data (like CFI DR PI), the difference between ownership and reference is reflected in the process of object creation and destruction. Indeed, whenever an object is destroyed, its owned objects are destroyed too, unlike the referenced objects which continue to exist.

Withdrawn
IECNORM.COM : Click to view the full PDF of IEC 62016:2003

The EXPRESS language does not provide a direct method of specifying whether an object is owned or referenced. However, the Alignment Model represents the two types of relationship in different ways by using the techniques and conventions described below.

3.1.1 Ownership

Let us consider the example presented in Figure 1 which shows the owner relationship between a *library* and its contained *cells*:

```
ENTITY library
  cells : OPTIONAL SET [1:?] OF cell;
  ...
END_ENTITY;


ENTITY cell
  ...
INVERSE
  containing_library : library FOR cells;
END_ENTITY;
```

Figure 1 – The owner relationship

In this example, ownership is modelled straightforwardly by using the EXPRESS INVERSE clause. Its meaning is that, for every instance of a *cell*, there is exactly one instance of the *library* that contains that *cell* instance.

There are cases, however, when an object may have several potential owners. The example in Figure 2 shows that a *documentation* object may be created in the context of a *connectivity_generic_net* or a *connectivity_generic_bus*, etc.

The owner relationship is modelled by using the INVERSE clause. Its meaning is that there may exist, at most, one instance of the *connectivity_generic_net* and, at most, one instance of a *connectivity_generic_bus*, etc, that contain a given *documentation* instance. However, any instance of the *documentation* may exist as a member of the "document" SET in a *connectivity_generic_net* or as a member of the "document" SET in a *connectivity_generic_bus*, etc, but not as all of them. Therefore, a domain rule (the WHERE clause) is used in order to ensure that there is only one owner of the *documentation* instance in the database. Of course, there are other objects that reference that *documentation* instance.

IECNORM.COM: Click to view the full PDF of IEC 62016:2003 Withdrawn

```
ENTITY connectivity_generic_net
  ABSTRACT SUPERTYPE OF (ONEOF(connectivity_net,
                               connectivity_sub_net));
  ...
  document    : OPTIONAL SET [1:?] OF documentation;
  ...
END_ENTITY;


ENTITY connectivity_generic_bus
  ABSTRACT SUPERTYPE OF (ONEOF(connectivity_bus,
                               connectivity_bus_slice,
                               connectivity_sub_bus));
  ...
  document    : OPTIONAL SET [1:?] OF documentation;
  ...
END_ENTITY;


ENTITY documentation;
  ...
INVERSE
  containing_generic_net:
    SET [0:1] OF connectivity_generic_net FOR document;
  containing_generic_bus:
    SET [0:1] OF connectivity_generic_bus FOR document;
  ...
WHERE
  containment_constraint:
    (* Each "documentation" is defined in only one place. *)
    SIZEOF(containing_generic_net) +
    SIZEOF(containing_generic_bus) +
  ...
END_ENTITY;
```

Figure 2 – Owner relationship with multiple potential owners

Withdrawn
IECNORM.COM : Click to view the full PDF of IEC 62016:2003

3.1.2 Reference

The reference mechanism can be used to share common data between several objects. Let us consider a *master_logical_port* which has a default connection to a *global_port*. Since it is possible for several *master_logical_ports* to have the same default connection, the *global_port* is referenced by the *master_logical_port*. In this case, the *global_port* does not contain an INVERSE attribute for the *master_logical_port* because it is not owned by it. In order to increase the readability of the model, the references to other objects are tagged as such. Figure 3 gives an example of object referencing.

```
ENTITY master_logical_port;
  default_connection : OPTIONAL global_port; -- reference
  ...
END_ENTITY;
```

Figure 3 – The reference mechanism

3.2 Uniqueness by value

EXPRESS defines the uniqueness in aggregates to be "by reference". This means that a set cannot contain the same object twice but it may contain two different objects with exactly the same value. Therefore, a special EXPRESS function is used whenever uniqueness by value is required, as shown in Figure 4.

```
ENTITY library
  document            : OPTIONAL SET [1:?] OF documentation;
  status_of_copyright : OPTIONAL SET [1:?] OF copyright;
  status_of_written   : OPTIONAL SET [1:?] OF written;
  ...
WHERE
  unique_status_of_copyright:
    value_unique(status_of_copyright);
  unique_status_of_written:
    value_unique(status_of_written);
  unique_document:
    value_unique(document);
END_ENTITY;
```

Figure 4 – Uniqueness by value

The example shows that a *library* should not contain two identical documentation, copyright information or author information, i.e. it cannot contain two instances of *documentation*, *copyright* or *written* with the same value.

3.3 Default values

The default values, which are used by the CAD systems, are relevant to the actual implementations rather than to the information models. Therefore, the Core Model does not contain a specification of these values.

3.4 Optional versus empty sets

In EXPRESS, a set which may have no elements can be described by either

- `OPTIONAL SET [1:upper_limit] OF base_type`, or by
- `SET [0:upper_limit] OF base_type`.

However, the concepts modelled by the two descriptions are quite different. An `OPTIONAL SET` shows the fact that the set may or may not exist whereas a `SET[0 : upper_limit]` shows that the set always exists but it may be empty. The Core Model uses the former method.

3.5 Model topology

The topology is the graph representation of the relationships between the concepts defined in the model. Two possible approaches have been considered.

- A tree representation in which there is a concept (the root of the tree) that is a generalization of all the other concepts defined in the model. An example of Information Model that uses this topology is the CFI DR IM.
- A forest representation in which there are several categories of concepts that do not have any common features. An example of Information Model that uses this topology is the EDIF IM.

Withdrawn
IECNORM.COM : Click to view the full PDF of IEC 62016:2003

The forest representation has been chosen for the Core Model. An advantage of this representation (as opposed to the tree representation) is that it reduces the maximum depth of the leaves in the model hierarchy.

4 Concepts

This chapter describes the fundamental concepts defined by the Core Model. These concepts may be thought of as creating a hierarchy which is abstract at its higher level and becomes progressively more detailed. At the highest level of the structure, an *information_base* may contain *designs* and *libraries*. Each *library* is a collection of *cells* which are grouped according to a set of common characteristics. A *cell* is the main design object which may be instantiated later in another *cell*, thus creating a design hierarchy. A *cell* may be connected to other *cells* by means of its interface. These concepts are described in the following sections.

4.1 The information base

An *information_base* describes the data that can be found in the database of a compliant CAD system. This information may include *libraries*, *designs*, *global_ports* and *global_port_bundles* and *unit* definitions. The Core Model is an information model that describes one information base. All the entities defined in the Core Model belong, directly or indirectly, to the *information_base* object. The model of the *information_base* entity can be found in the *information_base_model* schema. A partial EXPRESS-G diagram is also available.

4.2 Global ports and global port bundles

Certain ports such as GND, VCC or clocks are used in common by several modules of an electronic circuit. The Core Model describes this by using the concept of global port, which is modelled by the *global_port* entity. The *global_ports* are defined in the *information_base* and are, therefore, visible in all the *cells* and *designs* defined in this information base.

It is sometimes convenient to address several ports as if they were a single port. In order to do this the Core Model uses the concept of port bundle. A port bundle provides a means of creating a structured port by grouping other ports and/or port bundles defined in the same context. In the case of *global_ports*, the port bundle is modelled by the *global_port_bundle* entity.

Both the *global_port* and the *global_port_bundle* are described in the *information_base_model* schema. A partial EXPRESS-G diagram is also available.

4.3 Libraries

A *library* provides a means of grouping design units (*cell* definitions). In future versions of the Core Model a *library* will contain other classes of reusable objects, as well.

A *library* may be locally defined, in which case the definition of the contained *cells* is included in the information base. *Libraries* may also be externally defined, which implies that the details of the *cell* definitions are not available within the information base.

The *library_model* schema describes the model for the *library* entity. A partial EXPRESS-G diagram is also available. According to where the contained data can be found, the library entity is subtyped into *internal_library* and *external_library*. The *internal_library* corresponds to the case where the library is locally defined. Therefore, since all the data is present in the *information_base*, an *internal_library* may contain implementation information. An *external_library* corresponds to the case where the library data is not contained in the *information_base* and, therefore, it does not include implementation details.

Withdrawn
IECNORM.COM : Click to view the full PDF of IEC 62016:2003

4.4 Cells

A *cell* is the basic design unit described by the Core Model. In order to be able to represent complex circuits, a *cell* can be divided into other *cells*, thus creating a hierarchy. The model of the *cell* entity is given in the *hierarchy_model* schema. A partial EXPRESS-G *cell* entity is subtyped into *internal_cell* and *external_cell* according to whether it describes information defined locally or not. An *internal_cell* contains locally defined data and may, therefore, contain implementation information. An *external_cell* does not include implementation details. Since an *internal_cell* contains data defined in the current *information_base*, it can only belong to an *internal_library*. Similarly, an *external_cell* belongs to an *external_library*.

4.5 Clusters and cell representation sets

A *cell* is described by means of *cell_representations*. *Cell_representations* that share the same interface can be grouped into *clusters*. A *cell* can contain more than one *cluster*. A *cluster* may be instantiated in another cell view, thus creating a design hierarchy. A *cluster* is subtyped into *internal_cluster* and *external_cluster* which are used by *internal_cell* and *external_cell* respectively. An *internal_cluster* may only contain *internal_cell_representations*, and an *external_cluster* only *external_cell_representations*. The model of the *cluster* entity is given in the *hierarchy_model* schema. A partial EXPRESS-G diagram is also available.

Cell representation sets are used to group *cell_representations* of the same *cell* that have a close relationship based on reasons other than a common interface. Simple annotation is provided within the *cell representation set* in order to specify the reason for the grouping. The model for the *cell_representation_set* entity can be found in the *hierarchy_model* schema.

IECNORM.COM : Click to view the full PDF of IEC 62016:2003

4.6 Cell representations

A *cell_representation* describes chosen aspects of a *cell*. At present, the Core Model supports only one type of *cell_representation*, the connectivity view. Future versions of the model will be extended, however, with other view types such as schematic and pcblayout, as well as with symbols. A *cell_representation* may be either an *internal_cell_representation* or an *external_cell_representation*. An *internal_cell_representation* belongs to an *internal_cluster* and describes an *internal_cell*. Similarly, an *external_cell_representation* belongs to an *external_cluster* and describes an *external_cell*. An *internal_cell_representation* can establish a direct relationship with other *cell_representations* in order to provide versioning and data management control information. Hence, an *internal_cell_representation* may be a new version of another *cell_representation* of the same type or may be derived from another *cell_representation*. If the *internal_cell_representation* is derived from another *cell_representation*, their type can be different. The interface of a *cell_representation* is obtained from its containing *cluster*.

The model of the *cell_representation* entity can be found in the *hierarchy_model* schema. A partial EXPRESS-G *internal_connectivity_view* and *external_connectivity_view* are given in the *connectivity_view_model* schema.

4.7 Master ports and master port bundles

The interface of a *cluster* consists of ports and port bundles. These objects represent the only places where a connection can be made to a *cell*. The interface of a *cluster* is described by the *cluster_interface* entity.

In the Core Model, each port in the cluster interface is described by a *master_logical_port* entity which describes its logical connectivity information. A *master_logical_port* can be associated with one or several *master_structure_ports* which describe the structural connectivity information of the port. It is possible that a *master_logical_port* is not associated with any *master_structure_port*.

The size of the *master_logical_port* and *master_structure_port* is always one. The *master_logical_port* entity is subtyped into *input_master_logical_port*, *output_master_logical_port*, *bidirectional_master_logical_port* and *unspecified_direction_master_logical_port*, which correspond to the possible directions of the port. In addition, a port contains information such as its designator, its properties and its external load capacitance. It is possible to specify a *global_port* as the default connection for a *master_logical_port*.

It is sometimes convenient to address several ports as if they were a single port. In order to be able to do this, a grouping method is provided. The Core Model uses the concept of port bundle in order to describe a structured port. A port bundle in the cluster interface is described by a *master_logical_port_bundle* entity, which groups *master_logical_ports* and/or other *master_logical_port_bundles* defined in the same interface. The same *master_logical_port* may be referenced by more than one *master_logical_port_bundle*. However, it cannot be referenced more than once by the same *master_logical_port_bundle*, either directly or indirectly.

The availability of a *master_logical_port_bundle* for structural connectivity is indicated by the *master_structure_port_bundle* entity. A *master_logical_port_bundle* can be associated with more than one *master_structure_port_bundle*.

A structured port can be defined in the interface of a *cluster* by using the *port_structure* entity. A *port_structure* describes a possible structuring of the structure ports and port bundles of a cluster interface. It can represent either an ordered or an unordered structured port.

The models for the *master_logical_port*, *master_structure_port*, *master_logical_port_bundle*, *master_structure_port_bundle* and *port_structure* entities are given in the *hierarchy_model* schema. Partial EXPRESS-G diagrams for *cluster_interface* and *port_structure* are also available.

4.8 Instances

A *cluster* may be instantiated in a cell view, thus enabling the creation of a design hierarchy. The *instance* entity models the instance of a *cluster*. The interface of the *instance* consists of *instance_structure_ports* and *instance_structure_port_bundles* which must reference *master_structure_ports* and *master_structure_port_bundles* defined in the interface of the instantiated *cluster*. If the width of the *instance* is greater than one, the *instance* represents an arrayed *instance*.

The advantage of the instantiation mechanism is that the choice of a particular *cell_representation* can be delayed until the complete *design* is configured. For example, a connectivity view may contain *instances* of *clusters* of other *cells*. When the connectivity view is configured, *instances* are configured by selecting suitable views.

An *internal_connectivity_view* can define, for each of its instances, a *connectivity_instance_implementation* which describes the structural information associated with an *instance*. Since the only type of *cell_representation* defined in the current model is the connectivity view, the only significant structural information is the port structuring.

The model for the *instance* entity can be found in the *hierarchy_model* schema. A partial EXPRESS-G diagram is also available.

4.9 Instance ports and instance port bundles

The interface of the *instance* consists of *instance_structure_ports* and *instance_structure_port_bundles*. The *instance_structure_port* is defined by either a *connectivity_generic_bus* or by a *connectivity_generic_net* and associates a *master_structure_port* with an *instance*. The information contained in the *master_logical_port* associated with the referenced *master_structure_port* may be overridden by the *instance* if it defines an *instance_port_attributes* entity associated with the *master_logical_port*.

IECNORM.COM: Click to view the full PDF of IEC 62016:2003 Withdrawn

The *instance_port_attributes* entity is subtyped into *input_instance_port_attributes*, *output_instance_port_attributes*, *bidirectional_instance_port_attributes* and *unspecified_ direction_instance_port_attributes*, corresponding to the possible types of the associated *master_logical_port*.

An *instance_structure_port_bundle* entity references a *master_structure_port_bundle* which must be defined in the interface of the instantiated *cluster*. The *instance* may override some of the information contained in the *master_logical_port_bundle* associated with the referenced *master_structure_port_bundle* by defining an *instance_port_bundle_attributes* entity.

The Core Model defines the *instance* entity to be an indexed collection of *cluster* instances. The number of elements in the collection is given by the "width" attribute in the *instance* entity. A port in the interface of a member of the *instance* is modelled by the *instance_member_ logical_port* and *instance_member_structure_port* entities.

The models for the *instance_structure_port*, *instance_member_logical_port*, *instance_ member_structure_port*, *instance_structure_port_bundle*, *instance_port_attributes* and *instance_port_bundle_attributes* entities are given in the *hierarchy_model* schema.

IECNORM.COM: Click to view the full PDF of IEC 62016:2003
Withdrawn

5 Connectivity

The only view type supported by the model at present is the connectivity view. Future versions of the model may contain other views that support connectivity definitions such as schematic and pcblayout. The Core Model differentiates between two levels of connectivity information within the connectivity view.

- Logical connectivity information

  The logical connectivity describes the bit level, abstract electrical connectivity for a given level of a hierarchy, in terms of signals and signal bundles. Every type of view that supports connectivity definitions contains similar connectivity information.

- Structural connectivity information

  The structural connectivity describes the connectivity, for a given level of a hierarchy, from the structural point of view. The structural connectivity is specified in terms of buses, nets and rippers. Such a structural representation is used to provide support for physical implementation and annotation. The structural connectivity information is view-type specific. Indeed, the manner in which the connectivity view is structured is unlikely to be the same as that of a future schematic or pcb view.

5.1 Logical connectivity

The model for the logical connectivity can be found in the *logical_connectivity_model* schema. A partial EXPRESS-G diagram is also available. The logical connectivity of a view is described in terms of *signals* and *signal_bundles*.

5.1.1 Signals

A *signal* is defined as an object that provides a means by which all the *global_ports*, *instance_member_logical_ports* and *master_logical_ports* may be logically electrically common at a a given level of hierarchy, and to enable the logical flowing, sharing and exchanging of information and energy between the applicable blocks.

A *signal* is always one bit wide and is defined within an *internal_connectivity_view*. All *global_ports* joined by a *signal* are defined in the containing *information_base*. All *instance_member_logical_ports* joined by a *signal* must reference *instances* in the containing view. All *master_logical_ports* joined by a *signal* are defined in the interface of the containing *cluster*.

5.1.2 Signal_bundle

A *signal_bundle* provides a grouping mechanism for *signals*. A *signal_bundle* is defined within a view and is used to support structural connectivity. However, it does not provide additional connectivity information. A *signal_bundle* is specified as an ordered list of one or more *signals* or *signal_bundles* in the same containing view. The structure of a *signal_bundle* is independent of port ordering or port structure. No two *signal_bundles* may have the same structure of members. The same *signal* may be referenced by more than one *signal_bundle* and may be referenced more than once in the same *signal_bundle*. Since a *signal_bundle* may contain *signals* and *signal_bundles*, it creates a signal hierarchy as shown in the example in Figure 5.

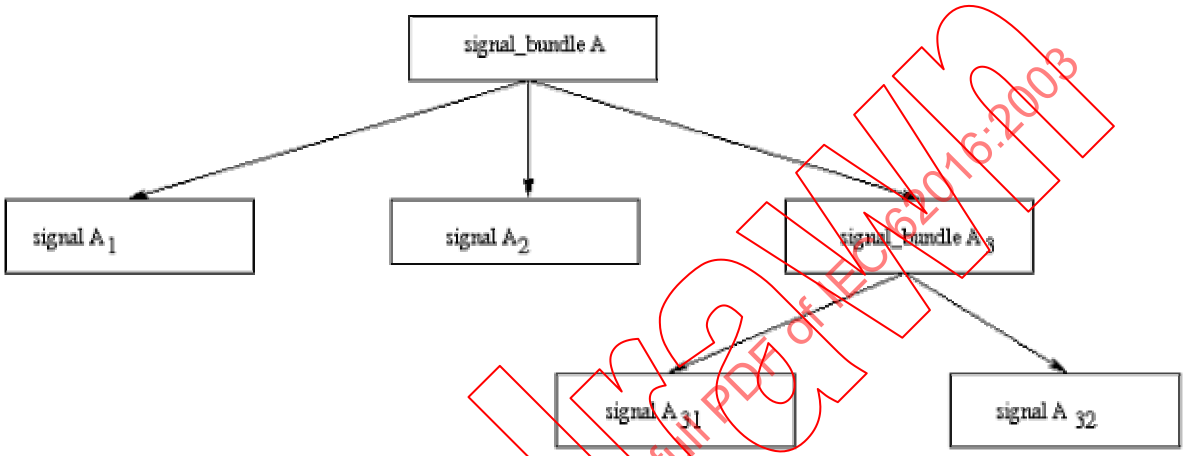


Figure 5 – An example of signal hierarchy

Such a hierarchical structure is used to support the structural connectivity expressed by buses.

5.2 Structural connectivity with wide instances

When an interconnect (net or bus) joins a port or a port bundle on a wide *instance*, the pattern of connectivity will be either in the commoned style or in the fanned-out style. For the commoned style, the joined *instance_structure_ports* or *instance_structure_port_bundles* reference *master_structure_ports* or *master_structure_port_bundles* whose size is the same as that of the interconnect. For the fanned-out style, the joined *instance_structure_ports* are flattened to lists of *instance_member_structure_ports*. The length of these lists is equal to the size of the interconnect. In the case of *instance_structure_port_bundles*, their size is equal to the size of the *master_logical_port_bundle* associated with the *master_structure_port_bundle* referenced by the *instance_structure_port_bundle*. The size of the *instance_structure_port_bundle* must be equal to the size of the joined interconnect (whose size is equal to the size of its associated *signal* or *signal_bundle*). A net is associated with a *signal* and, therefore, its size is always one. A bus is associated with a *signal_bundle* and, therefore, its size is equal to the length of the list created by flattening the *signal_bundle*. Several examples are given below.

5.2.1 Joining ports on instances in the commoned style

Figure 6 presents an example of a net joining a port on a wide *instance*, using the commoned style. The wide *instance* "I" has a width of three. The instantiated *cluster* defines a *master_structure_port* 'P' in the *cluster_interface*.

IECNORM.COM : Click to view the full PDF of IEC 62016:2003
Withdrawn

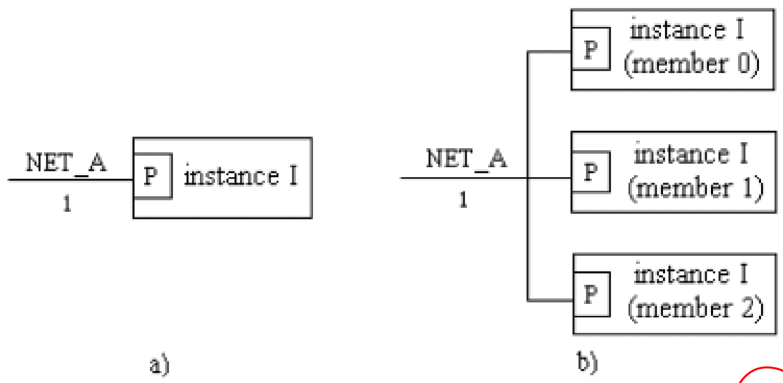


Figure 6 – A net joins a port on an instance in the commoned style

If the net “NET_A” joins port “P” in *instance* “I” as shown in Figure 6a, the underlying connectivity of “NET_A” will be equivalent to joining port “P” on the 0th member of *instance* “I”, port “P” on the 1st member of *instance* “I” and port “P” on the 2nd member of *instance* “I”, as shown in Figure 6b.

5.2.2 Joining port bundles on instances in the commoned style

Figure 7 shows an example of a bus joining a port bundle on a *instance* using the commoned style. The wide *instance* “I” has a width of three. The interface of the instantiated *cluster* contains two *master_structure_ports* “P1”, “P2” and a *master_structure_port_bundle* “PB” which groups “P1” and “P2”.

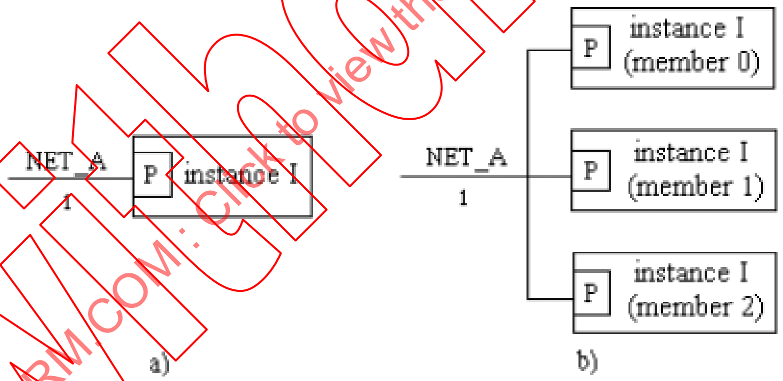


Figure 7 – A bus joins a port bundle on an instance in the commoned style

IECNORM.COM : Click to view the full PDF of IEC 62016:2003
Withdrawn

When a bus joins a port bundle on an *instance*, using the commoned style, the joined *instance_structure_port_bundle* must reference a *master_structure_port_bundle* whose size is the same as that of the bus. The size of the bus is determined by flattening its associated *signal_bundle*. In the example in Figure 7a, a bus “BUS_A” joins a port bundle “PB” on an *instance* “I”. The size of “PB” is the same as the size of “BUS_A”. This is equivalent to the situation presented in Figure 7b where the 0th bit of “BUS_A” joins port “P1” on the 0th member of *instance* “I”, port “P1” on the 1st member of *instance* “I” and port “P1” on the 2nd member of *instance* “I”, and the 1st bit of “BUS_A” joins port “P2” on the 0th member of *instance* “I”, port “P2” on the 1st member of *instance* “I” and port “P2” on the 2nd member of *instance* “I”.

5.2.3 Joining port bundles on instances in the fanned-out style

Figure 8 shows an example of a bus joining a port bundle on an *instance*, using the fanned-out style. The wide *instance* “I” has a width of three. The interface of the instantiated *cluster* defines two *master_structure_ports* “P1”, “P2” and a *master_structure_port_bundle* “PB” which groups “P” and “P2”.

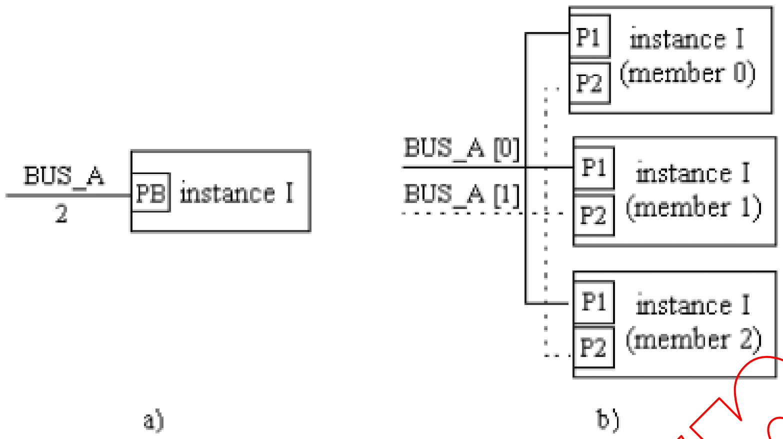


Figure 8 – A bus joins a port bundle on an instance in the fanned-out style

IECNORM.COM : Click to view the full PDF of IEC 62016:2003

When a bus joins a port bundle on an *instance*, using the fanned-out style, the size of port list resulting from expanding the *master_logical_port_bundle* associated with the *master_structure_port_bundle* referenced by the joined *instance_structure_port_bundle* must be the same as that of the bus. The size of the port list is the product of the *instance* width and the referenced *master_structure_port_bundle* size. In the example in Figure 8a, a bus “BUS_C” joins the port bundle “PB” on the *instance* “I”. “PB” implicitly represents a list of six ports (i.e. *instance* width × *master_structure_port_bundle* size = 3 × 2). The elements of the port bundle “PB” correspond to consecutive members of the list of ports. The size of bus “BUS_C” is also 6. The example is equivalent to the situation presented in Figure 8b where the 0th bit of the bus joins port “P1” on the 0th member of *instance* “I”, the 1st bit joins port “P2” on the 0th member of *instance* “I”, the 2nd bit joins port “P1” on the 1st member of *instance* “I”, the 3rd bit joins port “P2” on the 1st member of *instance* “I”, the 4th bit joins port “P1” on the 2nd member of *instance* “I” and the 5th bit joins port “P2” on the 2nd member of *instance* “I”.

5.3 Structural connectivity of connectivity views

The model of the structural connectivity is found in the *connectivity_structure_model* schema. The structural connectivity information is described by connectivity nets, connectivity buses and connectivity rippers. The structural representation is used to provide support for physical implementation and annotation.

5.3.1 Connectivity net and connectivity sub-net

A net is a structured representation of a *signal*. Each net is associated with a *signal* and may join single ports.

Sometimes it is necessary to divide a net into various parts so that properties may be added to one part but not to any of the others. Hence, a net may be divided into sub-nets which may be further subdivided into sub-nets. However, sub-nets do not give new connectivity information. The ports associated with a sub-net are referenced in the immediately containing net or sub-net. No two sub-nets at a given level refer to the same set of ports.

In the information model, the concept of net and sub-net are generalized into the concept of generic-net. A *connectivity_generic_net* is either a *connectivity_net* or a *connectivity_sub_net*. Each *connectivity_generic_net* is associated with a *signal*. It may join *global_ports*, *instance_member_structure_ports*, *instance_structure_ports*, *master_structure_ports* and *connectivity_rippers*. A *connectivity_generic_net* may contain *connectivity_sub_nets* which may be further divided into more *connectivity_sub_nets*.

A *connectivity_net* is defined within an *internal_connectivity_view*. Its associated *signal* and its joined *connectivity_rippers* are defined in the containing view. Furthermore, the joined *global_ports*, *instance_member_structure_ports* and *master_structure_ports* are mentioned in its associated *signal*. If a *connectivity_net* joins an *instance_structure_port*, then the list of *instance_member_structure_ports* joined implicitly is a subset of the *instance_member_structure_ports* joined by its associated *signal*.

A *connectivity_sub_net* is defined within another *connectivity_generic_net*. The associated *signal* of a *connectivity_sub_net* is the same as the one in its enclosing structure. Its joined ports or rippers are a subset of the ports or rippers referenced by its containing net.

A partial EXPRESS-G diagram of *connectivity_generic_net* and its subtypes is available.

5.3.2 Connectivity bus, connectivity bus slice and connectivity sub-bus

A bus is a structured representation of a collection of *signals*. Each bus is associated with a *signal_bundle* and may join wide ports.

A bus may be subdivided in two ways. It may be divided into bus-slices or into sub-buses. A bus-slice is used to define a part of a bus. Its associated *signal_bundle* is a member of the *signal_bundle* associated with its immediately containing bus. No two bus-slices at a given level refer to the same *signal_bundle*. A sub-bus is similar in nature to a sub-net in that it allows additional structural information to be given about a part of a bus. A sub-bus is associated with the same *signal_bundle* as that in the immediately containing bus. No two sub-buses at a given level reference the same set of ports or rippers.

In the information model, the concepts of bus, bus-slice and sub-bus are generalized into the concept of generic-bus. A *connectivity_generic_bus* is either a *connectivity_bus*, a *connectivity_bus_slice* or a *connectivity_sub_bus*. Each *connectivity_generic_bus* is associated with a *signal_bundle*. It may join *connectivity_rippers* and wide ports such as *global_port_bundles*, *local_master_port_bundles*, *instance_structure_port_bundles* and *master_structure_port_bundles*. It may be divided into *connectivity_bus_slices* or *connectivity_sub_buses* which may themselves be further subdivided.

All *global_port_bundles*, *local_master_port_bundles*, and *master_structure_port_bundles* joined by a *connectivity_generic_bus* have the same port size. This is the same as the size of the associated *signal_bundle*. Each of the logical ports associated with the ports of the joined port structures is referenced or defined in the corresponding signal in the flattened *signal_bundle*. If a generic-bus joins an *instance_structure_port_bundle*, the pattern of connectivity will be either in the commoned style or in the fanned-out style. For the commoned style, the joined *instance_structure_port_bundles* reference *master_structure_port_bundles* whose port size is the same as the size of the flattened *signal_bundle*. The *instance_structure_port_bundles* size is equal to the size of the *master_logical_port_bundle* associated with the *master_structure_port_bundle* referenced by the *instance_structure_port_bundle*. The size of the *instance_structure_port_bundle* must be equal to the size of the joined flattened *signal_bundle*.

A *connectivity_bus* is defined within an *internal_connectivity_view*. Its associated *signal_bundle* and its joined rippers are defined in the containing view.

A *connectivity_bus_slice* is defined within another *connectivity_generic_bus*, and its associated *signal_bundle* is a member of the *signal_bundle* associated with its enclosing structure.

A *connectivity_sub_bus* is defined within another *connectivity_generic_bus* and its associated *signal_bundle* is the same as that of its enclosing bus. Its joined rippers or ports are a subset of the rippers or ports referenced by the enclosing bus.

A partial EXPRESS-G diagram of *connectivity_generic_bus* and its subtypes is available.

IECNORM.COM : Click to view the full PDF of IEC 62016:2003

5.3.3 Connectivity ripper

A ripper associates a bus/bus-slice/sub-bus structure with other bus/bus-slice/sub-bus structures or with net/sub-net structures. Rippers can be explicitly joined by any number of net and bus structures, defined in the same view.

In the information model, the *connectivity_ripper* entity has two INVERSE attributes which identify the associated net and bus structures. All the related net or bus structures belong to the same view. Each structure is related to other structures by sharing at least one common *signal*.

6 The design hierarchy mechanism

The Core Model contains a design hierarchy mechanism which is similar to the EDIF design hierarchy and includes some of the capabilities given by the CFI DR view selection.

The need for a design hierarchy mechanism derives from the fact that a *cluster* contains one interface but may have multiple views. Therefore, in order to specify an unambiguous tree structure for a given design, it is necessary to be able to specify for each *instance* which view is to be used. Furthermore, it is necessary to be able to specify alternative configurations for *instances* so that different design trees may be represented.

In addition to choosing specific configurations of specific views it may also be necessary to specify that, in a particular configuration, a given *instance* may be a leaf and use none of the views of its cluster. It is also necessary to specify whether the configured *instance* belongs to an *internal_cell* or to an *external_cell*. Finally, not all configuration information may be available at a given moment. Therefore a mechanism for explicitly having unconfigured *instances* is needed.

6.1 The design hierarchy

A *design* is a set of hierarchies of *cell_representations*. Each hierarchy unambiguously describes an electrical circuit. The hierarchy is modelled by the *design_hierarchy* entity. A design hierarchy contains the description (annotation) of the top-level node of the hierarchy and an associated cluster configuration. A *design* may contain many design hierarchies. For each *design_hierarchy*, the chosen *cluster_configuration* is defined in the *cluster_interface* of the top *cell* of the *design*. According to the information contained in the top level, a *design_hierarchy* is subtyped into

- *external_design_hierarchy*. This corresponds to the case where the top level selects an *external_cluster_configuration*, which means that the actual *cell* information is external to the information base. According to the information that is available, the *external_design_hierarchy* is further sub-typed into
  – *expandable_external_design_hierarchy*. This corresponds to the case where an *expandable_external_cluster_configuration* is selected. The expandable external cluster configuration selects an *external_cell_representation* as the top view of the hierarchy. Although the information is external and, therefore, does not allow any further configuration, the top level may, potentially, be further expanded.
  – *leaf_external_design_hierarchy*. This corresponds to the case where the design hierarchy selects a *leaf_external_cluster_configuration*. The leaf external cluster configuration specifies that there is no further expansion of the hierarchy. Therefore, this design hierarchy corresponds to the case where the design consists of only one cell which is described externally.
- *internal_design_hierarchy*. This corresponds to the case where the top level selects an *internal_cluster_configuration*. According to the information contained in the cluster configuration, the design hierarchy may be further subdivided into

IECNORM.COM: Click to view the full PDF of IEC 62016:2003
Withdrawn

– *expandable_internal_design_hierarchy*. This corresponds to the case where it selects an *expandable_internal_cluster_configuration*. The expandable internal cluster configuration selects an *internal_cell_representation* as the top view of the hierarchy. Some of the instances contained in the top view may have associated instance configurations that are contained in the *expandable_internal_cluster_configuration*. Each of these *instance_configurations* selects another *cluster_configuration*, thus elaborating the hierarchy.

– *expandable_internal_connectivity_design_hierarchy*. This corresponds to the case where the design hierarchy selects an *expandable_internal_connectivity_view_configuration* which provides information for the expansion of a hierarchy that starts with an *internal_connectivity_view*.

– *leaf_internal_design_hierarchy*. This corresponds to the case where the design hierarchy selects a *leaf_internal_cluster_configuration*. The leaf internal cluster configuration specifies that there is no further expansion of the hierarchy. Therefore, this design hierarchy corresponds to the case where the *design* consists of only one *cell* which is described in the information base.

A partial EXPRESS-G diagram of *design* is available.

The *design_hierarchy* contains the chosen *cluster_configuration* for the top *cell*. The role of the *cluster_configuration* entity is to configure a *cluster* by providing additional information and to specify how the hierarchy is to be continued. The *cluster_configuration* is subtyped into:

- *external_cluster_configuration*. This is further subtyped into
  - *expandable_external_cluster_configuration*
  - *leaf_external_cluster_configuration*
- *internal_cluster_configuration* . This is further subtyped into
  - *expandable_internal_cluster_configuration*
  - *expandable_internal_connectivity_view_configuration*
  - *leaf_internal_cluster_configuration*

A partial EXPRESS-G diagram of *cluster_configuration* and its subtypes is available.

An expandable cluster configuration identifies a view for the further expansion of the hierarchy. The chosen view belongs to the *cluster* that contains the expandable cluster configuration.

In the case of *expandable_internal_cluster_configuration*, each *instance* defined within the chosen view is either configured or explicitly unconfigured. In order to configure an *instance*, it is necessary to specify an *instance_configuration*. An *instance_configuration* selects a valid *cluster_configuration* for the configured *instance*. The chosen *cluster_configuration* is defined in the instantiated *cluster* of the configured *instance*. To unconfigure an *instance*, it is necessary to state explicitly that the *instance* is required to be unconfigured. Thus, an *expandable_internal_cluster_configuration* may contain several *unconfigured_instances* and several *instance_configurations*. If an instance has an *instance_configuration* which references a leaf configuration, there is no further structure in the design tree below that instance. An *expandable_internal_connectivity_view_configuration* gives information for the expansion of a hierarchy that has a connectivity view *cell_representation* at the top level.

A partial EXPRESS-G diagram of *instance_configuration* and its subtypes is available.

In a given expansion, an *instance* is configured at most once. An expandable cluster configuration may also create a scope for global ports, at the corresponding level in the hierarchy.

IECNORM.COM : Click to view the full PDF of IEC 62016:2003

6.2 Annotations

The top level occurrence of a *design_hierarchy* is annotated by an *occurrence_hierarchy_annotate* entity. Master ports, interconnects and signals defined within the top view may also be annotated. A *master_port_annotate* attaches or modifies attributes and properties which are associated with a *master_logical_port* occurrence in the top-level occurrence. An *interconnect_annotate* associates physical characteristics with an interconnect structure within an occurrence hierarchy. A *signal_annotate* attaches or modifies properties which are associated with a *signal* occurrence within an occurrence hierarchy.

The *occurrence_hierarchy_annotate* entity is subtyped into

- *external_occurrence_hierarchy_annotate*. This is further subtyped into
  - *expandable_external_occurrence_hierarchy_annotate*.

    This corresponds to the case where the annotate belongs to an *expandable_external_design_hierarchy*.
  - *leaf_external_occurrence_hierarchy_annotate*.

    This corresponds to the case where the annotate belongs to a *leaf_external_design_hierarchy*.
- *internal_occurrence_hierarchy_annotate*. This is further subtyped into
  - *expandable_internal_occurrence_hierarchy_annotate*.

    This corresponds to the case where the annotate belongs to an *expandable_internal_design_hierarchy*.
  - *expandable_internal_connectivity_occurrence_hierarchy_annotate*.

    This corresponds to the case where the annotate belongs to an *expandable_internal_connectivity_design_hierarchy*.
  - *leaf_internal_occurrence_hierarchy_annotate*.

    This corresponds to the case where the annotate belongs to a *leaf_internal_design_hierarchy*.

A partial EXPRESS-G diagram of *occurrence_hierarchy_annotates* and its subtypes is available. An *expandable_internal_design_hierarchy* is the only truly expandable hierarchy since it contains information that is locally defined. However, only the *expandable_internal_connectivity_design_hierarchy* can be annotated since the type of the annotated view is known. Therefore, its associated *expandable_internal_connectivity_occurrence_hierarchy_annotate* may contain *occurrence_annotates*. They are used to provide annotation information for all the nodes in a design hierarchy which are not top level nodes.

The *occurrence_annotate* entity is subtyped into

- *external_occurrence_annotate*. This is further subtyped into
  - *expandable_external_occurrence_annotate*
  - *leaf_external_occurrence_annotate.*
- *internal_occurrence_annotate*. This is further subtyped into
  - *expandable_internal_occurrence_annotate*
  - *expandable_internal_connectivity_occurrence_annotate*
  - *leaf_internal_occurrence_annotate.*

Partial EXPRESS-G diagrams of *occurrence_annotate* and its subtypes are available.

IECNORM.COM : Click to view the full PDF of IEC 62016:2003
Withdrawn

The *occurrence_annotate* selects the annotated occurrence by referencing an *instance_ configuration* which specifies the annotated instance and the chosen *cluster_configuration* for that annotated instance. Instance ports on the annotated instance may be annotated by *instance_port_annotates*. An *instance_port_annotate* is used to attach or modify properties and attributes which are associated with an *instance_structure_port* occurrence within an occurrence hierarchy. In an *expandable_internal_occurrence_annotate* interconnects and signals defined within the chosen view may be annotated by *interconnect_annotates* and *signal_annotates* respectively.

Any leaf level occurrence of a *design_hierarchy* is annotated by a leaf occurrence annotate. A leaf level occurrence is selected by referencing an *instance_configuration* which points to a leaf cluster configuration.

7 Core Model for electronic design

The Core Model for electronic design describes the concepts which are necessary for the definition of bit-level electronic designs. These concepts include

- libraries
- re-usable design units (cells)
- cell definition hierarchies
- connectivity
- designs

7.1 Libraries

Within the Core Model, the units of designs, called cells, are grouped into libraries. All of the cells in a library are considered by the originating system to be related in some way . There are no pre-defined semantics to such groupings within the Core Model. An information base may contain one or more libraries.

Where a library, and the cells within it, are fully specified within the information base, the library and its contained objects are said to be "internal".

Alternatively, if the contents of the library are known to all systems accessing the information base, then only a minimal amount of information needs to be specified in the information base. Such a library and its contained objects are termed "external". The external library specifications within the information base can be considered as stubs or handles for the actual library which exists outside the information base.

Withdrawn
IECNORM.COM : Click to view the full PDF of IEC 62016:2003

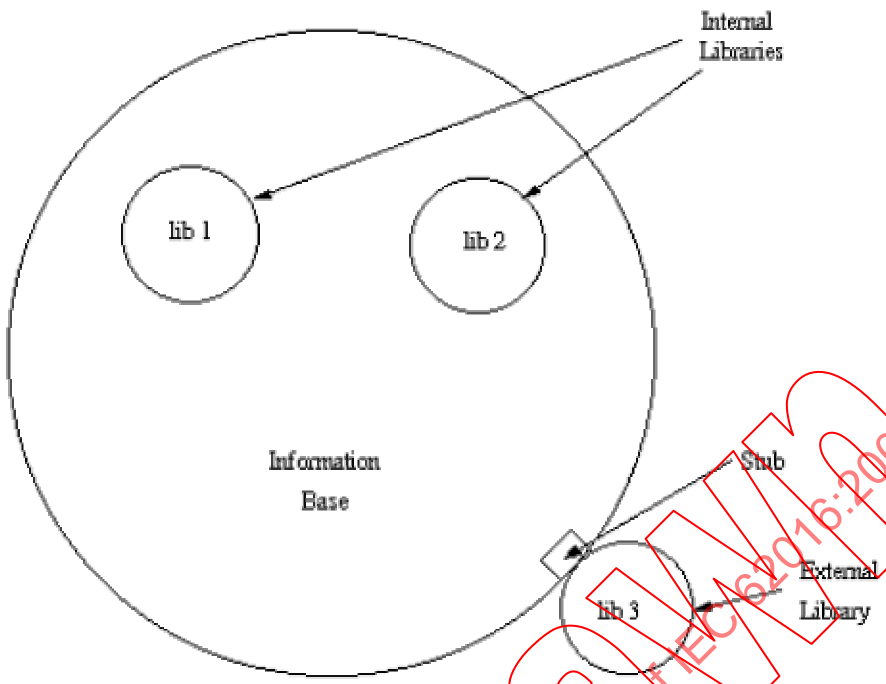


Figure 9 – Internal and external libraries

IECNORM.COM: Click to view the full PDF of IEC 62016:2003
Withdrawn

In the case of an external library, the only information which is specified about the library and its contents is that necessary to form relationships to the objects within it. For example, it is necessary to know the ports of objects within the external library so that connections can be made to those objects. However, details about the connectivity within the library's objects are omitted since they are assumed to be specified in the outside library.

7.2 Interfacing to cells

Within design systems, modules or gates can be connected together by specifying connections between their ports or pins. In addition, it is possible for different descriptions of the same gate to have different sets of pins. For example, a description of a two-input NAND gate could have three ports (two input and one output), or five (two input, one output, power and ground).

The concept of a port is considered to be split into master logical and master structure ports. A master structure port makes a master logical port available for use within structural connectivity.

Within the Core Model, the manner in which a cell can interface to other cells is specified by a set of master logical ports. Each master logical port can carry a single-bit of data. These master logical ports are grouped into a cluster interface. Different descriptions, termed cell_representations, of the cell which share the same set of master logical ports may be grouped into clusters.

The single-bit master logical ports can be grouped into master logical port bundles. A master logical port bundle references the list of master logical ports and/or other master logical port bundles which are within it. These master logical port bundles may be made available for wide-connections via a referencing master structure port bundle.

For example, the inputs to an adder could be considered as a single master structure port bundle as shown below. This master structure port bundle makes available the master logical port bundle which consists of a single carry in bit and two eight-bit wide master logical port bundles representing the numbers to be added. Each of the eight-bit master logical port bundles consists of eight single-bit master logical ports.

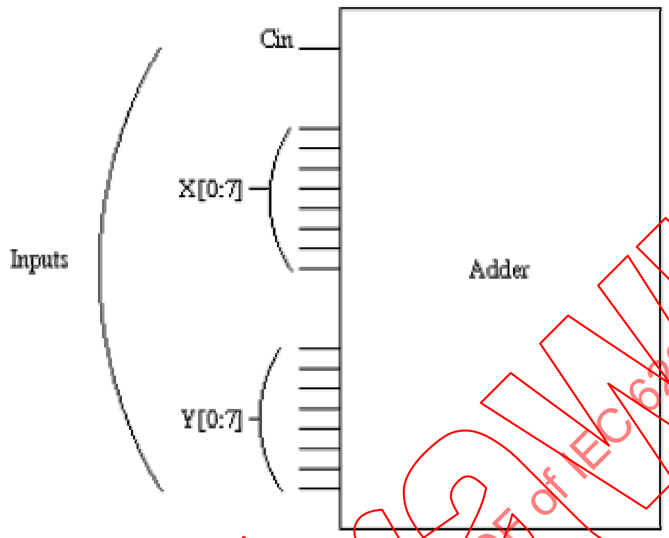


Figure 10 – Port bundling – 1

As may be expected from the above example, a master logical port may only be a member once of a given bundle.

For a given design unit, there may be several ways in which the same master logical ports may be grouped for different purposes. For example, it may be reasonable to group the two bit[0]s of the input numbers, the two bit[1]s and so on.

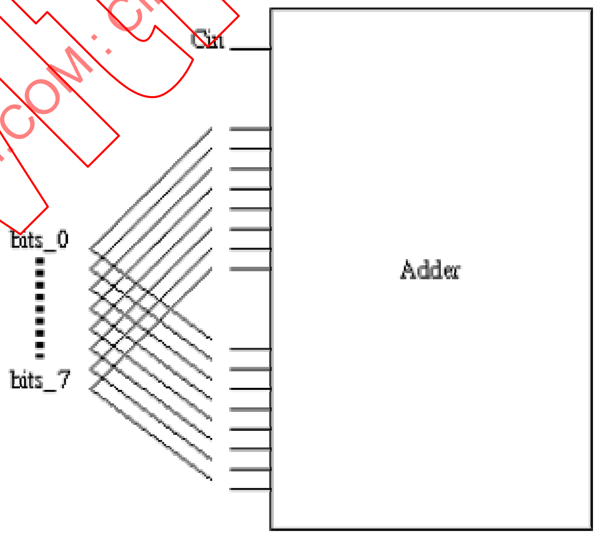


Figure 11 – Port bundling – 2

In order to support this alternative, there would be eight master structure port bundles each corresponding to two master logical ports.

Withdrawn
IECNORM.COM : Click to view the full PDF of IEC 62016:2003

The two groupings of ports described above are mutually exclusive i.e. they cannot be used for the same cell at the same time. For this reason, each set of master structure ports and master structure port bundle is specified within a different port structure. So the two example groupings of ports would be specified in two separate port structures. A port structure may additionally specify the order in which the originating system considers the ports to be structured.

7.3 Cell definition hierarchies

A cell can specify different possible realizations or implementations of its logic. In the current version of the Core Model, the only possible realization of an internal cell is as a connectivity view.

7.4 Instantiation

An internal connectivity view can be made up of sub-units. Within the Core Model, these sub-units are specified as instances of other cells. An instance may be viewed as an indication that one or more occurrences of the instantiated cell will be included in the overall design for each instance within an occurrence of the connectivity view. The number of occurrences of the instantiated cell is specified by the instance width.

For example, in the figure below, the definition of cell C includes a three-wide instance of cell A. Since cell C is instantiated by a five-wide instance (i3) within cell D, the top-level design, within the design there are 21 occurrences of cell A. Similarly, there are seven occurrences of cell B.

Withdrawn
IECNORM.COM : Click to view the full PDF of IEC 62016:2003

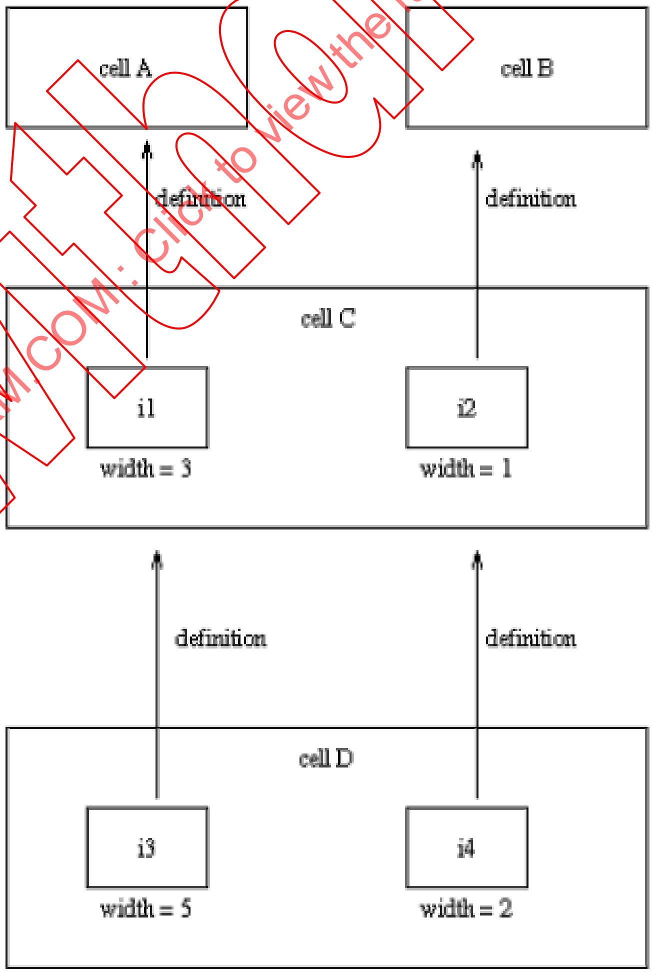


Figure 12 – Instantiation

In order for an instance to be sensibly used within a connectivity view, it is necessary to chose a particular logical interface of the instantiated cell, i.e. the instance identifies a particular cluster of the instantiated cell.

Attributes of the instantiated cluster and its logical ports may be overridden or added to within the instance. These changes only apply to the instance and corresponding occurrences i.e. they alter the copies of the instantiated cell not the instantiated cell itself.

7.5 Logical connectivity

Within a connectivity view, the bit-wide logical interconnection of the logical ports of the view and of any instances within it, is specified by signals. A signal within the Core Model just indicates a bit-wide connection. In particular, the scope of this Core Model definition does not

- indicate any value changes on the connection;
- allow the specification of type information about the data carried on the signal.

A signal may connect logical ports of the connectivity view. These are termed master logical ports. In strict terms, a cluster interface specifies master logical port(bundle)s and master structure port(bundle)s. A signal may also connect up to logical ports of instances within the connectivity view. Since an instance may represent more than one occurrence, it is necessary for the signal to specify which basic member of the instance the port interfaces to. This complex reference to a logical port of a member of the instance is termed an instance member logical port.

Signals may be grouped into signal bundles, in a similar manner to that described above for logical ports and logical port bundles. The instances, signals and signal bundles together specify the logical connectivity of the connectivity view.

7.6 Structural connectivity

The logical connectivity is sufficient to describe the connections between the master logical ports of the connectivity view and the instance member logical ports of the instances. However, sometimes systems describe details about the manner in which the logical signal should be realized. Such implementation information includes the specification that a given signal is realized by several sub-connections with different properties as shown below. The structured realization of a signal is termed a net.

IECNORM.COM : Click to view the full PDF of IEC 62016:2003
Withdrawn

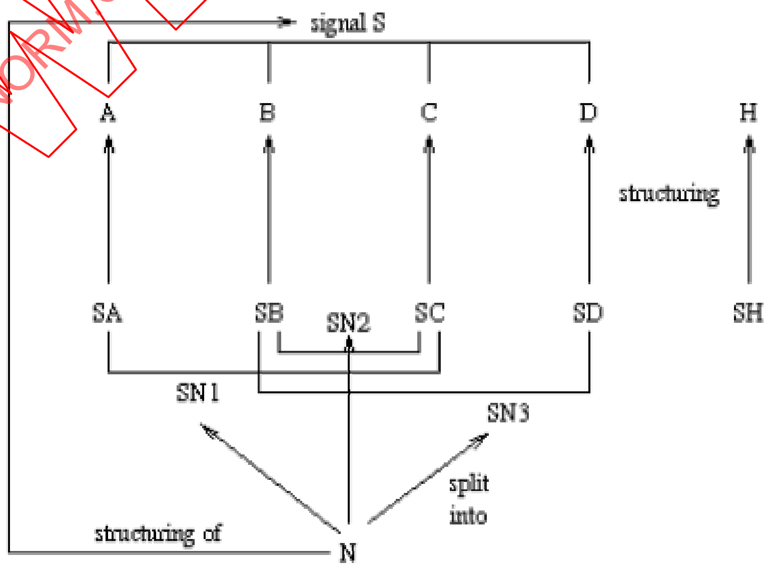


Figure 13 – Connectivity net

In the above figure, the logical signal S connecting together the logical port set [A, B, C, D] is realized by the net N which is split into three sub-nets, SN1, SN2 and SN3 connecting the structure ports {SA,SC}, {SC, SB} and {SB, SD} respectively.

As may be expected, there are constraints to ensure that the structure ports connected by N structure a subset of the logical ports joined by the signal S, i.e. N could not join to SH.

Groups of signals may also be realized by a single object termed a connectivity bus. Since the connectivity bus is wide, it connects together wide structured objects, i.e. master structure port bundles and instance structure port bundles. In addition, there is the capability within the Core Model for a connectivity view to specify *ad hoc* groupings of master ports, termed local master port bundles, which can also be connected by buses.

The simplest case of bus connectivity is where the joined objects are of the same size as the bus as shown below.

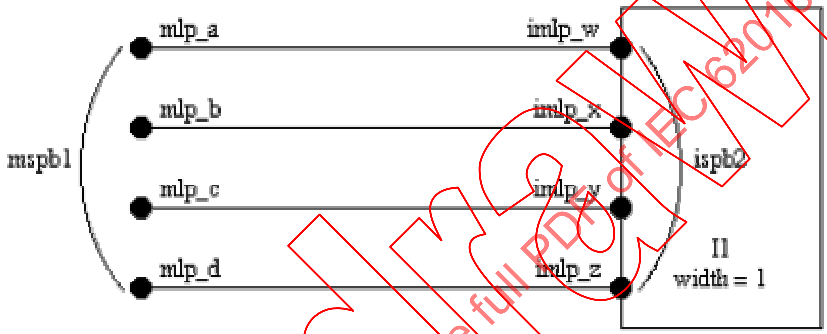


Figure 14 – Connectivity bus

IECNORM.COM : Click to view the full PDF of IEC 62016:2003

The master structure port bundle, mspb1 is joined by the bus to the instance structure port bundle of instance I1. In order for this to be a valid bus, master logical port mlp_a must be connected by a signal to instance member logical port imlp_w, mlp_b to imlp_x etc. It should be emphasized that structural connectivity never establishes “connectedness”; it structures the logical connectivity specified by signals.

When a bus connects to an instance structure port bundle, there are two ways in which the signals of the signal bundle which the bus realizes may connect up to the individual ports of the instance structure port bundle. The size of the port bundle may equal that of the bus, in which case the first element of the port bundles are connected together as shown below. This type of connection is termed commoning because the corresponding members of the port bundles of the instance members are connected.

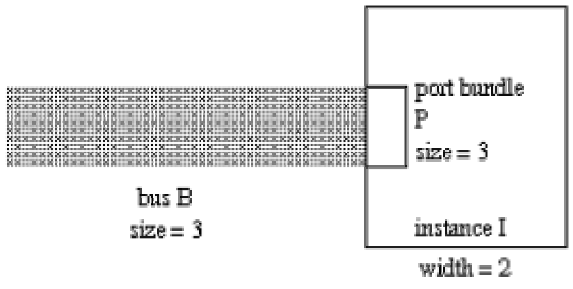


Figure 15 – Connectivity bus – Commoning

In the above example, the first signal associated with bus B is connected to the first element of port bundle P of each member of instance I, i.e. since the width of instance I is 2, the signal is connected to 2 instance member logical ports as shown below.

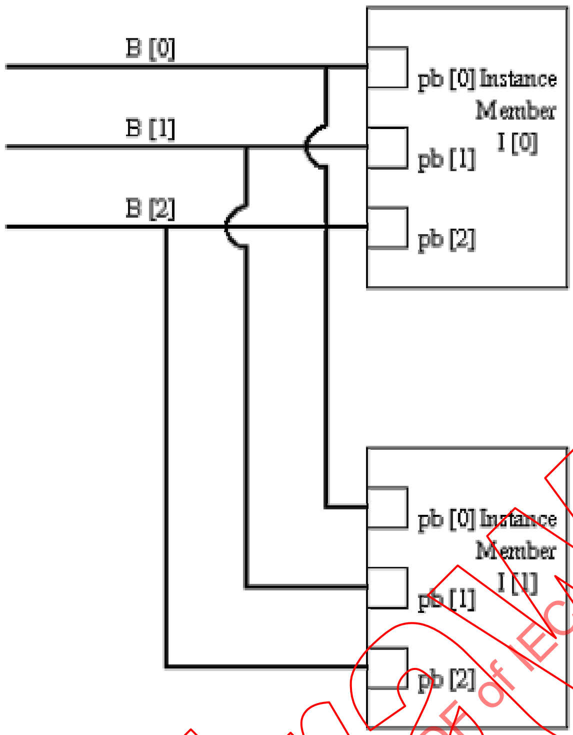


Figure 16 – Connectivity bus – Logical equivalent of commoning

Alternatively, the port bundle may be smaller than the bus and all of the members of the port bundles of all of the members of the instance may be needed to connect up to the bus as shown below. This type of connection is termed fanning out as the bus may be considered to fan out to the members of the instances.

A special case of fanning-out may occur where the bus is connected up to a single port of a wide instance. Each port of the members of the instance is then connected up to a different signal of the bus as shown below.

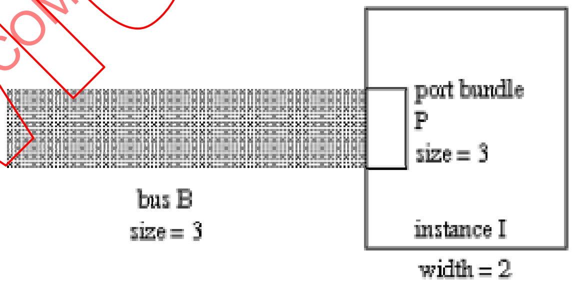


Figure 17 – Connectivity bus – Fanning-out

In the above example, the first signal associated with bus B is connected to an instance member logical port corresponding to port P of the first member of instance I, i.e. since the width of the instance is 3, there are three signals associated with bus B as shown below

IECNORM.COM : Click to view the full PDF of IEC 62016:2003
Withdrawn

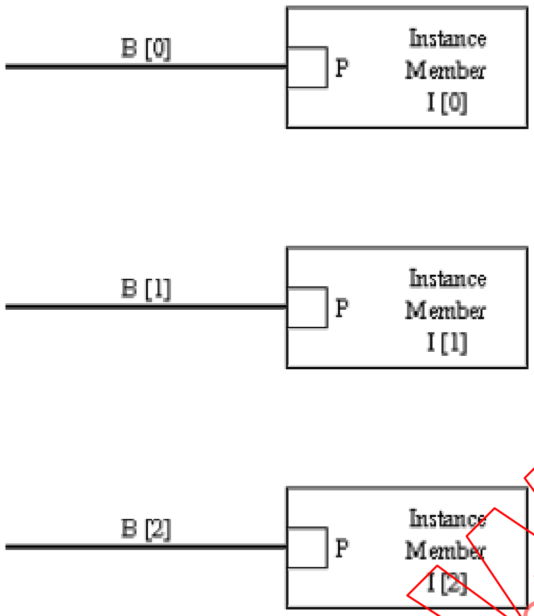


Figure18 – Connectivity bus – Logical equivalent of fanning-out

IECNORM.COM : Click to view the full PDF of IEC 62016:2003
Withdrawn

There is a degenerate case of a connectivity bus which structures a signal bundle which contains only one signal i.e. the connectivity bus is one-bit wide. In this case, it can join instance member structure ports and master structure ports.

Since a connectivity bus exists to specify a structured realization of a signal bundle, its sub-structure can be specified by connectivity sub-buses and/or connectivity bus-slices.

A connectivity sub-bus is similar in concept to a connectivity sub-net and may be used to specify details such as criticality, or merely to indicate that the originating system considers the connectivity bus to be broken down in this manner.

In the case of a connectivity sub-bus, the connectivity which is realized by the connectivity sub-bus does not state all of the structure ports joined. Conversely a connectivity bus-slice considers a sub-set of the sets of structure ports joined by the overall connectivity bus. An example of this is shown below.

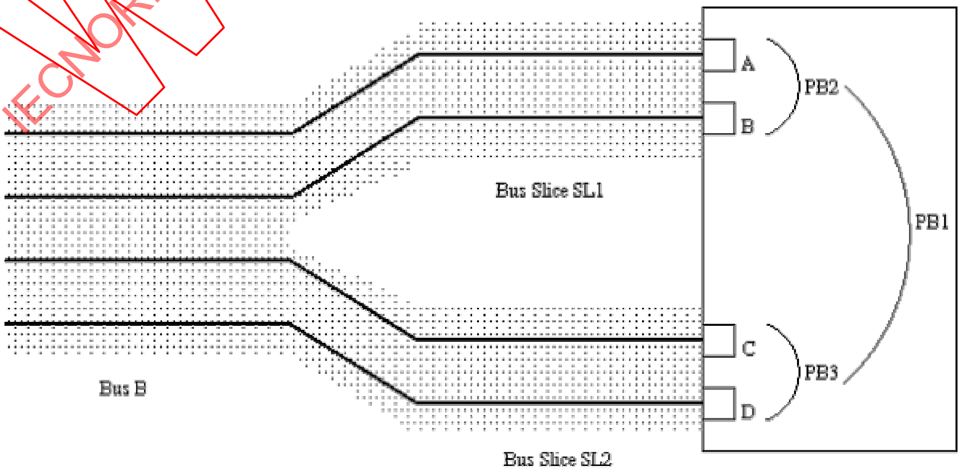


Figure 19 – Connectivity bus-slice

Bus B connects to port bundle PB1. It is sliced into two separate bus slices, SL1 and SL2, which connect to subsets of PB1, PB2 and PB3 respectively. It should be noted that it is not necessary for bus slices to connect to all the parts of PB1; the removal of SL2 would still result in a valid description.

A sub-set of a signal's logical connectivity may be realized by one connectivity net or bus and another sub-set of its connectivity by part of a different connectivity bus. The Core Model allows the description of the relationship between two discrete structured realizations of a signal. This relationship is specified by a connectivity ripper which is referenced by the connectivity nets and buses, as shown below.

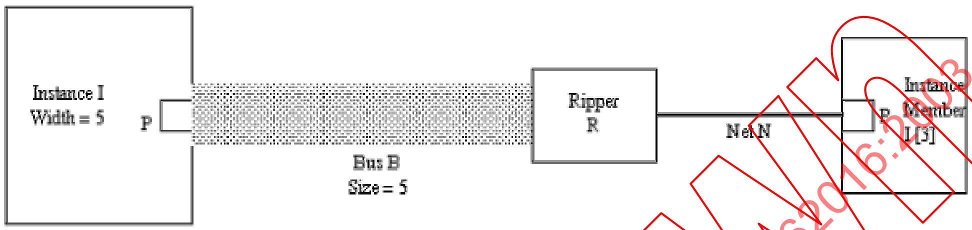


Figure 20 – Connectivity ripper

In the above example, bus B connects to all ports P of the members instance I. However, net N connects up to port P of I[3]. In order to relate the two structurings of the connection to P of I[3], the ripper is referenced by both bus B and net N.

7.7 Global connectivity

The descriptions of logical and structural connectivity above, considered the explicit definition of connectedness. Within the Core Model, it is possible to implicitly specify connectivity throughout sub-sets of a design. This is useful for cases such as gates which connect up to power and ground but where the system does not specify such connections throughout the design hierarchy. There is, thus, a need to state that individual logical ports of the gates should eventually be connected up to power (or ground). This is achieved within the Core Model by the specification of global ports which may be considered to be place-holders for actual connections within a design hierarchy.

A signal may connect explicitly to a global port, thus indicating that the master logical ports and instance member logical ports also joined by the signal will eventually be connected up to something which implements the global port requirement.

Alternatively, a master logical port may specify that it is by default connected to a given global port. In this case, unless the master logical port is connected up, it is implicitly connected to the global port.

The global port connections are resolved during the design hierarchy specification (see below). Such resolutions are termed global port scopes. The specification of a global port scope for a given global port indicates that all ports, within that subset of the design hierarchy, which are connected to the global port should be assumed to be connected together; moreover, those ports should not be considered to be connected to ports in other parts of the design hierarchy unless explicitly connected by signals. Such a mechanism may be used to allow the specification of re-usable modules which require clock signals throughout their hierarchy as shown below.

IECNORM.COM : Click to view the full PDF of IEC 62016:2003
Withdrawn

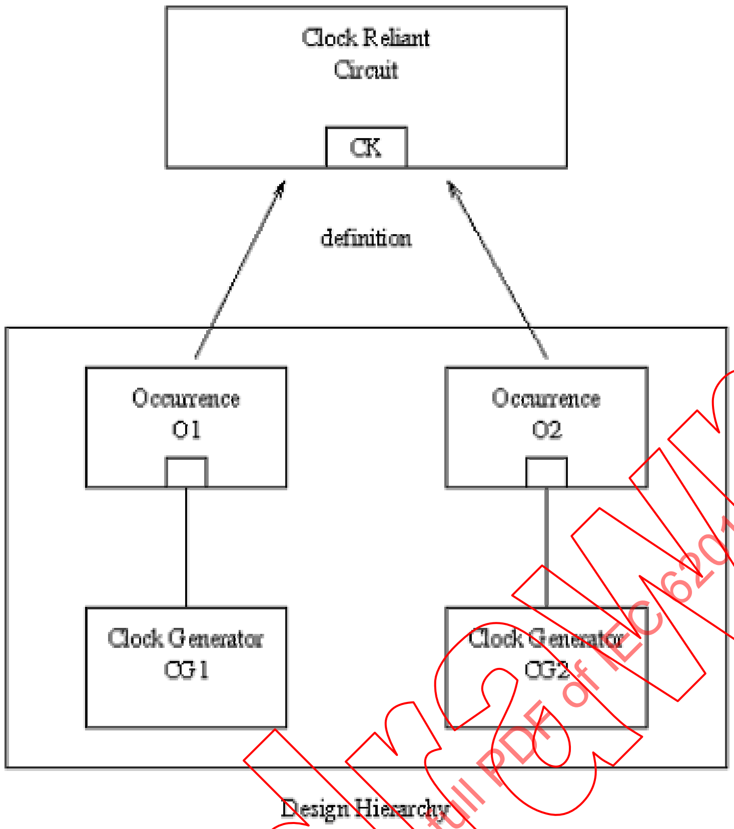


Figure 21 – Global port scoping

IECNORM.COM : Click to view the full PDF of IEC 62016:2003

O1 and O2 are occurrences of the same cluster which specifies circuitry which relies upon the presence of a clock signal. For example, the port CK may be explicitly connected to the global port clock. Global port scoping allows O1 and O2 to have their global port clock scoped given that the requirement for CK to be connected to a clock is satisfied by the connections to the two separate clock generators.

A global port will commonly be scoped because a master logical port of the current level of the design hierarchy will realize the necessary connection, for example, by feeding in a clock signal as shown above. This is not necessarily the case, however; for example, a set of modules in the subset of the hierarchy may read and write data onto a common bus without that data being required outside the subset of the hierarchy.

7.8 Design and configuration

The libraries specify sets of cells. In many cases, however, it is necessary to identify the top level cell of a design. Since a cell may interact with the outside world in different ways, as specified by the available interfaces, there may be several different manners in which the design may be considered; these are termed design hierarchies.

Different design hierarchies may also be specified according to the way in which the instances within the cell are configured.

A cluster may have several cell representations which interact in the same manner (as specified by the cluster interface). As a result, the instantiation of a cluster does not determine the exact cell representation which will occur within the design hierarchy. As a result, an instance may be specified as being

- Unconfigured

  The system has not determined whether or not the instance will be associated with a particular cell representation.
- A leaf

  No corresponding cell representation should be used to expand the hierarchy.
- Expandable

  A particular cell representation is to further expand the hierarchy.

There are two different ways in which instance configuration could have been specified.

1. By defining the configuration for each occurrence of an instance within the design hierarchy.
2. By the definition within a cluster of the configurations which the system considers to be reasonable. A design hierarchy then identifies the chosen configuration of a cluster of the top-level design cell.

Within the Core Model, the second mechanism has been chosen.

7.9 Annotation

In the context of a design hierarchy, new information or overriding values for previously defined values may be specified for parts of the design hierarchy.

Within any occurrence, properties and attributes of the cell, cluster and its ports may be added or overridden.

Within an occurrence which has been configured to a specific cell representation properties of the cell representation may be overridden.

Within an occurrence which has been configured to an internal connectivity view, properties and attributes of the signals, nets and buses, and sub-occurrences, i.e. instance members, may be overridden.

8 Core Model EXPRESS-G

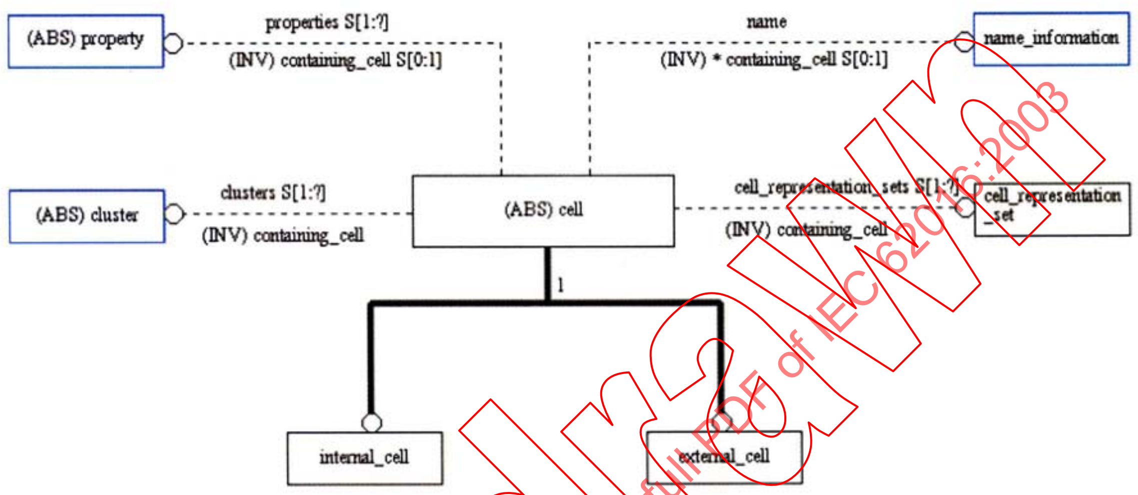
This section contains partial EXPRESS-G of some of the key objects of this Core Model of the electronics domain.

Withdrawn
IECNORM.COM : Click to view the full PDF of IEC 62016:2003

8.1 Partial EXPRESS-G of cell

The following EXPRESS-G diagram is incomplete and does not necessarily include all the relationships of cell. It is included for informative purposes only. In case of any discrepancy with the EXPRESS description, the EXPRESS shall take precedence.

The blue boxes are linked to further EXPRESS-G diagrams. The black boxes are, apart from simple EXPRESS types, linked to the EXPRESS description of the object.

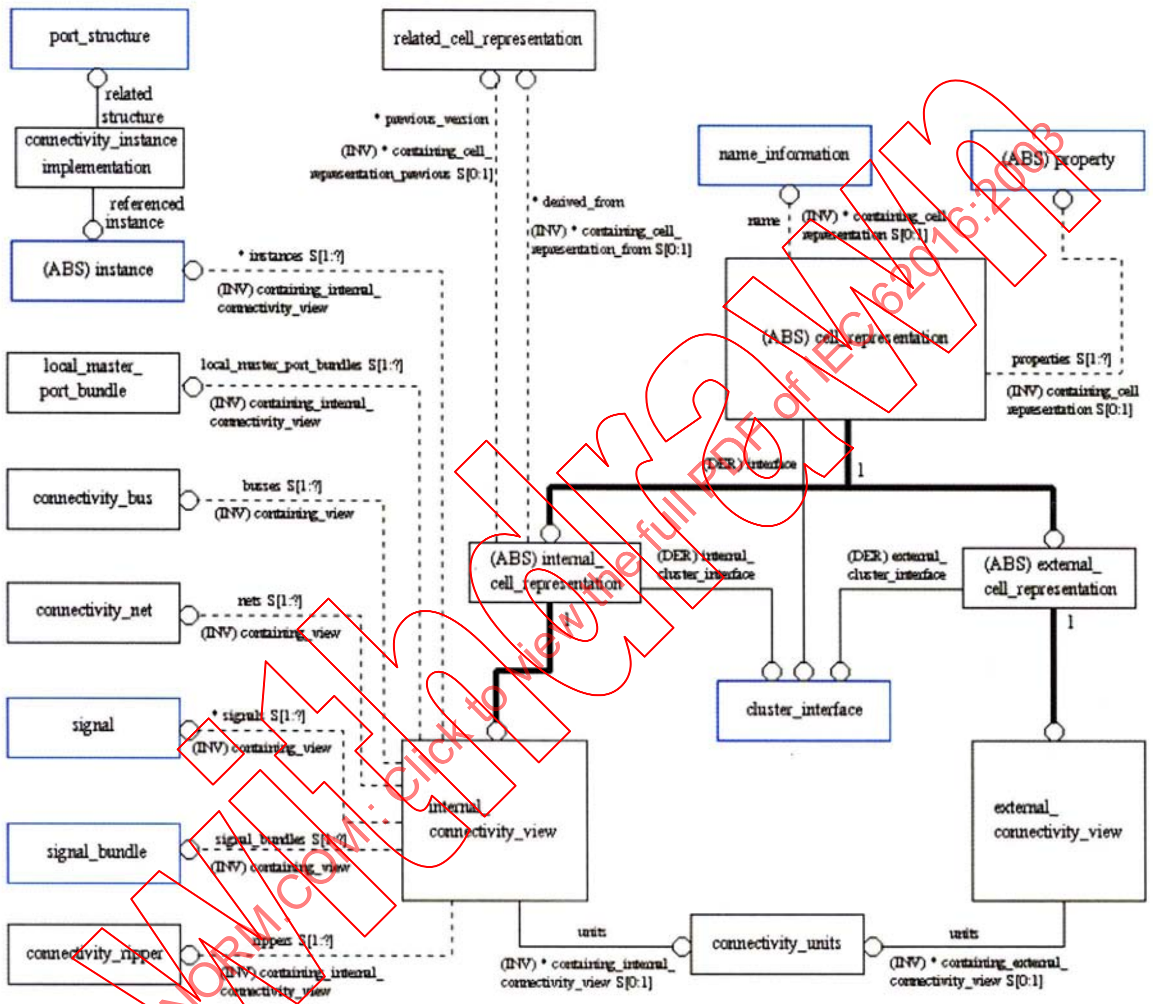


Withdrawn
IECNORM.COM : Click to view the full PDF of IEC 62016:2003

8.2 Partial EXPRESS-G of cell_representation

The following EXPRESS-G diagram is incomplete and does not necessarily include all the relationships of cell_representation. It is included for informative purposes only. In case of any discrepancy with the EXPRESS description, the EXPRESS shall take precedence.

The blue boxes are linked to further EXPRESS-G diagrams. The black boxes are, apart from simple EXPRESS types, linked to the EXPRESS description of the object.

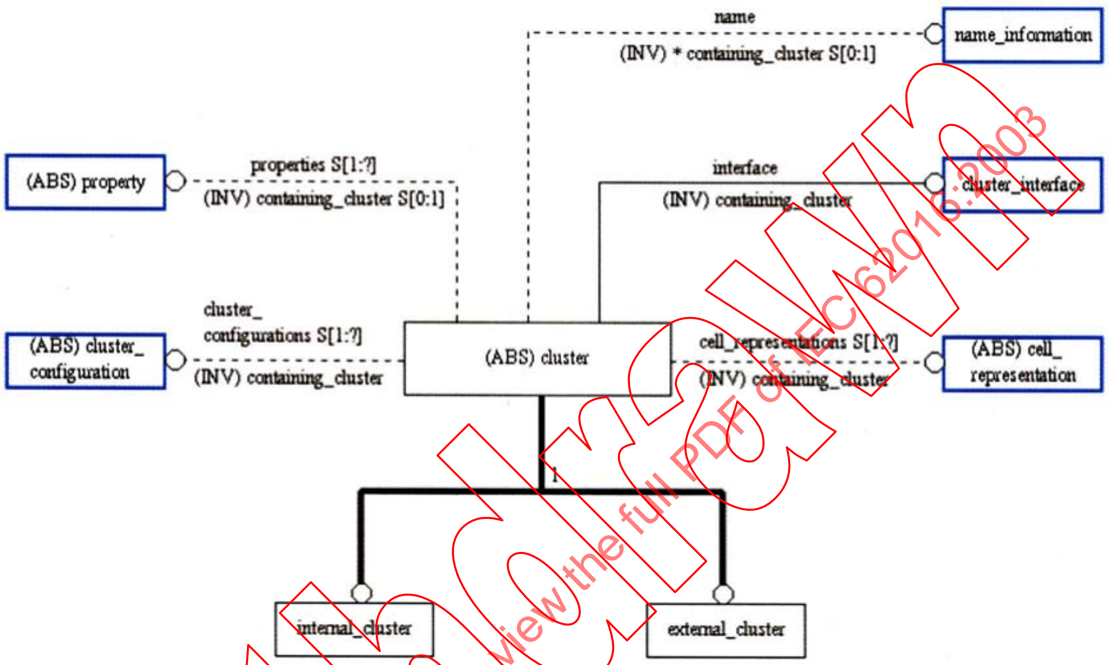

port_structure
related structure
connectivity_instance implementation
referenced instance
(ABS) instance
* instances S[1:?]
(INV) containing_internal_connectivity_view
local_master_port_bundle
local_master_port_bundles S[1:?]
(INV) containing_internal_connectivity_view
connectivity_bus
buses S[1:?]
(INV) containing_view
connectivity_net
nets S[1:?]
(INV) containing_view
signal
* signals S[1:?]
(INV) containing_view
signal_bundle
signal_bundles S[1:?]
(INV) containing_view
connectivity_ripper
rippers S[1:?]
(INV) containing_internal_connectivity_view
related_cell_representation
* previous_version
(INV) * containing_cell_representation_previous S[0:1]
* derived_from
(INV) * containing_cell_representation_from S[0:1]
name_information
name
(INV) * containing_cell_representation S[0:1]
(ABS) property
(ABS) cell_representation
properties S[1:?]
(INV) containing_cell_representation S[0:1]
(DER) interface
1
(ABS) internal_cell_representation
(DER) internal_cluster_interface
(DER) external_cluster_interface
(ABS) external_cell_representation
1
cluster_interface
internal_connectivity_view
external_connectivity_view
units
(INV) * containing_internal_connectivity_view S[0:1]
connectivity_units
units
(INV) * containing_external_connectivity_view S[0:1]

IECNORM.COM : Click to view the full PDF of IEC 62016:2003

8.3 Partial EXPRESS-G of cluster

The following EXPRESS-G diagram is incomplete and does not, necessarily, include all the relationships of cluster. It is included for informative purposes only. In case of any discrepancy with the EXPRESS description, the EXPRESS shall take precedence.

The blue boxes are linked to further EXPRESS-G diagrams. The black boxes are, apart from simple EXPRESS types, linked to the EXPRESS description of the object.

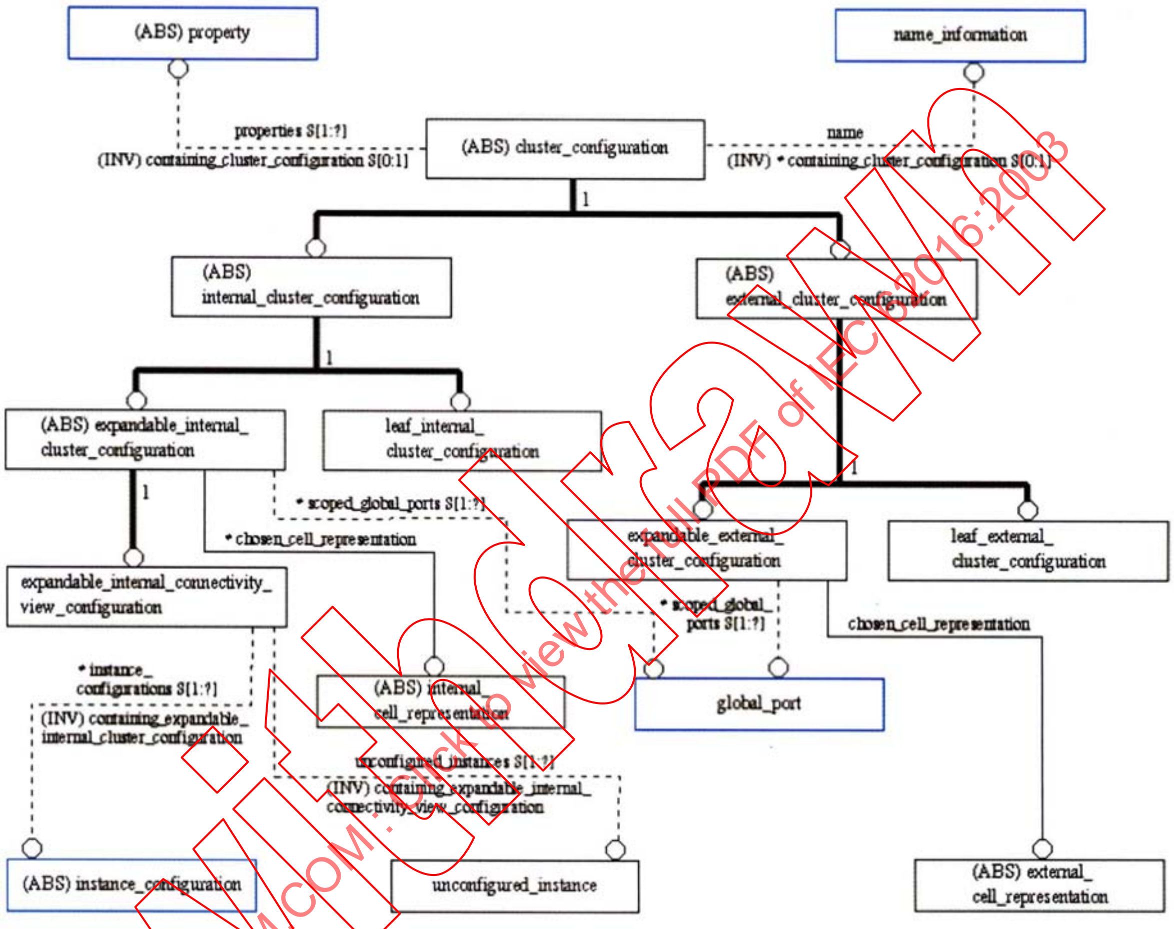


Withdrawn
IECNORM.COM : Click to view the full PDF of IEC 62016:2003

8.4 Partial EXPRESS-G of cluster_configuration

The following EXPRESS-G diagram is incomplete and does not necessarily include all the relationships of cluster_configuration. It is included for informative purposes only. In case of any discrepancy with the EXPRESS description, the EXPRESS shall take precedence.

The blue boxes are linked to further EXPRESS-G diagrams. The black boxes are, apart from simple EXPRESS types, linked to the EXPRESS description of the object.

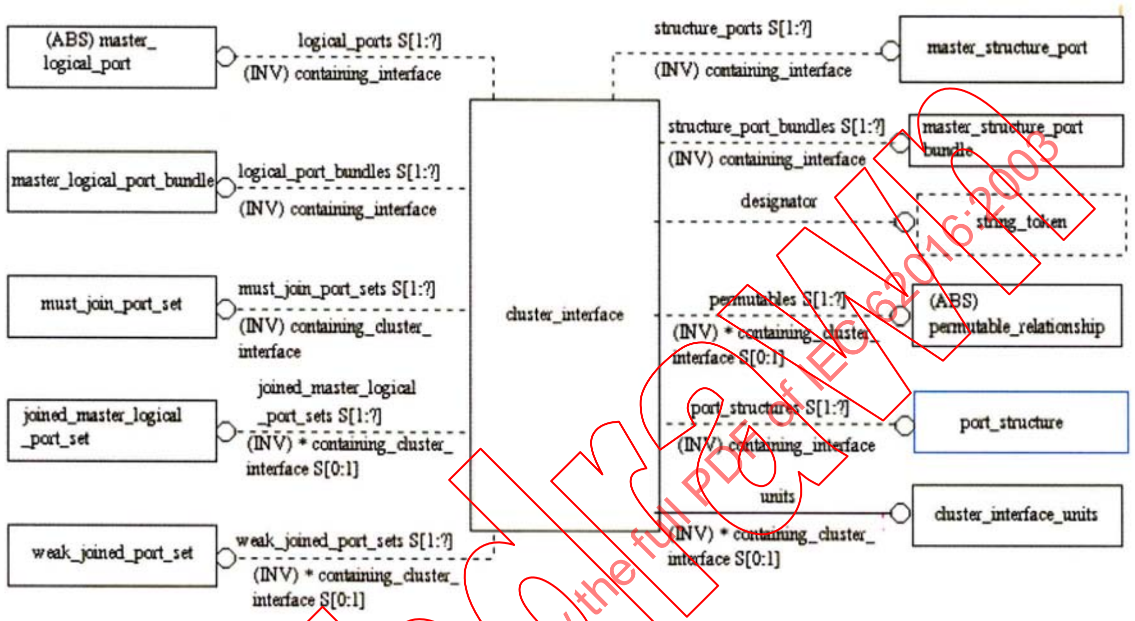

(ABS) property
name_information
properties S[1:?]
(INV) containing_cluster_configuration S[0:1]
(ABS) cluster_configuration
name
(INV) *containing_cluster_configuration S[0:1]
1
(ABS) internal_cluster_configuration
(ABS) external_cluster_configuration
1
(ABS) expandable_internal_cluster_configuration
leaf_internal_cluster_configuration
1
*scoped_global_ports S[1:?]
*chosen_cell_representation
1
expandable_external_cluster_configuration
leaf_external_cluster_configuration
expandable_internal_connectivity_view_configuration
*scoped_global_ports S[1:?]
chosen_cell_representation
*instance_configurations S[1:?]
(INV) containing_expandable_internal_cluster_configuration
(ABS) internal_cell_representation
global_port
unconfigured_instances S[1:?]
(INV) containing_expandable_internal_connectivity_view_configuration
(ABS) instance_configuration
unconfigured_instance
(ABS) external_cell_representation

IECNORM.COM : Click to view the full PDF of IEC 62016:2003
Withdrawn

8.5 Partial EXPRESS-G of cluster_interface

The following EXPRESS-G diagram is incomplete and does not necessarily include all the relationships of cluster_interface. It is included for informative purposes only. In case of any discrepancy with the EXPRESS description, the EXPRESS shall take precedence.

The blue boxes are linked to further EXPRESS-G diagrams. The black boxes are, apart from simple EXPRESS types, linked to the EXPRESS description of the object.

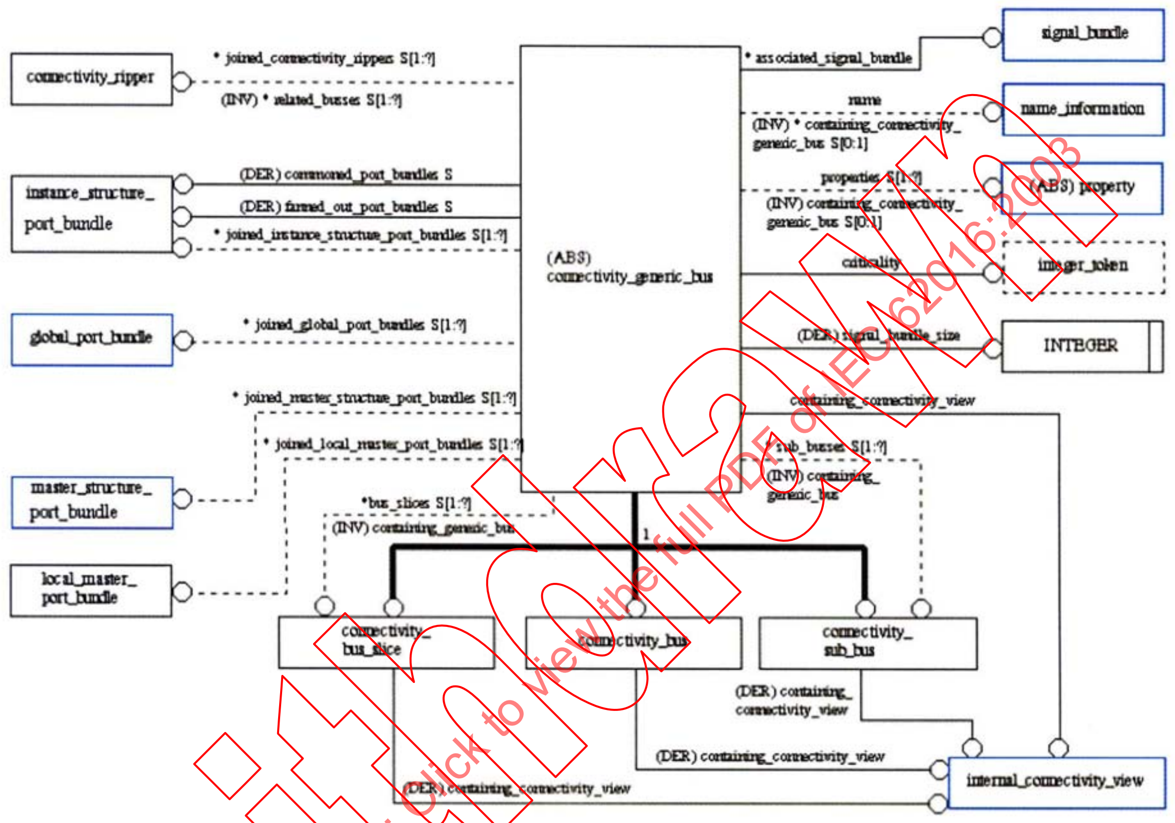

(ABS) master_logical_port
logical_ports S[1:?]
(INV) containing_interface
master_logical_port_bundle
logical_port_bundles S[1:?]
(INV) containing_interface
must_join_port_set
must_join_port_sets S[1:?]
(INV) containing_cluster_interface
joined_master_logical_port_set
joined_master_logical_port_sets S[1:?]
(INV) * containing_cluster_interface S[0:1]
weak_joined_port_set
weak_joined_port_sets S[1:?]
(INV) * containing_cluster_interface S[0:1]
cluster_interface
structure_ports S[1:?]
(INV) containing_interface
master_structure_port
structure_port_bundles S[1:?]
(INV) containing_interface
master_structure_port_bundle
designator
string_token
permutables S[1:?]
(INV) * containing_cluster_interface S[0:1]
(ABS) permutable_relationship
port_structures S[1:?]
(INV) containing_interface
port_structure
units
(INV) * containing_cluster_interface S[0:1]
cluster_interface_units

Withdrawn
IECNORM.COM : Click to view the full PDF of IEC 62016:2003

8.6 Partial EXPRESS-G of connectivity_generic_bus

The following EXPRESS-G diagram is incomplete and does not necessarily include all the relationships of connectivity_generic_bus. It is included for informative purposes only. In case of any discrepancy with the EXPRESS description, the EXPRESS shall take precedence.

The blue boxes are linked to further EXPRESS-G diagrams. The black boxes are, apart from simple EXPRESS types, linked to the EXPRESS description of the object.



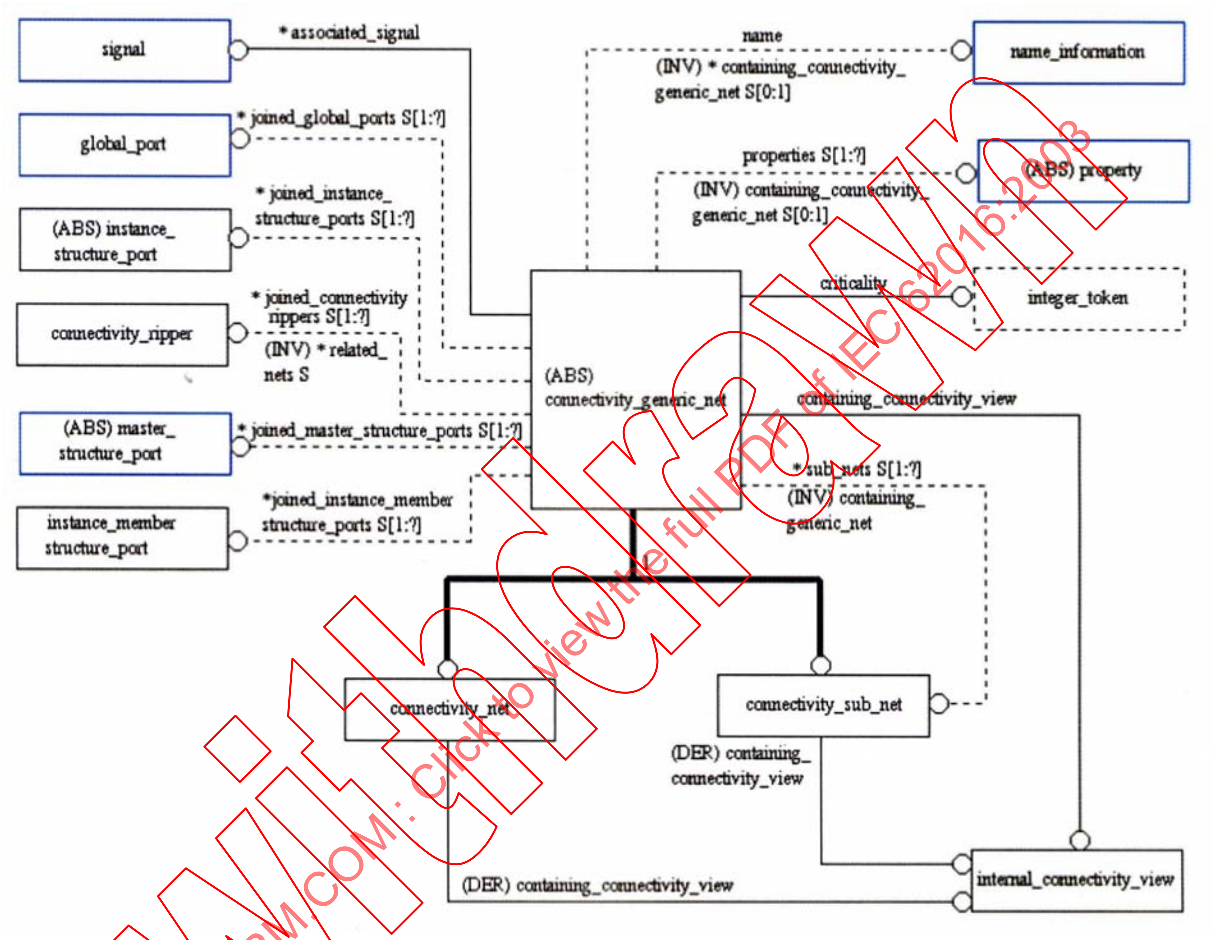
connectivity_ripper
* joined_connectivity_rippers S[1:?]
(INV) * related_busses S[1:?]
instance_structure_port_bundle
(DER) commoned_port_bundles S
(DER) fanned_out_port_bundles S
* joined_instance_structure_port_bundles S[1:?]
global_port_bundle
* joined_global_port_bundles S[1:?]
* joined_master_structure_port_bundles S[1:?]
* joined_local_master_port_bundles S[1:?]
master_structure_port_bundle
local_master_port_bundle
*bus_slices S[1:?]
(INV) containing_generic_bus
(ABS) connectivity_generic_bus
* associated_signal_bundle
signal_bundle
name
name_information
(INV) * containing_connectivity_generic_bus S[0:1]
properties S[1:?]
(ABS) property
(INV) containing_connectivity_generic_bus S[0:1]
criticality
integer_token
(DER) signal_bundle_size
INTEGER
containing_connectivity_view
* sub_busses S[1:?]
(INV) containing_generic_bus
1
connectivity_bus_slice
connectivity_bus
connectivity_sub_bus
(DER) containing_connectivity_view
(DER) containing_connectivity_view
(DER) containing_connectivity_view
internal_connectivity_view

IECNORM.COM : Click to view the full PDF of IEC 62016:2003
Withdrawn

8.7 Partial EXPRESS-G of connectivity_generic_net

The following EXPRESS-G diagram is incomplete and does not necessarily include all the relationships of connectivity_generic_net. It is included for informative purposes only. In case of any discrepancy with the EXPRESS description, the EXPRESS shall take precedence.

The blue boxes are linked to further EXPRESS-G diagrams. The black boxes are, apart from simple EXPRESS types, linked to the EXPRESS description of the object.

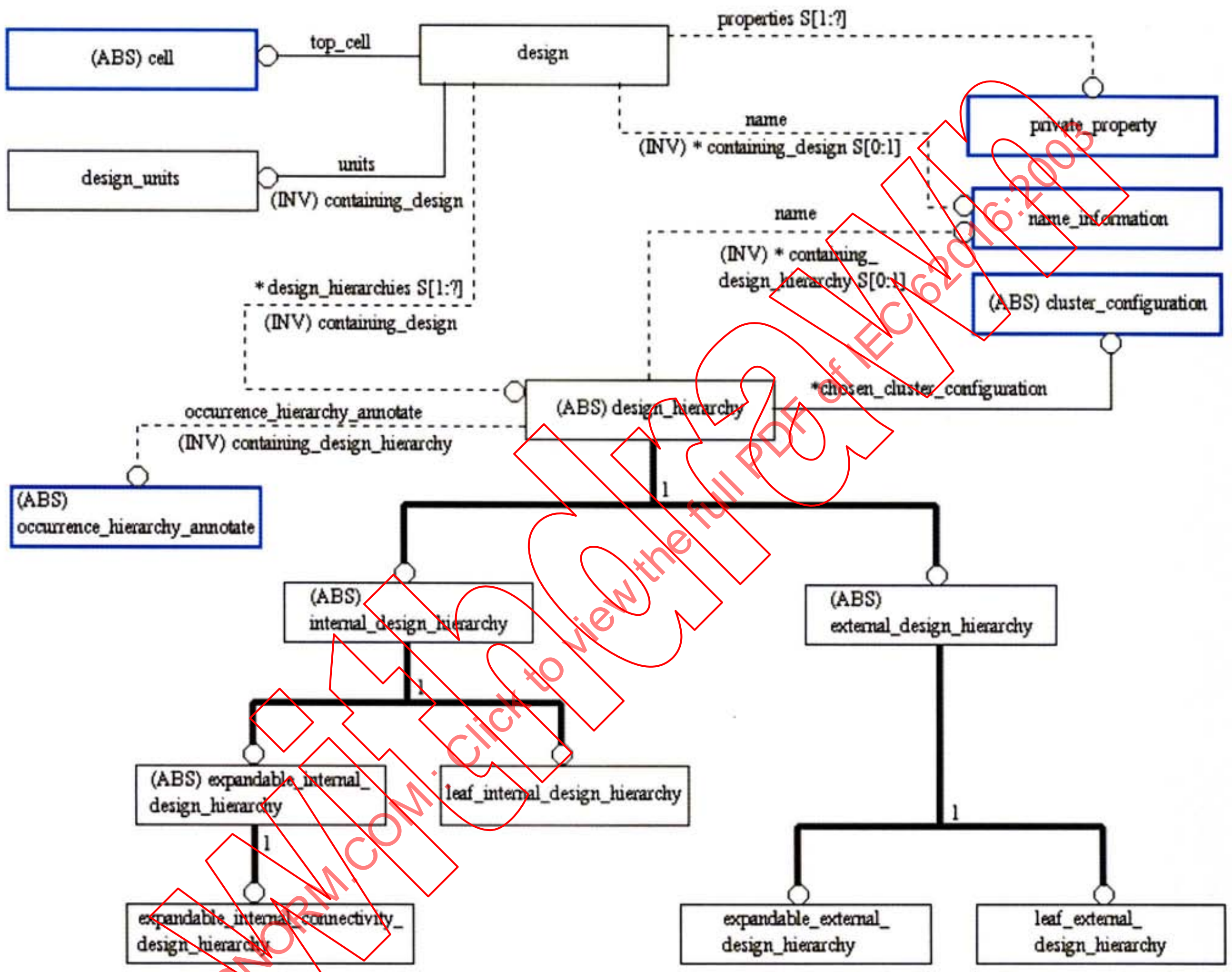

signal
* associated_signal
global_port
* joined_global_ports S[1:?]
(ABS) instance_ structure_port
* joined_instance_ structure_ports S[1:?]
connectivity_ripper
* joined_connectivity rippers S[1:?]
(INV) * related_ nets S
(ABS) master_ structure_port
* joined_master_structure_ports S[1:?]
instance_member structure_port
*joined_instance_member structure_ports S[1:?]
name
(INV) * containing_connectivity_ generic_net S[0:1]
name_information
properties S[1:?]
(INV) containing_connectivity_ generic_net S[0:1]
(ABS) property
criticality
integer_token
(ABS) connectivity_generic_net
containing_connectivity_view
* sub_nets S[1:?]
(INV) containing_ generic_net
connectivity_net
connectivity_sub_net
(DER) containing_ connectivity_view
(DER) containing_connectivity_view
internal_connectivity_view

IECNORM.COM : Click to view the full PDF of IEC 62016:2003
Withdrawn

8.8 Partial EXPRESS-G of design

The following EXPRESS-G diagram is incomplete and does not necessarily include all the relationships of design. It is included for informative purposes only. In case of any discrepancy with the EXPRESS description, the EXPRESS shall take precedence.

The blue boxes are linked to further EXPRESS-G diagrams. The black boxes are, apart from simple EXPRESS types, linked to the EXPRESS description of the object.

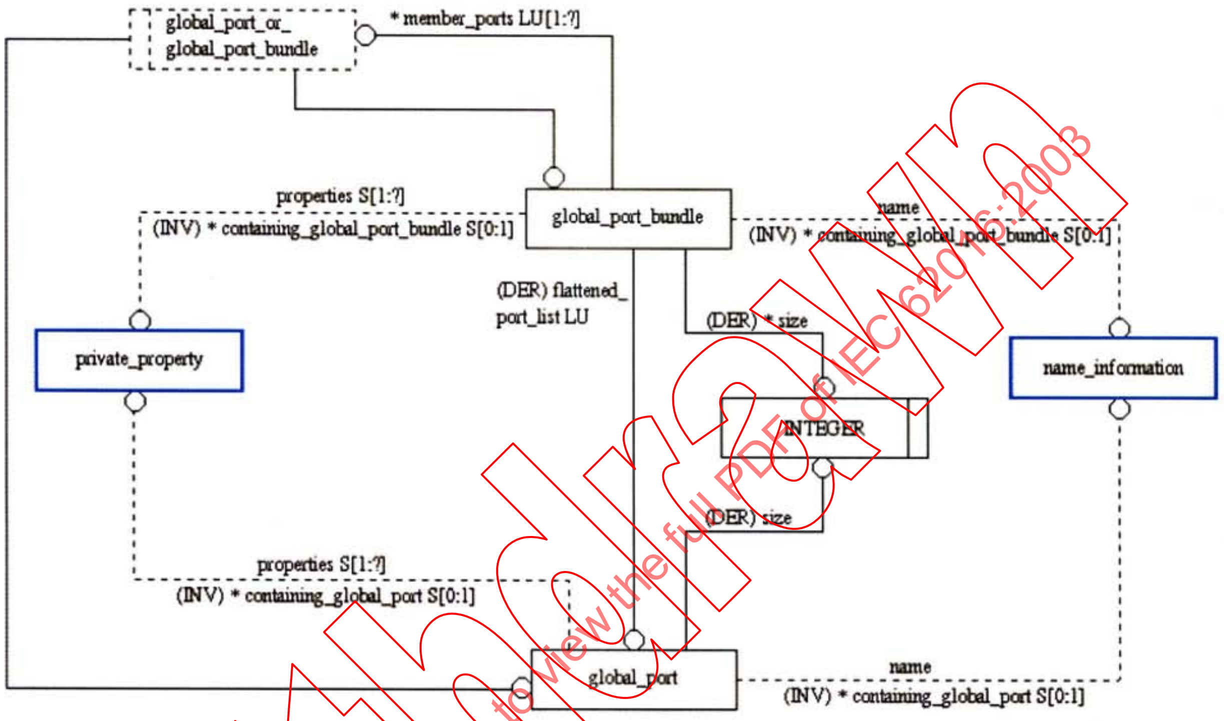

(ABS) cell
top_cell
design
properties S[1:?]
private_property
design_units
units
(INV) containing_design
name
(INV) * containing_design S[0:1]
name_information
name
(INV) * containing_
design_hierarchy S[0:1]
(ABS) cluster_configuration
* design_hierarchies S[1:?]
(INV) containing_design
*chosen_cluster_configuration
occurrence_hierarchy_annotate
(INV) containing_design_hierarchy
(ABS) design_hierarchy
(ABS)
occurrence_hierarchy_annotate
1
(ABS)
internal_design_hierarchy
(ABS)
external_design_hierarchy
1
(ABS) expandable_internal_
design_hierarchy
leaf_internal_design_hierarchy
1
1
expandable_internal_connectivity_
design_hierarchy
expandable_external_
design_hierarchy
leaf_external_
design_hierarchy

IECNORM.COM : Click to view the full PDF of IEC 62016:2003
Withdrawn

8.9 Partial EXPRESS-G of global_port

The following EXPRESS-G diagram is incomplete and does not necessarily include all the relationships of global_port. It is included for informative purposes only. In case of any discrepancy with the EXPRESS description, the EXPRESS shall take precedence.

The blue boxes are linked to further EXPRESS-G diagrams. The black boxes are, apart from simple EXPRESS types, linked to the EXPRESS description of the object.

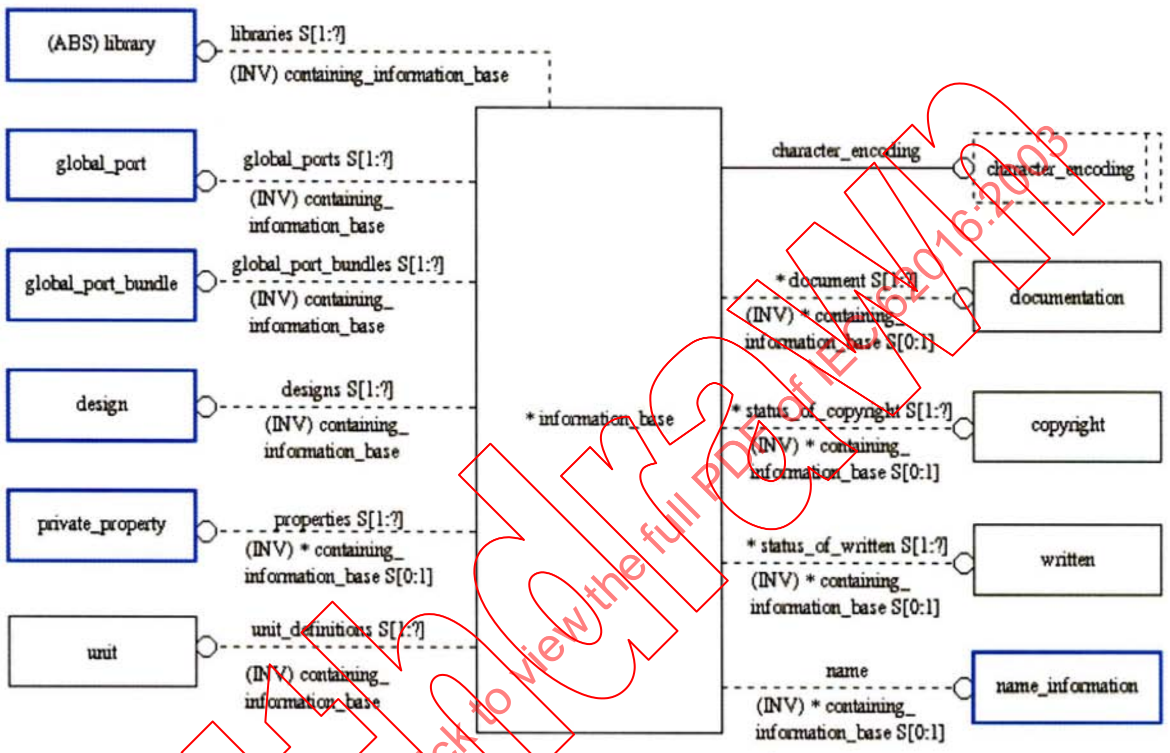


Withdrawn
IECNORM.COM : Click to view the full PDF of IEC 62016:2003

8.10 Partial EXPRESS-G of information_base

The following EXPRESS-G diagram is incomplete and does not necessarily include all the relationships of information_base. It is included for informative purposes only. In case of any discrepancy with the EXPRESS description, the EXPRESS shall take precedence.

The blue boxes are linked to further EXPRESS-G diagrams. The black boxes are, apart from simple EXPRESS types, linked to the EXPRESS description of the object.

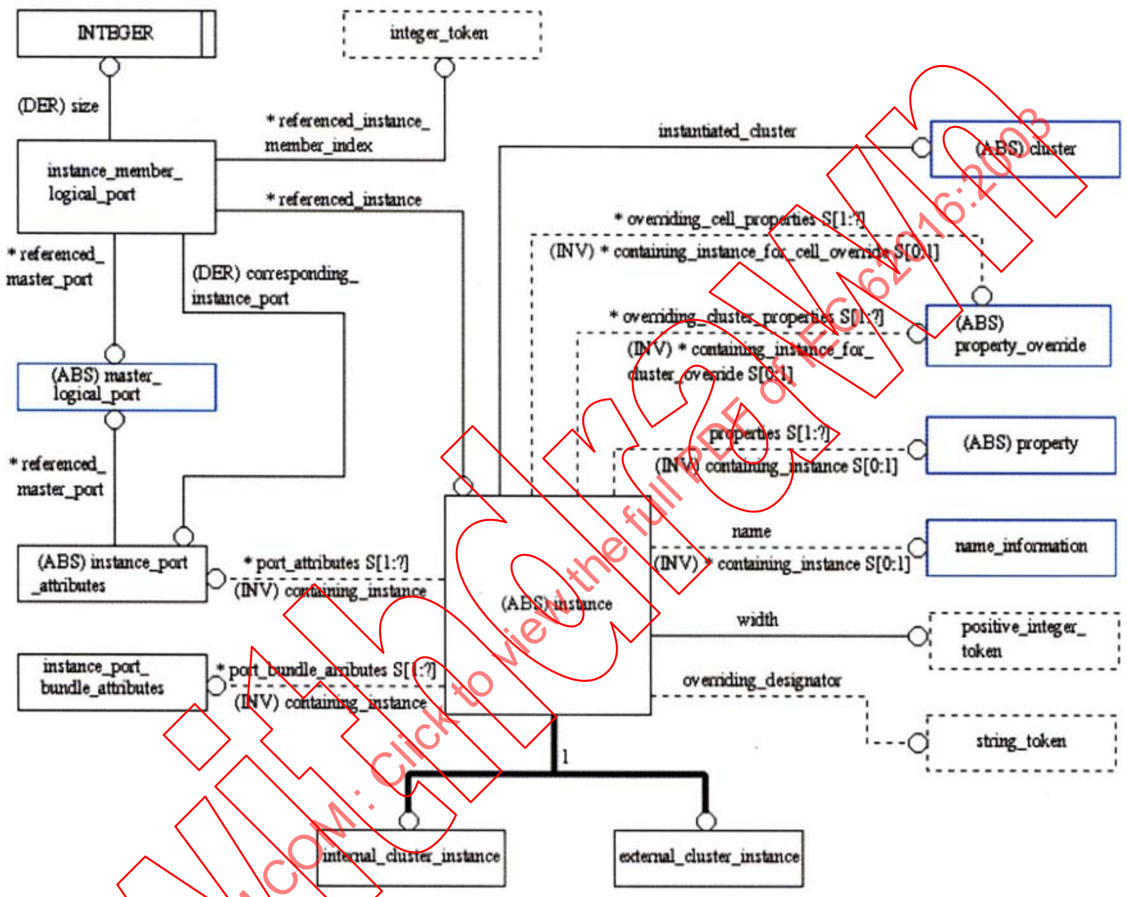

(ABS) library
libraries S[1:?]
(INV) containing_information_base
global_port
global_ports S[1:?]
(INV) containing_ information_base
global_port_bundle
global_port_bundles S[1:?]
(INV) containing_ information_base
design
designs S[1:?]
(INV) containing_ information_base
private_property
properties S[1:?]
(INV) * containing_ information_base S[0:1]
unit
unit_definitions S[1:?]
(INV) containing_ information_base
* information_base
character_encoding
character_encoding
* document S[1:?]
(INV) * containing_ information_base S[0:1]
documentation
* status_of_copyright S[1:?]
(INV) * containing_ information_base S[0:1]
copyright
* status_of_written S[1:?]
(INV) * containing_ information_base S[0:1]
written
name
(INV) * containing_ information_base S[0:1]
name_information

IECNORM.COM : Click to view the full PDF of IEC 62016:2003
Withdrawn

8.11 Partial EXPRESS-G of instance

The following EXPRESS-G diagram is incomplete and does not necessarily include all the relationships of instance. It is included for informative purposes only. In case of any discrepancy with the EXPRESS description, the EXPRESS shall take precedence.

The blue boxes are linked to further EXPRESS-G diagrams. The black boxes are, apart from simple EXPRESS types, linked to the EXPRESS description of the object.

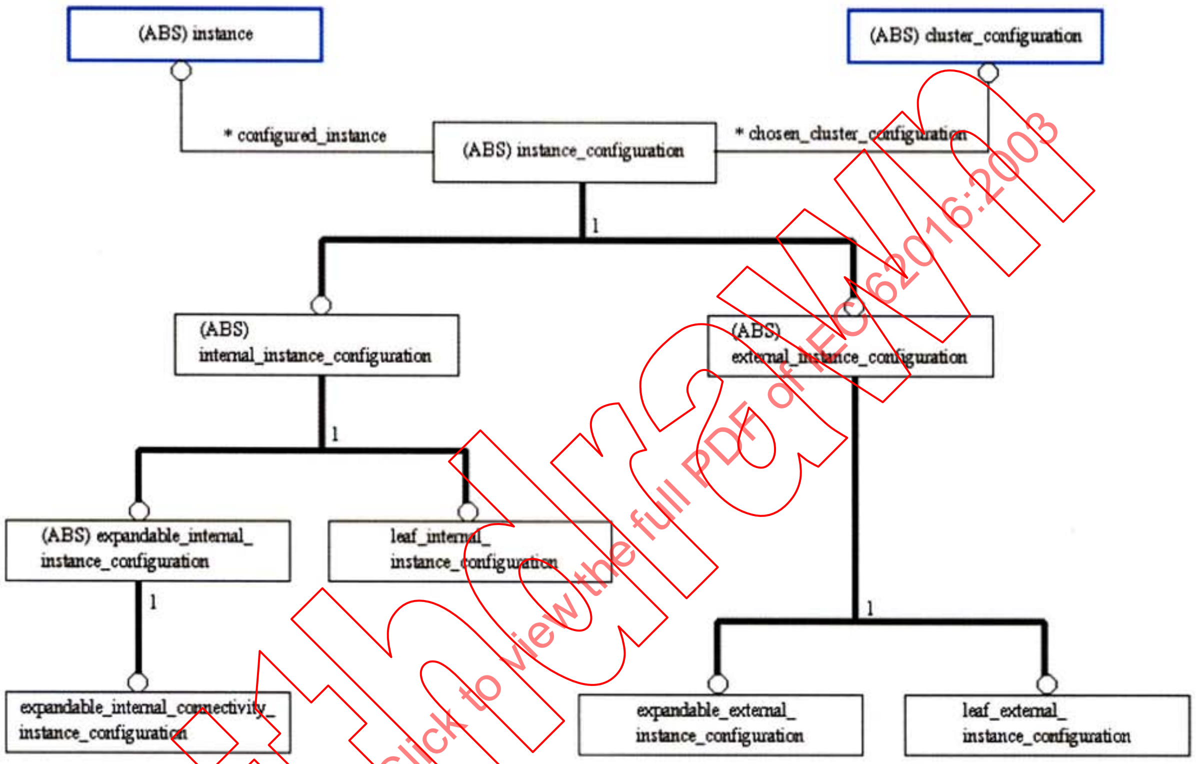

INTEGER
integer_token
(DER) size
* referenced_instance_member_index
instantiated_cluster
(ABS) cluster
instance_member_logical_port
* referenced_instance
* overriding_cell_properties S[1:?]
(INV) * containing_instance_for_cell_override S[0:1]
* referenced_master_port
(DER) corresponding_instance_port
* overriding_cluster_properties S[1:?]
(INV) * containing_instance_for_cluster_override S[0:1]
(ABS) property_override
(ABS) master_logical_port
properties S[1:?]
(INV) containing_instance S[0:1]
(ABS) property
* referenced_master_port
name
(INV) * containing_instance S[0:1]
name_information
(ABS) instance_port_attributes
* port_attributes S[1:?]
(INV) containing_instance
(ABS) instance
width
positive_integer_token
instance_port_bundle_attributes
* port_bundle_attributes S[1:?]
(INV) containing_instance
overriding_designator
string_token
1
internal_cluster_instance
external_cluster_instance

Withdrawn
IECNORM.COM : Click to view the full PDF of IEC 62016:2003

8.12 Partial EXPRESS-G of instance_configuration

The following EXPRESS-G diagram is incomplete and does not necessarily include all the relationships of instance_configuration. It is included for informative purposes only. In case of any discrepancy with the EXPRESS description, the EXPRESS shall take precedence.

The blue boxes are linked to further EXPRESS-G diagrams. The black boxes are, apart from simple EXPRESS types, linked to the EXPRESS description of the object.

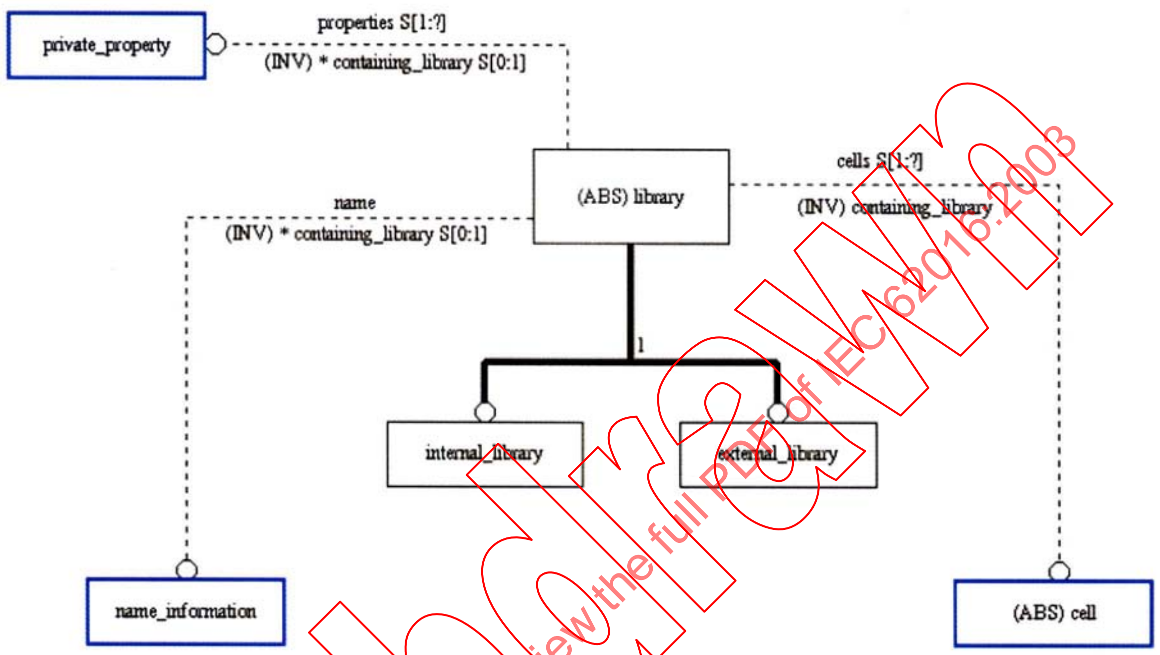

(ABS) instance
(ABS) cluster_configuration
* configured_instance
(ABS) instance_configuration
* chosen_cluster_configuration
1
(ABS) internal_instance_configuration
(ABS) external_instance_configuration
1
(ABS) expandable_internal_instance_configuration
leaf_internal_instance_configuration
1
1
expandable_internal_connectivity_instance_configuration
expandable_external_instance_configuration
leaf_external_instance_configuration

Withdrawn
IECNORM.COM : Click to view the full PDF of IEC 62016:2003

8.13 Partial EXPRESS-G of library

The following EXPRESS-G diagram is incomplete and does not necessarily include all the relationships of library. It is included for informative purposes only. In case of any discrepancy with the EXPRESS description, the EXPRESS shall take precedence.

The blue boxes are linked to further EXPRESS-G diagrams. The black boxes are, apart from simple EXPRESS types, linked to the EXPRESS description of the object.

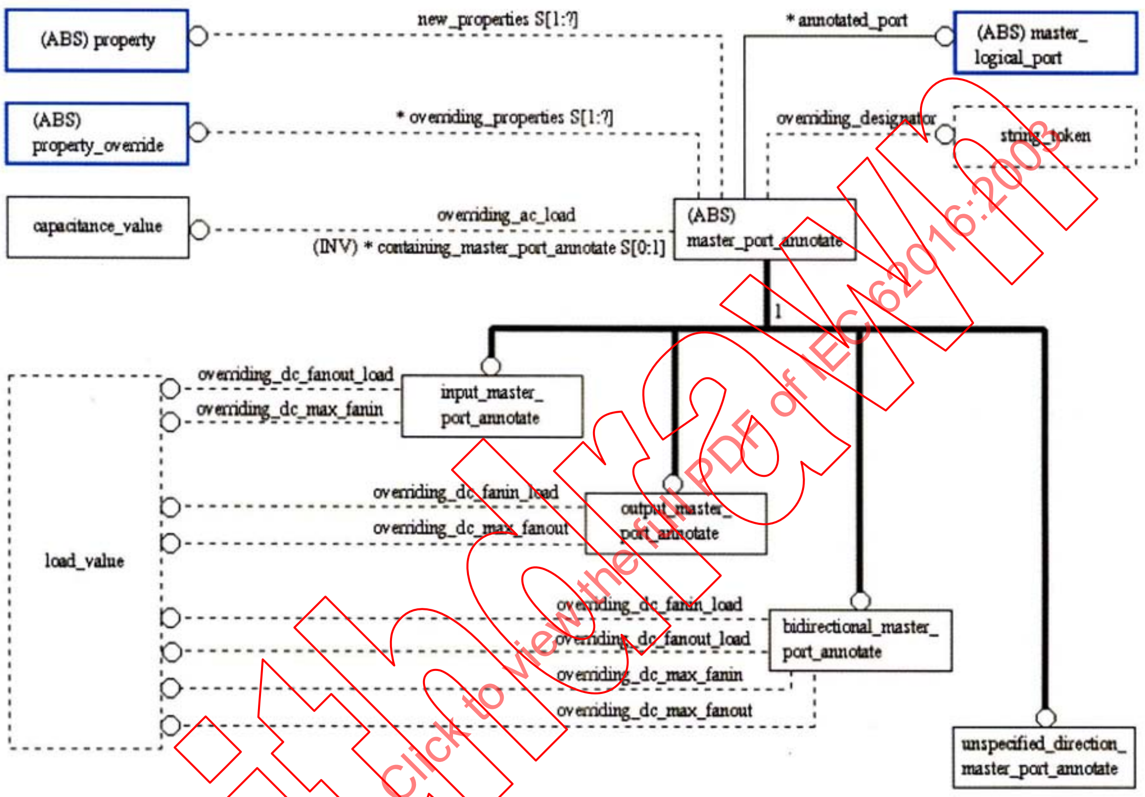


Withdrawn
IECNORM.COM : Click to view the full PDF of IEC 62016:2003

8.14 Partial EXPRESS-G of master_port_annotate

The following EXPRESS-G diagram is incomplete and does not necessarily include all the relationships of master_port_annotate. It is included for informative purposes only. In case of any discrepancy with the EXPRESS description, the EXPRESS shall take precedence.

The blue boxes are linked to further EXPRESS-G diagrams. The black boxes are, apart from simple EXPRESS types, linked to the EXPRESS description of the object.

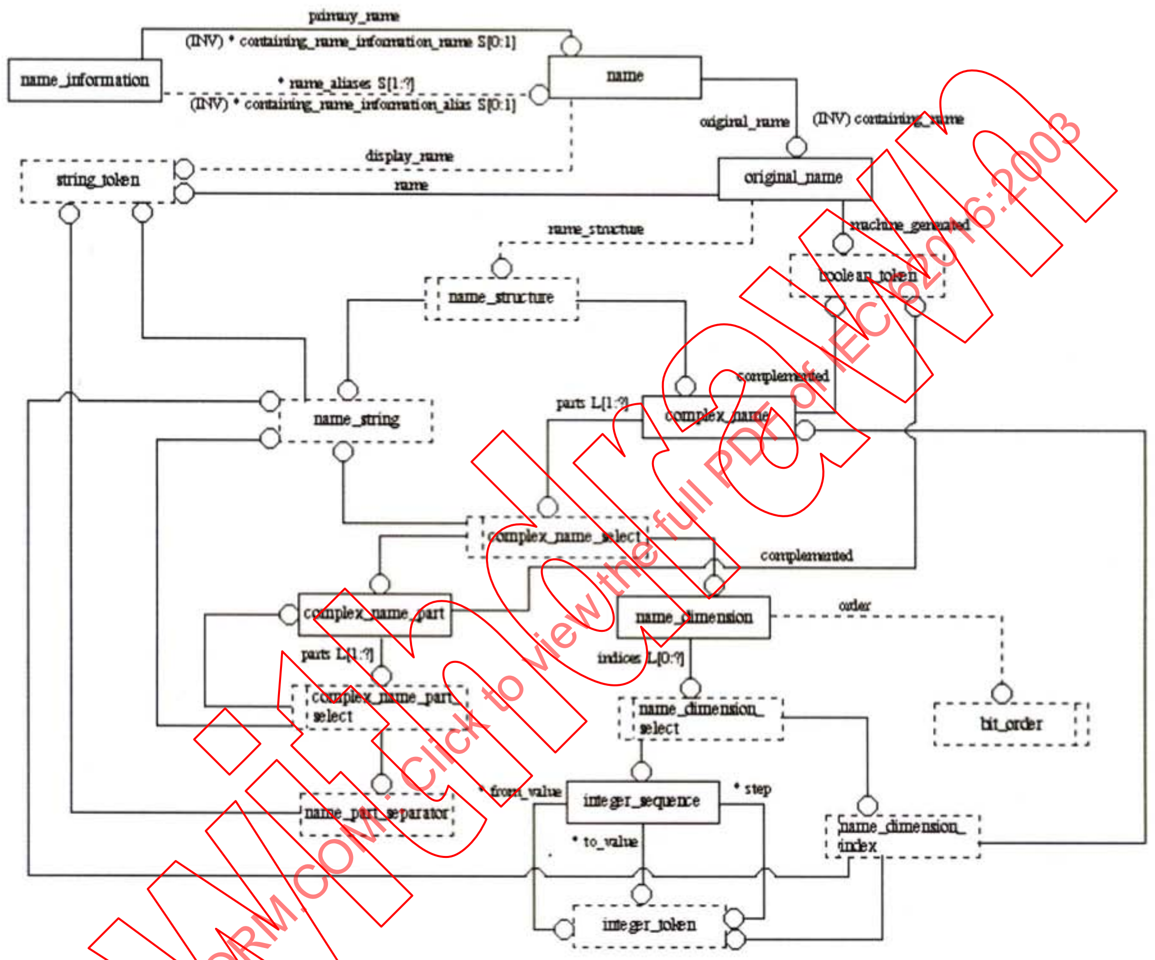

(ABS) property
new_properties S[1:?]
* annotated_port
(ABS) master_ logical_port
(ABS) property_override
* overriding_properties S[1:?]
overriding_designator
string_token
capacitance_value
overriding_ac_load
(INV) * containing_master_port_annotate S[0:1]
(ABS) master_port_annotate
1
overriding_dc_fanout_load
overriding_dc_max_fanin
input_master_ port_annotate
overriding_dc_fanin_load
overriding_dc_max_fanout
output_master_ port_annotate
load_value
overriding_dc_fanin_load
overriding_dc_fanout_load
overriding_dc_max_fanin
overriding_dc_max_fanout
bidirectional_master_ port_annotate
unspecified_direction_ master_port_annotate

Withdrawn
IECNORM.COM : Click to view the full PDF of IEC 62016:2003

8.15 Partial EXPRESS-G of name_information

The following EXPRESS-G diagram is incomplete and does not necessarily include all the relationships of name_information. It is included for informative purposes only. In case of any discrepancy with the EXPRESS description, the EXPRESS shall take precedence.

The blue boxes are linked to further EXPRESS-G diagrams. The black boxes are, apart from simple EXPRESS types, linked to the EXPRESS description of the object.



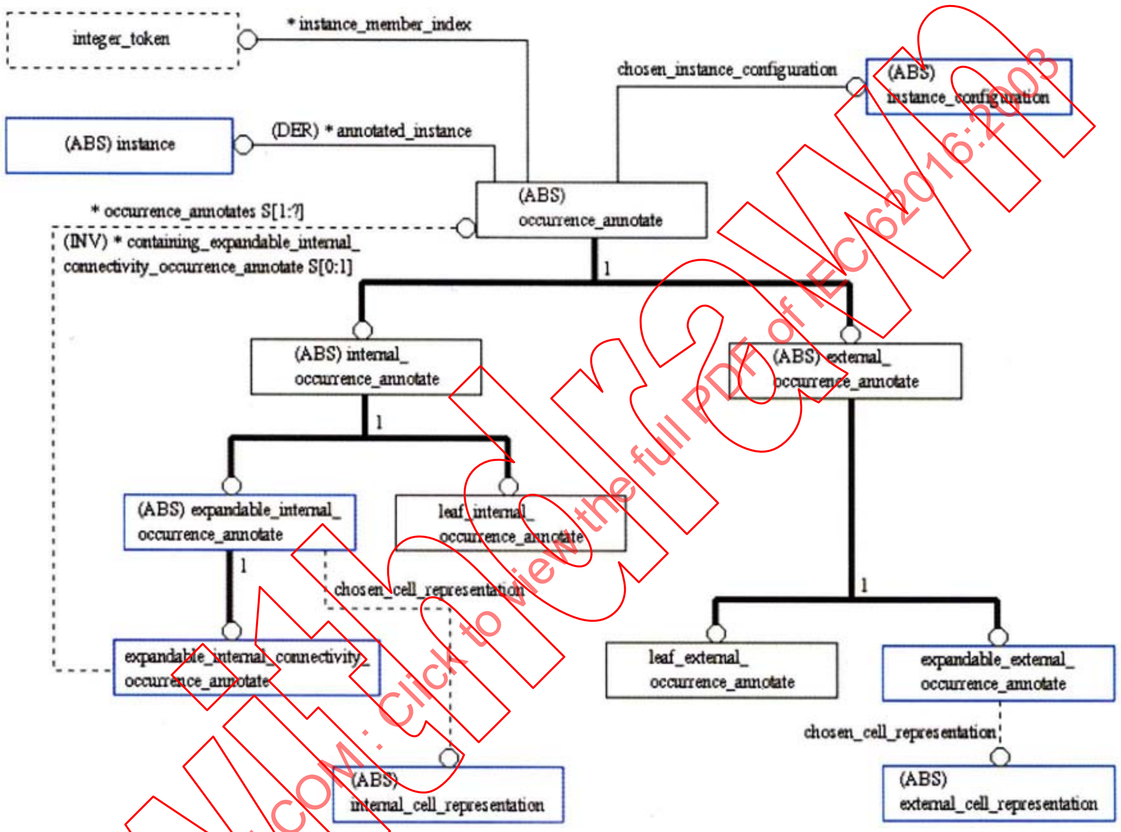
IECNORM.COM : Click to view the full PDF of IEC 62016:2003
Withdrawn

8.16 Partial EXPRESS-G of occurrence_annotate

The following EXPRESS-G diagram is incomplete and does not necessarily include all the relationships of occurrence_annotate. It is included for informative purposes only. In case of any discrepancy with the EXPRESS description, the EXPRESS shall take precedence.

See also occurrence_annotate – 2 and occurrence_annotate – 3.

The blue boxes are linked to further EXPRESS-G diagrams. The black boxes are, apart from simple EXPRESS types, linked to the EXPRESS description of the object.


integer_token
* instance_member_index
chosen_instance_configuration
(ABS) instance_configuration
(ABS) instance
(DER) * annotated_instance
(ABS) occurrence_annotate
* occurrence_annotates S[1:?]
(INV) * containing_expandable_internal_connectivity_occurrence_annotate S[0:1]
1
(ABS) internal_occurrence_annotate
(ABS) external_occurrence_annotate
1
(ABS) expandable_internal_occurrence_annotate
leaf_internal_occurrence_annotate
1
chosen_cell_representation
expandable_internal_connectivity_occurrence_annotate
1
leaf_external_occurrence_annotate
expandable_external_occurrence_annotate
chosen_cell_representation
(ABS) internal_cell_representation
(ABS) external_cell_representation

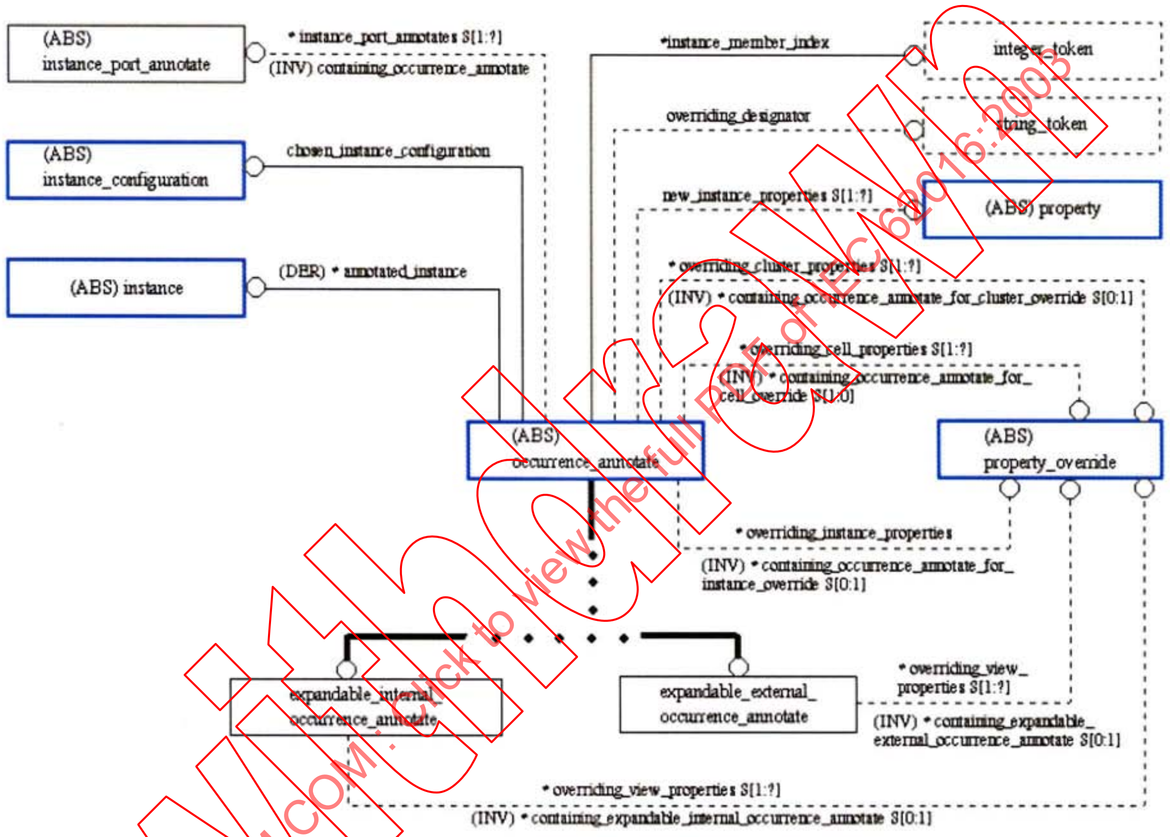
IECNORM.COM : Click to view the full PDF of IEC 62016:2003
Withdrawn

8.17 Partial EXPRESS-G of occurrence_annotate

The following EXPRESS-G diagram is incomplete and does not necessarily include all the relationships of occurrence_annotate. It is included for informative purposes only. In case of any discrepancy with the EXPRESS description, the EXPRESS shall take precedence.

See also occurrence_annotate – 1 and occurrence_annotate – 3.

The blue boxes are linked to further EXPRESS-G diagrams. The black boxes are, apart from simple EXPRESS types, linked to the EXPRESS description of the object.



(ABS) instance_port_annotate
* instance_port_annotates S[1:?]
(INV) containing_occurrence_annotate
(ABS) instance_configuration
chosen_instance_configuration
(ABS) instance
(DER) * annotated_instance
*instance_member_index
integer_token
overriding_designator
string_token
new_instance_properties S[1:?]
(ABS) property
* overriding_cluster_properties S[1:?]
(INV) * containing_occurrence_annotate_for_cluster_override S[0:1]
* overriding_cell_properties S[1:?]
(INV) * containing_occurrence_annotate_for_cell_override S[1:0]
(ABS) occurrence_annotate
(ABS) property_override
* overriding_instance_properties
(INV) * containing_occurrence_annotate_for_instance_override S[0:1]
expandable_internal_occurrence_annotate
expandable_external_occurrence_annotate
* overriding_view_properties S[1:?]
(INV) * containing_expandable_external_occurrence_annotate S[0:1]
* overriding_view_properties S[1:?]
(INV) * containing_expandable_internal_occurrence_annotate S[0:1]

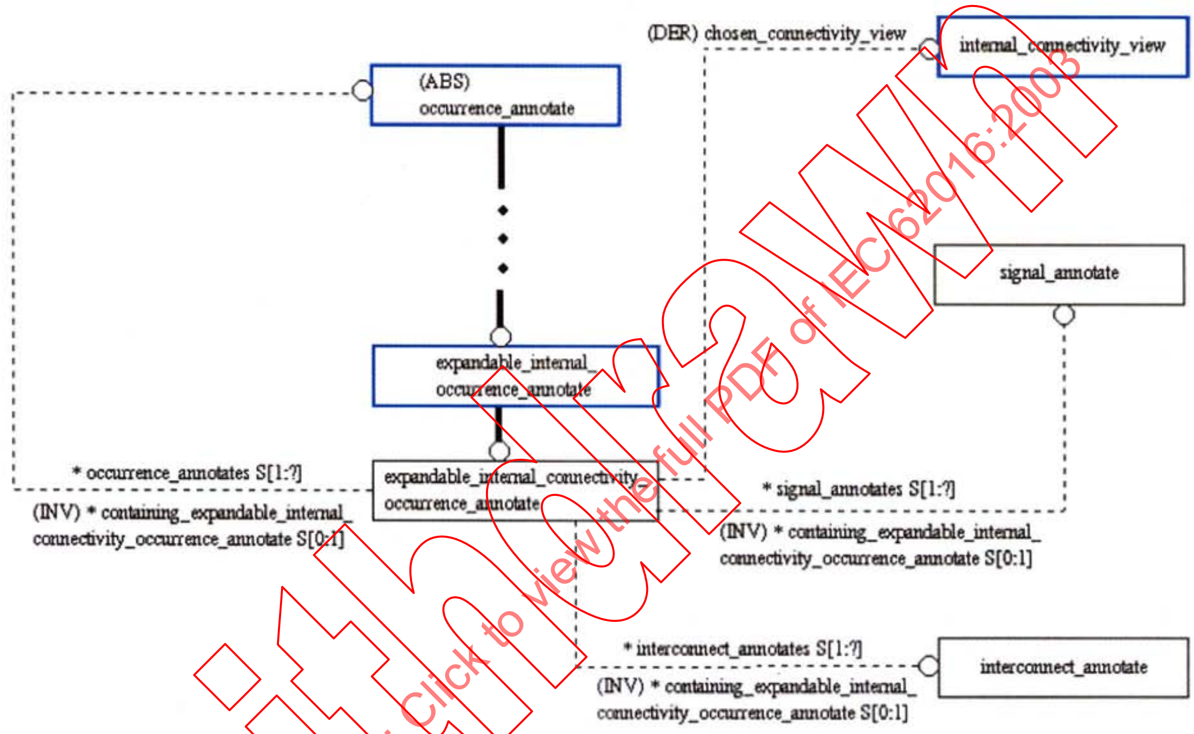
IECNORM.COM : Click to view the full PDF of IEC 62016:2003
Withdrawn

8.18 Partial EXPRESS-G of occurrence_annotate

The following EXPRESS-G diagram is incomplete and does not necessarily include all the relationships of occurrence_annotate. It is included for informative purposes only. In case of any discrepancy with the EXPRESS description, the EXPRESS shall take precedence.

See also occurrence_annotate – 1 and occurrence_annotate – 2.

The blue boxes are linked to further EXPRESS-G diagrams. The black boxes are, apart from simple EXPRESS types, linked to the EXPRESS description of the object.

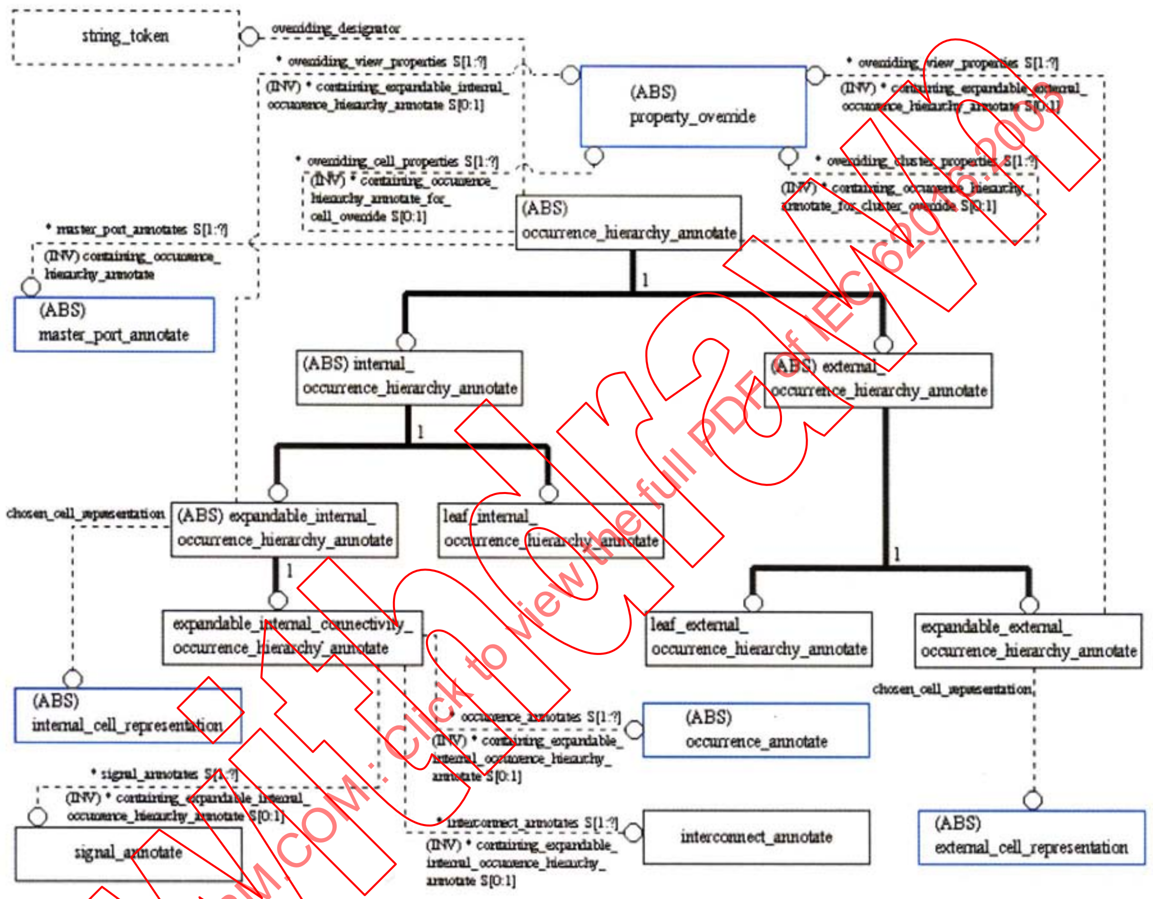

(DER) chosen_connectivity_view
internal_connectivity_view
(ABS)
occurrence_annotate
signal_annotate
expandable_internal_
occurrence_annotate
* occurrence_annotates S[1:?]
expandable_internal_connectivity_
occurrence_annotate
(INV) * containing_expandable_internal_
connectivity_occurrence_annotate S[0:1]
* signal_annotates S[1:?]
(INV) * containing_expandable_internal_
connectivity_occurrence_annotate S[0:1]
* interconnect_annotates S[1:?]
interconnect_annotate
(INV) * containing_expandable_internal_
connectivity_occurrence_annotate S[0:1]

Withdrawn
IECNORM.COM : Click to view the full PDF of IEC 62016:2003

8.19 Partial EXPRESS-G of occurrence_hierarchy_annotate

The following EXPRESS-G diagram is incomplete and does not necessarily include all the relationships of occurrence_hierarchy_annotate. It is included for informative purposes only. In case of any discrepancy with the EXPRESS description, the EXPRESS shall take precedence.

The blue boxes are linked to further EXPRESS-G diagrams. The black boxes are, apart from simple EXPRESS types, linked to the EXPRESS description of the object.

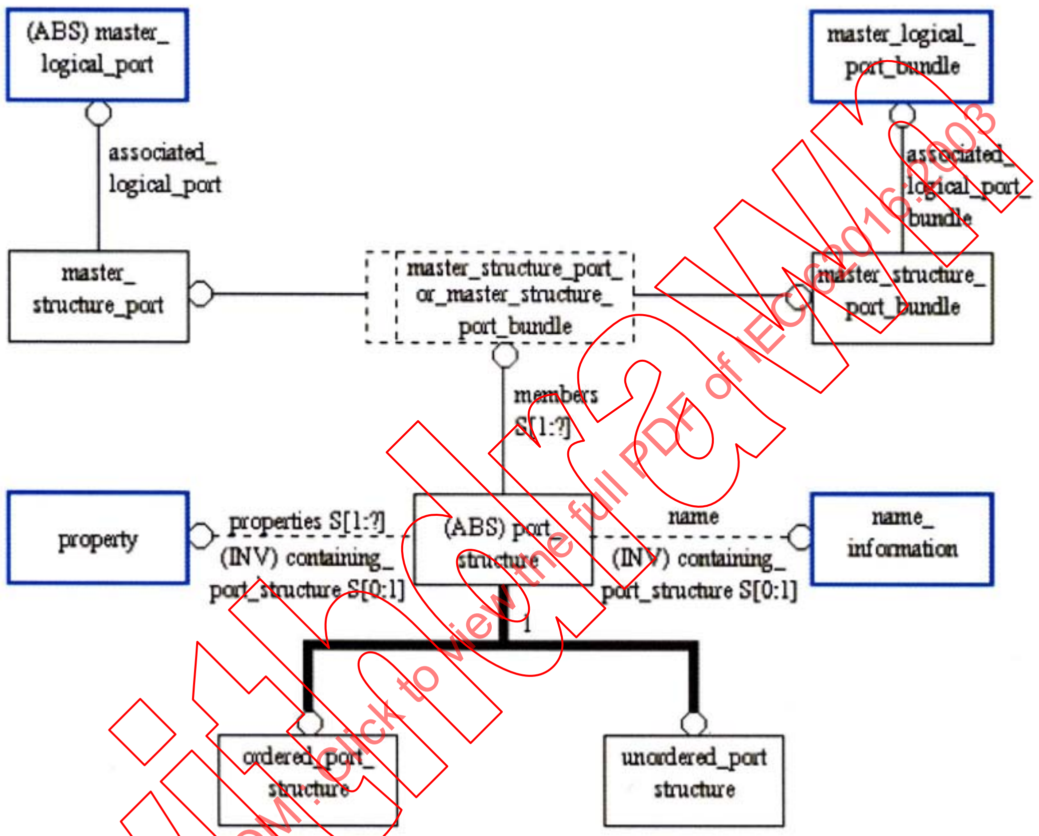

string_token
overriding_designator
* overriding_view_properties S[1:?]
(INV) * containing_expandable_internal_occurrence_hierarchy_annotate S[0:1]
(ABS) property_override
* overriding_view_properties S[1:?]
(INV) * containing_expandable_external_occurrence_hierarchy_annotate S[0:1]
* overriding_cell_properties S[1:?]
(INV) * containing_occurrence_hierarchy_annotate_for_cell_override S[0:1]
* overriding_cluster_properties S[1:?]
(INV) * containing_occurrence_hierarchy_annotate_for_cluster_override S[0:1]
* master_port_annotates S[1:?]
(INV) containing_occurrence_hierarchy_annotate
(ABS) occurrence_hierarchy_annotate
(ABS) master_port_annotate
1
(ABS) internal_occurrence_hierarchy_annotate
(ABS) external_occurrence_hierarchy_annotate
1
chosen_cell_representation
(ABS) expandable_internal_occurrence_hierarchy_annotate
leaf_internal_occurrence_hierarchy_annotate
1
1
expandable_internal_connectivity_occurrence_hierarchy_annotate
leaf_external_occurrence_hierarchy_annotate
expandable_external_occurrence_hierarchy_annotate
chosen_cell_representation
(ABS) internal_cell_representation
* occurrence_annotates S[1:?]
(INV) * containing_expandable_internal_occurrence_hierarchy_annotate S[0:1]
(ABS) occurrence_annotate
* signal_annotates S[1:?]
(INV) * containing_expandable_internal_occurrence_hierarchy_annotate S[0:1]
signal_annotate
* interconnect_annotates S[1:?]
(INV) * containing_expandable_internal_occurrence_hierarchy_annotate S[0:1]
interconnect_annotate
(ABS) external_cell_representation

IECNORM.COM : Click to view the full PDF of IEC 62016:2003
Withdrawn

8.20 Partial EXPRESS-G of port_structure

The following EXPRESS-G diagram is incomplete and does not necessarily include all the relationships of port_structure. It is included for informative purposes only. In case of any discrepancy with the EXPRESS description, the EXPRESS shall take precedence.

The blue boxes are linked to further EXPRESS-G diagrams. The black boxes are, apart from simple EXPRESS types, linked to the EXPRESS description of the object.

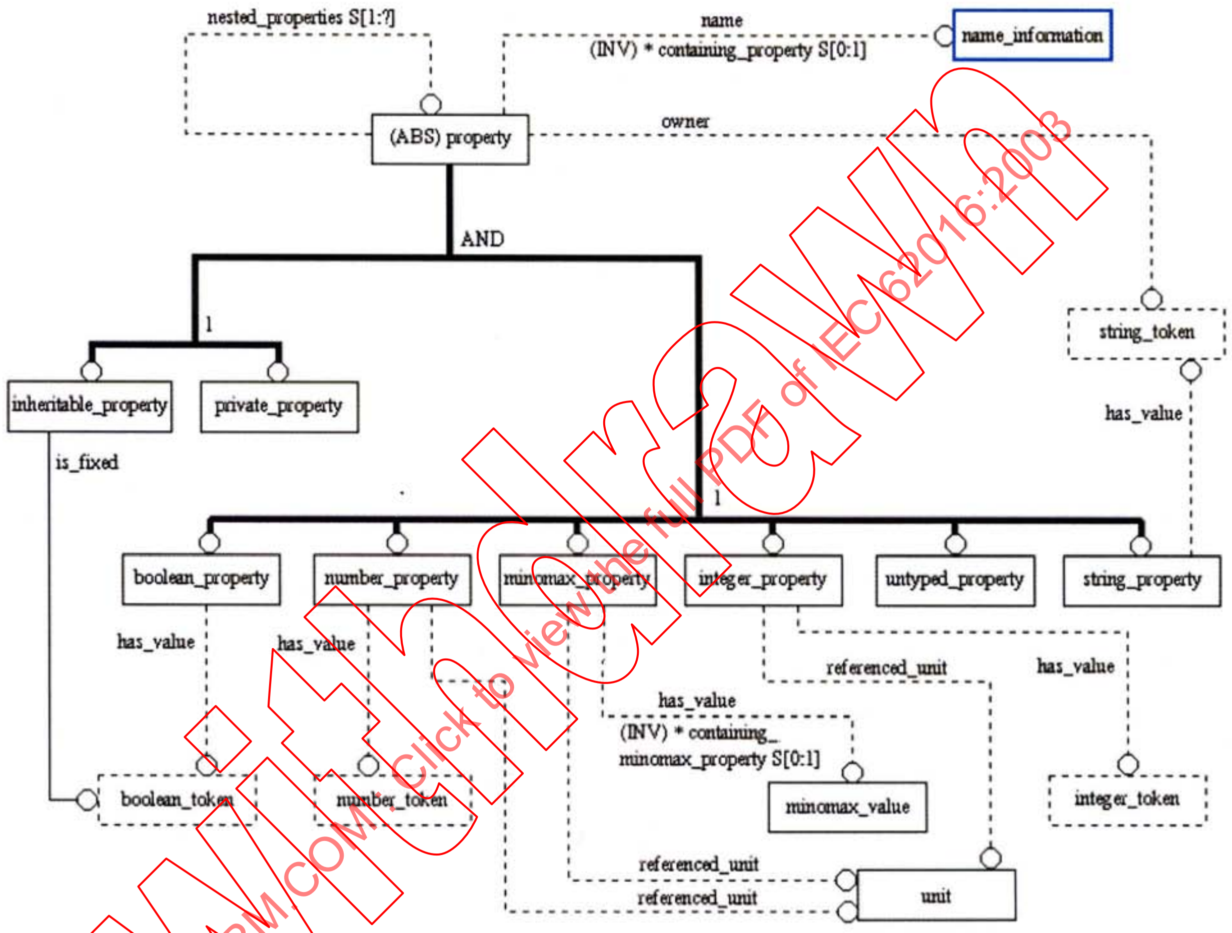

(ABS) master_ logical_port
associated_ logical_port
master_ structure_port
master_structure_port_ or_master_structure_ port_bundle
master_logical_ port_bundle
associated_ logical_port_ bundle
master_structure_ port_bundle
members S[1:?]
property
properties S[1:?]
(INV) containing_ port_structure S[0:1]
(ABS) port_ structure
name
(INV) containing_ port_structure S[0:1]
name_ information
1
ordered_port_ structure
unordered_port structure

IECNORM.COM : Click to view the full PDF of IEC 62016:2003
Withdrawn

8.21 Partial EXPRESS-G of property

The following EXPRESS-G diagram is incomplete and does not necessarily include all the relationships of property. It is included for informative purposes only. In case of any discrepancy with the EXPRESS description, the EXPRESS shall take precedence.

The blue boxes are linked to further EXPRESS-G diagrams. The black boxes are, apart from simple EXPRESS types, linked to the EXPRESS description of the object.

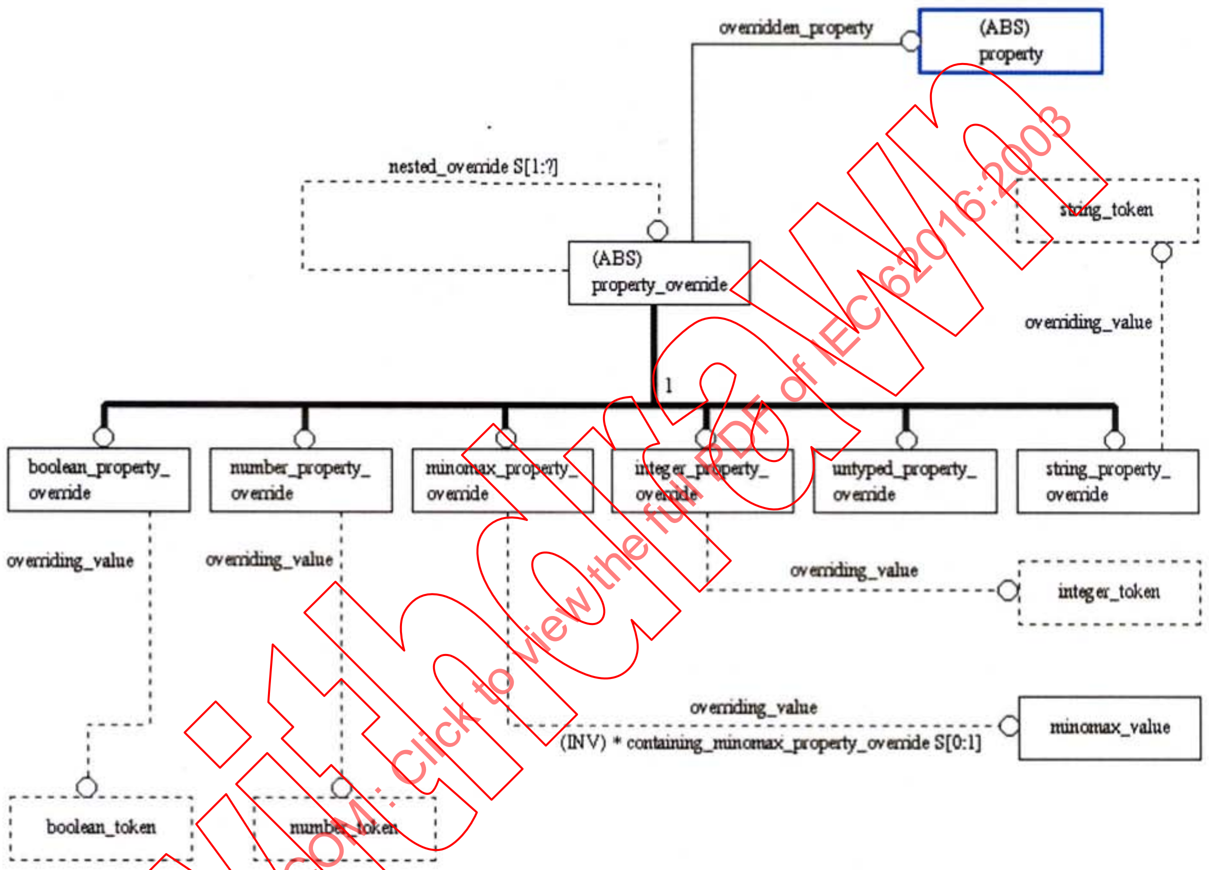


IECNORM.COM : Click to view the full PDF of IEC 62016:2003
Withdrawn

8.22 Partial EXPRESS-G of property_override

The following EXPRESS-G diagram is incomplete and does not necessarily include all the relationships of property_override. It is included for informative purposes only. In case of any discrepancy with the EXPRESS description, the EXPRESS shall take precedence.

The blue boxes are linked to further EXPRESS-G diagrams. The black boxes are, apart from simple EXPRESS types, linked to the EXPRESS description of the object.

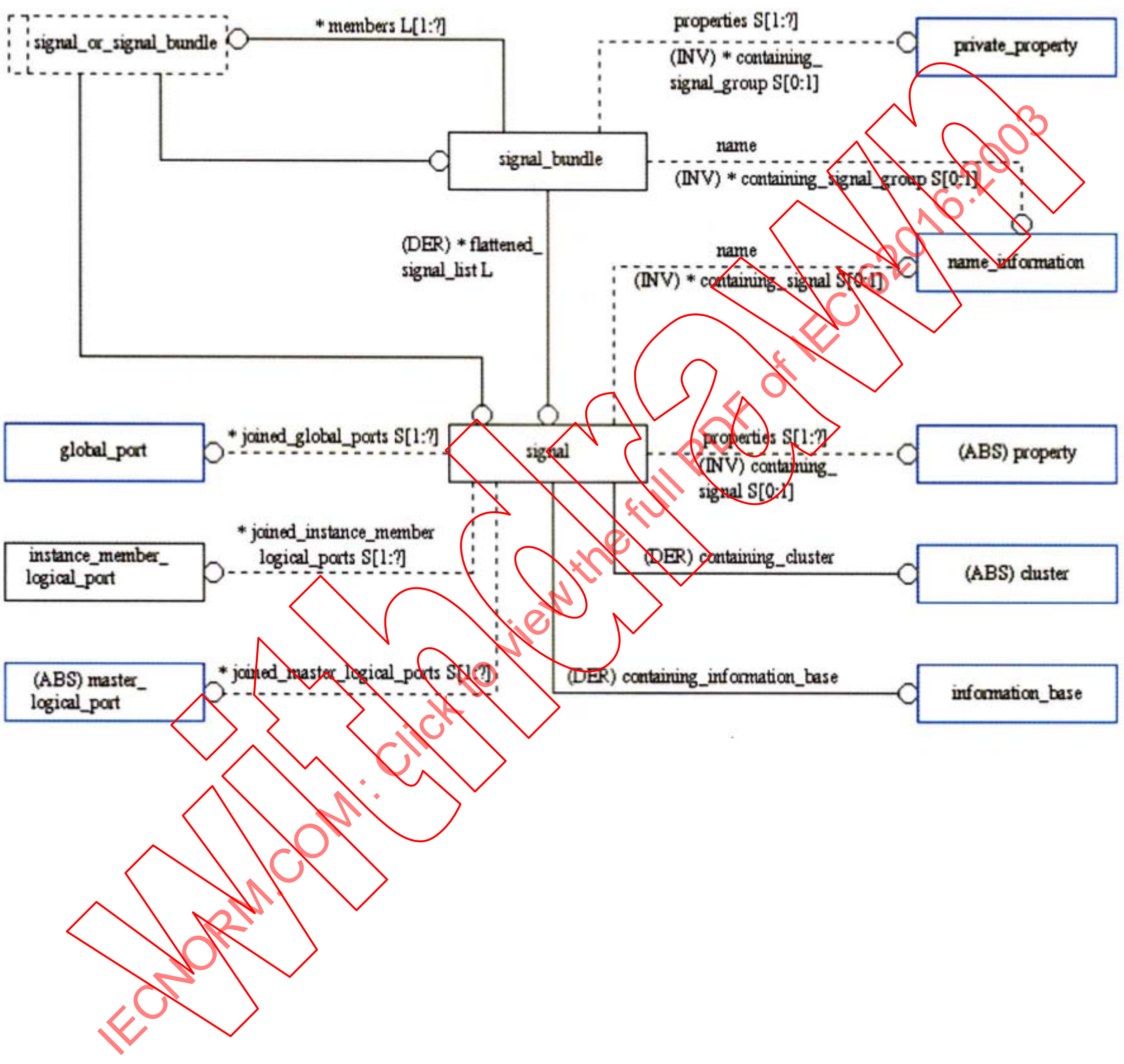


Withdrawn
IECNORM.COM: Click to view the full PDF of IEC 62016:2003

8.23 Partial EXPRESS-G of signal

The following EXPRESS-G diagram is incomplete and does not necessarily include all the relationships of signal. It is included for informative purposes only. In case of any discrepancy with the EXPRESS description, the EXPRESS shall take precedence.

The blue boxes are linked to further EXPRESS-G diagrams. The black boxes are, apart from simple EXPRESS types, linked to the EXPRESS description of the object.


signal_or_signal_bundle
* members L[1:?]
properties S[1:?]
(INV) * containing_ signal_group S[0:1]
private_property
signal_bundle
name
(INV) * containing_signal_group S[0:1]
(DER) * flattened_ signal_list L
name
(INV) * containing_signal S[0:1]
name_information
global_port
* joined_global_ports S[1:?]
signal
properties S[1:?]
(INV) containing_ signal S[0:1]
(ABS) property
instance_member_ logical_port
* joined_instance_member logical_ports S[1:?]
(DER) containing_cluster
(ABS) cluster
(ABS) master_ logical_port
* joined_master_logical_ports S[1:?]
(DER) containing_information_base
information_base

Withdrawn
IECNORM.COM : Click to view the full PDF of IEC 62016:2003

9 Core Model schemas

This section defines the EXPRESS schemas of the information model of the Core Model of the electronics domain.

9.1 connectivity_structure_model

```
SCHEMA connectivity_structure_model;

  REFERENCE FROM connectivity_view_model
    (internal_connectivity_view);

  REFERENCE FROM documentation_model
    (documentation);

  REFERENCE FROM hierarchy_model
    (instance_member_structure_port,
     instance_member_logical_port,
     instance_structure_port,
     instance_structure_port_bundle,
     local_master_port_bundle,
     master_structure_port,
     master_logical_port,
     master_structure_port_bundle);

  REFERENCE FROM information_base_model
    (global_port,
     global_port_bundle);

  REFERENCE FROM logical_connectivity_model
    (signal,
     signal_bundle);

  REFERENCE FROM support_definition_model
    (integer_token,
     name_information,
     not_exists,
     property,
     there_exists);
```

- check_structural_connectivity (function)
- check_valid_joined_instance_ports (function)
- connectivity_bus (entity)
- connectivity_bus_slice (entity)
- connectivity_generic_bus (entity)
- connectivity_generic_net (entity)
- connectivity_net (entity)
- connectivity_ripper (entity)
- connectivity_sub_bus (entity)
- connectivity_sub_net (entity)

```
END_SCHEMA;
```

9.1.1 Description

The connectivity_structure_model schema describes the structural connectivity information of a connectivity view. That includes definitions of connectivity_bus, connectivity_net and connectivity_ripper.

Withdrawn
IECNORM.COM : Click to view the full PDF of IEC 62016:2003

9.2 connectivity_view_model

```
SCHEMA connectivity_view_model;

  REFERENCE FROM hierarchy_model
    (external_cell_representation,
     expandable_internal_cluster_configuration,
     expandable_internal_connectivity_view_configuration,
     instance,
     internal_cell_representation,
     local_master_port_bundle,
     port_structure);

  REFERENCE FROM logical_connectivity_model
    (signal,
     signal_bundle);

  REFERENCE FROM connectivity_structure_model
    (connectivity_bus,
     connectivity_net,
     connectivity_ripper);

  REFERENCE FROM support_definition_model
    (capacitance_unit,
     name_information,
     not_exists,
     property,
     property_override,
     there_exists);
```

- connectivity_instance_implementation (entity)
- connectivity_units (entity)
- external_connectivity_view (entity)
- internal_connectivity_view (entity)

```
END_SCHEMA;
```

9.2.1 Description

The connectivity_view_model schema describes what information can be found in a connectivity view. The connectivity view conveys hierarchical and connectivity information. The hierarchy information describes how a circuit is broken down into a number of sub-circuits. The connectivity information defines how circuits are connected.

Withdrawn
IECNORM.COM : Click to view the full PDF of IEC 62016:2003

9.3 design_hierarchy_model

```
SCHEMA design_hierarchy_model;

  REFERENCE FROM connectivity_structure_model
    (connectivity_generic_net,
     connectivity_generic_bus);

  REFERENCE FROM connectivity_view_model
    (internal_connectivity_view);

  REFERENCE FROM design_management_model
    (copyright,
     written);

  REFERENCE FROM documentation_model
    (documentation);

  REFERENCE FROM hierarchy_model
    (bidirectional_master_logical_port,
     cell,
     cell_representation,
     cluster_configuration,
     expandable_external_cluster_configuration,
     expandable_external_instance_configuration,
     expandable_internal_cluster_configuration,
     expandable_internal_connectivity_instance_configuration,
     expandable_internal_connectivity_view_configuration,
     expandable_internal_instance_configuration,
     external_cell_representation,
     external_cluster_configuration,
     external_cluster_instance,
     external_instance_configuration,
     find_instance_port_attributes,
     input_master_logical_port,
     instance,
     instance_configuration,
     instance_port_attributes,
     instance_port_bundle_attributes,
     instance_structure_port,
     internal_cell_representation,
     internal_cluster_configuration,
     internal_cluster_instance,
     internal_instance_configuration,
     leaf_external_cluster_configuration,
     leaf_external_instance_configuration,
     leaf_internal_cluster_configuration,
     leaf_internal_instance_configuration,
     load_value,
     master_logical_port,
     output_master_logical_port,
     referenced_interconnect,
     unspecified_direction_master_logical_port);

  REFERENCE FROM information_base_model
    (information_base);

  REFERENCE FROM logical_connectivity_model
    (signal);

  REFERENCE FROM support_definition_model
    (capacitance_unit,
```

IECNORM.COM : Click to view the full PDF of IEC 62016:2003
Withdrawn

```
      capacitance_value,
      integer_token,
      name_information,
      not_exists,
      string_token,
      private_property,
      property,
      property_override,
      there_exists);
```

- bidirectional_instance_port_annotate (entity)
- bidirectional_master_port_annotate (entity)
- design (entity)
- design_hierarchy (entity)
- design_units (entity)
- expandable_external_design_hierarchy (entity)
- expandable_external_occurrence_annotate (entity)
- expandable_external_occurrence_hierarchy_annotate (entity)
- expandable_internal_connectivity_design_hierarchy (entity)
- expandable_internal_connectivity_occurrence_annotate (entity)
- expandable_internal_connectivity_occurrence_hierarchy_annotate (entity)
- expandable_internal_design_hierarchy (entity)
- expandable_internal_occurrence_annotate (entity)
- expandable_internal_occurrence_hierarchy_annotate (entity)
- external_design_hierarchy (entity)
- external_occurrence_annotate (entity)
- external_occurrence_hierarchy_annotate (entity)
- input_instance_port_annotate (entity)
- input_master_port_annotate (entity)
- instance_port_annotate (entity)
- interconnect_annotate (entity)
- internal_design_hierarchy (entity)
- internal_occurrence_annotate (entity)
- internal_occurrence_hierarchy_annotate (entity)
- leaf_external_design_hierarchy (entity)
- leaf_external_occurrence_annotate (entity)
- leaf_external_occurrence_hierarchy_annotate (entity)
- leaf_internal_design_hierarchy (entity)
- leaf_internal_occurrence_annotate (entity)
- leaf_internal_occurrence_hierarchy_annotate (entity)
- master_port_annotate (entity)
- occurrence_annotate (entity)
- occurrence_hierarchy_annotate (entity)
- output_instance_port_annotate (entity)
- output_master_port_annotate (entity)
- signal_annotate (entity)
- unspecified_direction_instance_port_annotate (entity)
- unspecified_direction_master_port_annotate (entity)

```
END_SCHEMA;
```

9.3.1 Description

The design_hierarchy_model schema describes the annotation data relating to an occurrence hierarchy. A design_hierarchy is defined by choosing the top cluster_configuration. It is annotated by an occurrence_hierarchy_annotate. Interconnects and master ports within the top occurrence can be annotated by interconnect_annotate and master_port_annotate respectively. If further annotation is to be given for occurrences further down in the occurrence hierarchy, occurrence_annotate specify which occurrence is to be annotated.

IECNORM.COM : Click to view the full PDF of IEC 62016:2003 Withdrawn

Similarly, interconnects and instance ports within an occurrence can be annotated by interconnect_annotate and instance_port_annotate respectively.

9.4 design_management_model

```
SCHEMA design_management_model;

  REFERENCE FROM documentation_model
    (documentation);

  REFERENCE FROM hierarchy_model
    (cell,
     cell_representation,
     cluster);

  REFERENCE FROM information_base_model
    (information_base);

  REFERENCE FROM library_model
    (library);

  REFERENCE FROM support_definition_model
    (integer_token,
     positive_integer_token,
     private_property,
     string_token);

  REFERENCE FROM design_hierarchy_model
    (design);
```

- copyright (entity)
- date (entity)
- time (entity)
- time_stamp (entity)
- valid_date (function)
- version_information (entity)
- written (entity)

```
END_SCHEMA;
```

9.4.1 Description

The design_management_model schema provides the design management information. It records a history of modifications. It provides the information needed to trace back to the origin or the owner of the data, and also identifies the software or program name which was responsible for creating the data.

Withdrawn
IECNORM.COM : Click to view the full PDF of IEC 62016:2003

9.5 documentation_model

```
SCHEMA documentation_model;

  REFERENCE FROM connectivity_structure_model
    (connectivity_generic_bus,
     connectivity_generic_net);

  REFERENCE FROM design_management_model
    (copyright,
     written);

  REFERENCE FROM hierarchy_model
    (cell,
     cell_representation,
     cluster,
     cell_representation_set);

  REFERENCE FROM information_base_model
    (information_base);

  REFERENCE FROM library_model
    (library);

  REFERENCE FROM design_hierarchy_model
    (design);

  REFERENCE FROM support_definition_model
    (string_token);
```

- documentation (entity)
- section (entity)
- section_element (type)

```
END_SCHEMA;
```

9.5.1 Description

The documentation_model schema describes the documentation provided for an object. A documentation may consists of several sections. Each section may have text and nested sections.

withdrawn
IECNORM.COM : Click to view the full PDF of IEC 62016:2003

9.6 hierarchy_model

```
SCHEMA hierarchy_model;

  REFERENCE FROM connectivity_structure_model
    (connectivity_generic_bus,
     connectivity_generic_net);

  REFERENCE FROM logical_connectivity_model
    (signal);

  REFERENCE FROM connectivity_view_model
    (external_connectivity_view,
     internal_connectivity_view);

  REFERENCE FROM design_management_model
    (copyright,
     time_stamp,
     written);

  REFERENCE FROM documentation_model
    (documentation);

  REFERENCE FROM information_base_model
    (global_port);

  REFERENCE FROM library_model
    (external_library,
     internal_library,
     library);

  REFERENCE FROM support_definition_model
    (boolean_token,
     capacitance_unit,
     capacitance_value,
     integer_token,
     name_information,
     not_exists,
     number_token,
     positive_integer_token,
     private_property,
     property,
     property_override,
     string_token,
     there_exists);
```

- bidirectional_instance_port_attributes (entity)
- bidirectional_master_logical_port (entity)
- cell (entity)
- cell_representation (entity)
- cell_representation_set (entity)
- check_derived_recursion (function)
- check_non_recursive_member_cell_representation_sets (function)
- check_previous_recursion (function)
- check_valid_port_structure_members (function)
- check_valid_status_in_previous_version (function)
- cluster (entity)
- cluster_configuration (entity)
- cluster_interface (entity)
- cluster_interface_units (entity)
- compare_date (function)

Withdrawn
IECNORM.COM : Click to view the full PDF of IEC 62016:2003

- expandable_external_cluster_configuration (entity)
- expandable_external_instance_configuration (entity)
- expandable_internal_cluster_configuration (entity)
- expandable_internal_connectivity_instance_configuration (entity)
- expandable_internal_connectivity_view_configuration (entity)
- expandable_internal_instance_configuration (entity)
- external_cell (entity)
- external_cell_representation (entity)
- external_cluster (entity)
- external_cluster_configuration (entity)
- external_cluster_instance (entity)
- external_instance_configuration (entity)
- find_instance_port_attributes (function)
- flatten_instance_structure_port (function)
- flatten_local_master_port_bundle (function)
- flatten_master_logical_port_bundle (function)
- input_instance_port_attributes (entity)
- input_master_logical_port (entity)
- instance (entity)
- instance_configuration (entity)
- instance_member_logical_port (entity)
- instance_member_structure_port (entity)
- instance_port_attributes (entity)
- instance_port_bundle_attributes (entity)
- instance_structure_port (entity)
- instance_structure_port_bundle (entity)
- internal_cell (entity)
- internal_cell_representation (entity)
- internal_cluster (entity)
- internal_cluster_configuration (entity)
- internal_cluster_instance (entity)
- internal_instance_configuration (entity)
- joined_master_logical_port_set (entity)
- leaf_external_cluster_configuration (entity)
- leaf_external_instance_configuration (entity)
- leaf_internal_cluster_configuration (entity)
- leaf_internal_instance_configuration (entity)
- load_value (type)
- local_master_port_bundle (entity)
- master_logical_port (entity)
- master_logical_port_bundle (entity)
- master_logical_port_or_master_logical_port_bundle (type)
- master_structure_port (entity)
- master_structure_port_bundle (entity)
- master_structure_port_or_master_structure_port_bundle (type)
- must_join_port_set (entity)
- non_permutable_master_logical_port_set (entity)
- non_permutable_relationship (entity)
- non_permutable_structure (entity)
- non_permutable_structure_size (function)
- ordered_port_structure (entity)
- output_instance_port_attributes (entity)
- output_master_logical_port (entity)
- permutable_master_port_set (entity)
- permutable_relationship (entity)
- permutable_structure (entity)
- permutable_structure_equal_member_size (function)
- permutable_structure_size (function)

IECNORM.COM : Click to view the full PDF of IEC 62016:2003
Withdrawn

- port_structure (entity)
- referenced_interconnect (type)
- related_cell_representation (entity)
- unconfigured_instance (entity)
- unordered_port_structure (entity)
- unspecified_direction_instance_port_attributes (entity)
- unspecified_direction_master_logical_port (entity)
- weak_joined_port_set (entity)

```
END_SCHEMA;
```

9.6.1 Description

The hierarchy_model schema describes the hierarchical information of a cell which is the basic unit of design. A cell can have multiple views of the same type. Views which share the same interface are grouped into a cluster. Views may also be grouped into a cell_representation_set to indicate a particularly close relationship to each other. In the information base clusters are instantiated within other views.

Withdrawn
IECNORM.COM : Click to view the full PDF of IEC 62016:2003

9.7 information_base_model

```
SCHEMA information_base_model;

  REFERENCE FROM connectivity_view_model
    (connectivity_units);

  REFERENCE FROM design_hierarchy_model
    (design);

  REFERENCE FROM design_management_model
    (copyright,
     written);

  REFERENCE FROM documentation_model
    (documentation);

  REFERENCE FROM hierarchy_model
    (cluster_interface_units);

  REFERENCE FROM library_model
    (library);

  REFERENCE FROM support_definition_model
    (boolean_token,
     name_information,
     not_exists,
     private_property,
     string_token,
     unit);
```

- character_encoding (type)
- check_valid_member_ports (function)
- flatten_global_port_bundle (function)
- global_port (entity)
- global_port_bundle (entity)
- global_port_or_global_port_bundle (type)
- information_base (entity)
- only_one_information_base (rule)

```
END_SCHEMA;
```

9.7.1 Description

The information_base_model schema describes the information held by an information base. An information base may contain several designs and libraries of cell definitions. In addition, it specifies the interpretation of the data contained in quoted strings.

Withdrawn
IECNORM.COM : Click to view the full PDF of IEC 62016:2003

9.8 library_model

```
SCHEMA library_model;

  REFERENCE FROM design_management_model
    (copyright,
     written);

  REFERENCE FROM documentation_model
    (documentation);

  REFERENCE FROM hierarchy_model
    (cell,
     external_cell,
     internal_cell);

  REFERENCE FROM information_base_model
    (information_base);

  REFERENCE FROM support_definition_model
    (name_information,
     not_exists,
     private_property,
     property,
     unit);
```

- external_library (entity)
- internal_library (entity)
- library (entity)

```
END_SCHEMA;
```

9.8.1 Description

The library_model schema describes what information can be found in an library. A library is a grouping of reusable objects. A library contains cell definitions.

Withdrawn
IECNORM.COM : Click to view the full PDF of IEC 62016:2003

9.9 logical_connectivity_model

```
SCHEMA logical_connectivity_model;

  REFERENCE FROM connectivity_view_model
    (internal_connectivity_view);

  REFERENCE FROM hierarchy_model
    (cell_representation,
     cluster,
     find_instance_port_attributes,
     instance_member_logical_port,
     instance_port_attributes,
     master_logical_port);

  REFERENCE FROM information_base_model
    (global_port,
     information_base);

  REFERENCE FROM support_definition_model
    (name_information,
     not_exists,
     private_property,
     property);
```

- flatten_signal_bundle (function)
- is_unused_externally (function)
- not_recursive_signal_bundle (function)
- signal (entity)
- signal_bundle (entity)
- signal_or_signal_bundle (type)

```
END_SCHEMA;
```

9.9.1 Description

The logical_connectivity_model schema describes what logical connectivity information can be found in an information base. The logical connectivity of a view may include signals and signal_bundles. A signal is defined as a named object which lists all the global_ports, instance_member_logical_ports and master_logical_ports which are electrically common at a given level of hierarchy. A signal_bundle provides a grouping mechanism for signals. It does not modify connectivity.

Withdrawn
IECNORM.COM : Click to view the full PDF of IEC 62016:2003

9.10 support_definition_model

```
SCHEMA support_definition_model;

  REFERENCE FROM connectivity_structure_model
    (connectivity_generic_bus,
     connectivity_generic_net);

  REFERENCE FROM connectivity_view_model
    (connectivity_units,
     connectivity_instance_implementation);

  REFERENCE FROM design_hierarchy_model
    (design,
     design_hierarchy,
     design_units,
     expandable_external_occurrence_annotate,
     expandable_external_occurrence_hierarchy_annotate,
     expandable_internal_occurrence_annotate,
     expandable_internal_occurrence_hierarchy_annotate,
     instance_port_annotate,
     interconnect_annotate,
     master_port_annotate,
     occurrence_annotate,
     occurrence_hierarchy_annotate,
     signal_annotate);

  REFERENCE FROM hierarchy_model
    (cell,
     cell_representation,
     cluster,
     cluster_configuration,
     cluster_interface_units,
     instance,
     instance_port_attributes,
     instance_port_bundle_attributes,
     local_master_port_bundle,
     master_structure_port,
     master_structure_port_bundle,
     master_logical_port,
     master_logical_port_bundle,
     port_structure,
     cell_representation_set);

  REFERENCE FROM information_base_model
    (global_port,
     global_port_bundle,
     information_base);

  REFERENCE FROM library_model
    (library);

  REFERENCE FROM logical_connectivity_model
    (signal,
     signal_bundle);
```

- bit_order (type)
- boolean_property (entity)
- boolean_property_override (entity)
- boolean_token (type)
- capacitance_unit (entity)

Withdrawn IECNORM.COM : Click to view the full PDF of IEC 62016:2003

- capacitance_value (entity)
- complex_name (entity)
- complex_name_part (entity)
- complex_name_part_select (type)
- complex_name_select (type)
- exponent_value (type)
- fraction (entity)
- inheritable_property (entity)
- integer_property (entity)
- integer_property_override (entity)
- integer_sequence (entity)
- integer_token (type)
- minomax_property (entity)
- minomax_property_override (entity)
- minomax_value (entity)
- name (entity)
- name_dimension (entity)
- name_dimension_index (type)
- name_dimension_select (type)
- name_information (entity)
- name_part_separator (type)
- name_string (type)
- name_structure (type)
- non_negative_capacitance (function)
- not_exists (function)
- not_fixed (type)
- not_greater_than (function)
- number_or_not_fixed (type)
- number_property (entity)
- number_property_override (entity)
- number_token (type)
- original_name (entity)
- physical_dimension (entity)
- physical_unit (type)
- positive_integer_token (type)
- private_property (entity)
- property (entity)
- property_override (entity)
- string_property (entity)
- string_property_override (entity)
- string_token (type)
- there_exists (function)
- unit (entity)
- untyped_property (entity)
- untyped_property_override (entity)
- valid_range (function)
- valid_unit_type (function)

```
END_SCHEMA;
```

9.10.1 Description

The support_definition_model schema contains general information which applies to other schemas. Typical examples are the entities name_information and unit which are not confined to individual schemas. In addition, it holds definitions of some generic functions whose sole purpose is to clarify and improve understanding of constraints written in EXPRESS.

Withdrawn
IECNORM.COM : Click to view the full PDF of IEC 62016:2003

10 Core Model information model

This section defines all of the objects comprising the EXPRESS information model of the Core Model of the electronics domain.

10.1 connectivity_structure_model

Description

The connectivity_structure_model schema describes the structural connectivity information of a connectivity view. That includes definitions of connectivity_bus, connectivity_net and connectivity_ripper.

10.1.1 connectivity_generic_net

```
  ENTITY connectivity_generic_net
    ABSTRACT SUPERTYPE OF (ONEOF(connectivity_net,
                                 connectivity_sub_net));
    associated_signal     : signal;                          --
reference
    criticality           : integer_token;
    document              : OPTIONAL SET [1:?] OF documentation;
    joined_global_ports                                       --
reference
                          : OPTIONAL SET [1:?] OF global_port;
    joined_instance_member_structure_ports
                          : OPTIONAL SET [1:?] OF
                            instance_member_structure_port;
    joined_instance_structure_ports
                         : OPTIONAL SET [1:?] OF
instance_structure_port;
    joined_master_structure_ports                              --
reference
                          : OPTIONAL SET [1:?] OF
master_structure_port;
    joined_connectivity_rippers
                         : OPTIONAL SET [1:?] OF connectivity_ripper;
-- reference
    name                  : OPTIONAL name_information;
    properties            : OPTIONAL SET [1:?] OF property;
    sub_nets              : OPTIONAL SET [1:?] OF
connectivity_sub_net;
    containing_connectivity_view                              --
reference
                          : internal_connectivity_view;
                                                     -- derived in
subtypes
  WHERE
    valid_sub_nets :
      (* No two connectivity_sub_nets refer to the same
         set of ports and rippers. *)
      not_exists(QUERY(net1 <* sub_nets |
        there_exists(QUERY(net2 <* sub_nets – net1 |
          (NVL(net1.joined_global_ports,[]) =
NVL(net2.joined_global_ports,[]))
          AND (NVL(net1.joined_instance_member_structure_ports, []) =
               NVL (net2.joined_instance_member_structure_ports, []))
          AND (NVL (net1.joined_instance_structure_ports, []) =
               NVL (net2.joined_instance_structure_ports, []))
          AND (NVL (net1.joined_master_structure_ports, []) =
               NVL (net2.joined_master_structure_ports, []))
```

IECNORM.COM: Click to view the full PDF of IEC 62016:2003
Withdrawn

```
          AND (NVL (net1.joined_connectivity_rippers, []) =
               NVL (net2.joined_connectivity_rippers, []))))));

    unique_document :
      (* No documents have the same contents. *)
      value_unique(document);
  END_ENTITY;
```

10.1.1.1 Description

A connectivity_generic_net is either a connectivity_net or a connectivity_sub_net. A connectivity_generic_net is used in the model because of the similarity between a connectivity_net and a connectivity_sub_net. A connectivity_net may be divided into connectivity_sub_nets and a connectivity_sub_net itself may also be divided into connectivity_sub_nets. No two connectivity_sub_nets at a given level join the same set of ports and rippers and no two documents have the same contents.

10.1.1.2 Used by

connectivity_ripper connectivity_sub_net documentation instance_member_structure_port instance_structure_port name_information property referenced_interconnect

10.1.2 connectivity_net

```
  ENTITY connectivity_net
    SUBTYPE OF (connectivity_generic_net);
  DERIVE
    SELF\connectivity_generic_net.containing_connectivity_view
      : internal_connectivity_view
          := containing_view;
  INVERSE
    containing_view : internal_connectivity_view FOR nets;
  WHERE
    valid_associated_signal :
      (* The signal associated with a connectivity_net is defined in
the
         containing internal_connectivity_view. *)
      associated_signal.containing_view :=: containing_view;

    valid_joined_instance_ports1 :
      (* The same instance_member_structure_port is not referenced
more than
         once within a net.  The instance_member_structure_ports
joined
         implicitly by instance_structure_ports do not overlap with
the set of
         instance_member_structure_ports joined explicitly by the net.
*)
      check_valid_joined_instance_ports(SELF);

    valid_joined_global_ports  :
      (* The global_ports joined by a connectivity_net are a subset of
the
         global_ports joined by its associated signal. *)
      not_exists(QUERY(gp <* NVL(joined_global_ports, []) |
                    NOT (gp IN
                         NVL(associated_signal.joined_global_ports,
[]))));

    valid_instance_member_structure_ports :
```

IECNORM.COM : Click to view the full PDF of IEC 62016:2003
Withdrawn

```
      (* The instance_member_logical_ports associated with the
         instance_member_structure_ports joined by a connectivity_net
         are a subset of the instance_member_logical_ports joined by
its
         associated signal. *)
      not_exists(QUERY(imsp <*
NVL(joined_instance_member_structure_ports, []) |
        not_exists(QUERY(imlp <*

NVL(associated_signal.joined_instance_member_logical_ports, []) |
                   (imlp.referenced_instance :=:
imsp.referenced_instance) AND
                   (imlp.referenced_instance_member_index =
                    imsp.referenced_instance_member_index) AND
                   (imlp.referenced_master_port :=:

imsp.referenced_master_port.associated_logical_port)))));

    valid_joined_instance_structure_ports :
      (* The instance_member_logical_ports associated with the
         instance_member_structure_ports of the
instance_structure_ports joined
         by a connectivity_net are a subset of the
         instance_member_logical_ports joined by its associated
signal. *)
      not_exists(QUERY(ip <* NVL(joined_instance_structure_ports, [])
|
        there_exists(QUERY(imsp <* ip.flattened_port_list |
          not_exists(QUERY(imlp <*
                   NVL
(associated_signal.joined_instance_member_logical_ports, []) |
                   (imlp.referenced_instance :=:
imsp.referenced_instance) AND
                   (imlp.referenced_instance_member_index =
                    imsp.referenced_instance_member_index) AND
                   (imlp.referenced_master_port :=:

imsp.referenced_master_port.associated_logical_port)))))));

    valid_master_structure_ports :
      (* The master_logical_ports associated with the
master_structure_ports
         joined by a connectivity_sub_net are a  subset of the
         master_logical_ports joined by its associated signal. *)
      not_exists(QUERY(msp <* NVL(joined_master_structure_ports, []) |
                   NOT (msp.associated_logical_port IN

NVL(associated_signal.joined_master_logical_ports,[]))));

    valid_joined_connectivity_rippers :
      (* The rippers joined by a connectivity_net are defined in the
         containing internal_connectivity_view. *)
      NVL (joined_connectivity_rippers, []) <=
        NVL (containing_view.rippers, []);
  END_ENTITY;
```

10.1.2.1 Description

A connectivity_net is a structured representation of connectivity in a connectivity view. It is only allowed in an internal_connectivity_view. Its associated signal is defined in its containing view. The global_ports, and the logical ports associated with the instance_member_ structure_ports and master_structure_ports joined by a connectivity_net are a subset of the

IECNORM.COM: Click to view the full PDF of IEC 62016:2003
Withdrawn

ones joined by its associated signal. Also, all joined connectivity_rippers are defined in the containing view. A connectivity_net can only join instance_member_structure_port whose corresponding instance_member_logical_port are explicitly joined by the signal associated with the connectivity_net itself.

10.1.2.2 Used by

internal_connectivity_view

10.1.3 check_valid_joined_instance_ports

```
  FUNCTION check_valid_joined_instance_ports
    (cn : connectivity_net) : BOOLEAN;
    LOCAL
      imp : SET OF instance_member_structure_port := [];
    END_LOCAL;
    REPEAT i := 1 TO SIZEOF(NVL (cn.joined_instance_structure_ports,
[]));
      REPEAT j:=1 TO
SIZEOF(cn.joined_instance_structure_ports[i].flattened_port_list);
        IF
cn.joined_instance_structure_ports[i].flattened_port_list[j] IN imp
THEN
          RETURN(FALSE);
        END_IF;
      END_REPEAT;
      imp := imp +
cn.joined_instance_structure_ports[i].flattened_port_list;
    END_REPEAT;
    REPEAT i := 1 TO SIZEOF(imp);
      IF there_exists(QUERY(imlp <*

cn.associated_signal.joined_instance_member_logical_ports |
                          (imlp.referenced_instance :=:
                           imp[i].referenced_instance) AND
                          (imlp.referenced_instance_member_index =

imp[i].referenced_instance_member_index) AND
                          (imlp.referenced_master_port :=:

imp[i].referenced_master_port.associated_logical_port)))
      THEN
        RETURN(FALSE);
      END_IF;
    END_REPEAT;
    RETURN(TRUE);
  END_FUNCTION;
```

Withdrawn
IECNORM.COM : Click to view the full PDF of IEC 62016:2003

10.1.3.1 Description

check_valid_joined_instance_ports checks that the same instance_member_structure_port is not referenced more than once in a connectivity_net. It also checks that the instance_member_structure_ports joined implicitly by the net do not overlap with the set of instance_member_structure_ports joined explicitly by the net.

10.1.3.2 Used by

connectivity_net

10.1.4 connectivity_sub_net

```
  ENTITY connectivity_sub_net
    SUBTYPE OF (connectivity_generic_net);
  DERIVE
    SELF\connectivity_generic_net.containing_connectivity_view
      : internal_connectivity_view
          := containing_generic_net.containing_connectivity_view;
  INVERSE
    containing_generic_net : connectivity_generic_net FOR sub_nets;
  WHERE
    valid_associated_signal :
      (* The signal associated with a connectivity_sub_net is the same
as its
         immediately containing connectivity_generic_net. *)
      associated_signal :=: containing_generic_net.associated_signal;

    valid_joined_global_ports  :
      (* The global_ports joined by a connectivity_sub_net are a
         subset of the global_ports joined by its containing
         connectivity_generic_net. *)
      NVL (joined_global_ports, []) <=
        NVL (containing_generic_net.joined_global_ports, []);

    valid_instance_member_structure_ports :
      (* The instance_member_structure_ports joined by a
connectivity_sub_net
         are a subset of the instance_member_structure_ports joined by
its
         containing connectivity_generic_net. *)
      NVL (joined_instance_member_structure_ports, []) <=
        NVL
(containing_generic_net.joined_instance_member_structure_ports, []);

    valid_instance_structure_ports :
      (* The instance ports joined by a connectivity_sub_net are a
         subset of the instance ports joined by its containing
         connectivity_generic_net. *)
      NVL (joined_instance_structure_ports, []) <=
        NVL (containing_generic_net.joined_instance_structure_ports,
[]);

    valid_master_structure_ports :
      (* The master_structure_ports joined by a connectivity_sub_net
are a
         subset of the master_structure_ports joined by its containing
         connectivity_generic_net. *)
      NVL (joined_master_structure_ports, []) <=
        NVL (containing_generic_net.joined_master_structure_ports,
[]);

    valid_connectivity_rippers :
      (* The rippers joined by a connectivity_sub_net are a
         subset of the ripper ports joined by its containing
         connectivity_generic_net. *)
      NVL (joined_connectivity_rippers, []) <=
        NVL (containing_generic_net.joined_connectivity_rippers, []);
  END_ENTITY;
```

IECNORM.COM : Click to view the full PDF of IEC 62016:2003
Withdrawn

10.1.4.1 Description

A connectivity_sub_net joins to ports and rippers which are joined by its immediately containing connectivity_generic_net. In addition, its associated signal is the same as that of its immediately containing connectivity_generic_net. The global_ports instance_member_ structure_ports, instance_ports and master_structure_ports are all joined by a connectivity_sub_net and are a subset of the the global_ports, instance_member_ structure_ports, instance_ports and master_structure_ports, respectively. They are all joined by their containing connectivity_generic_net. The rippers joined by a connectivity_sub_net are a subset of the ripper_ports joined by its containing connectivity_generic_net.

10.1.4.2 Used by

connectivity_generic_net

10.1.5 connectivity_generic_bus

```
  ENTITY connectivity_generic_bus
    ABSTRACT SUPERTYPE OF (ONEOF(connectivity_bus,
                                 connectivity_bus_slice,
                                 connectivity_sub_bus));
    associated_signal_bundle                 : signal_bundle;
-- reference
    bus_slices                               : OPTIONAL SET [1:?] OF

       connectivity_bus_slice;
    criticality                              : integer_token;
    document                                 : OPTIONAL SET [1:?] OF
                                                    documentation;
    joined_global_port_bundles               : OPTIONAL SET [1:?] OF
-- reference

       global_port_bundle;
    joined_instance_structure_port_bundles      : OPTIONAL SET [1:?]
OF          -- reference

       instance_structure_port_bundle;
    joined_local_master_port_bundles         : OPTIONAL SET [1:?] OF
-- reference

       local_master_port_bundle;
    joined_master_structure_port_bundles             : OPTIONAL SET
[1:?] OF         -- reference

       master_structure_port_bundle;
    joined_connectivity_rippers                      : OPTIONAL SET
[1:?] OF         -- reference

       connectivity_ripper;
    name                                     : OPTIONAL
name_information;
    properties                               : OPTIONAL SET [1:?] OF
                                                    property;
    sub_busses                               : OPTIONAL SET [1:?] OF

       connectivity_sub_bus;
    containing_connectivity_view             :
internal_connectivity_view;
                                                        -- derived in
subtypes
  DERIVE
    commoned_port_bundles : SET OF instance_structure_port_bundle
```

Withdrawn
IECNORM.COM: Click to view the full PDF of IEC 62016:2003

```
      := QUERY(instanceStructurePortBundle <*
joined_instance_structure_port_bundles |

instanceStructurePortBundle.referenced_master_structure_port_bundle.si
ze =
               signal_bundle_size);
    fanned_out_port_bundles : SET OF instance_structure_port_bundle
      := QUERY(instanceStructurePortBundle <*
joined_instance_structure_port_bundles |
               instanceStructurePortBundle.size = signal_bundle_size);
    signal_bundle_size : INTEGER
      := SIZEOF(associated_signal_bundle.flattened_signal_list);
  WHERE
    valid_joined_global_port_bundle_size :
      (* All joined global_port_bundles have the same port size as
         the size of the flattened signal_bundle. *)
      not_exists(QUERY(gpb <* joined_global_port_bundles |
             gpb.size <> signal_bundle_size));

    valid_joined_local_master_port_bundle_size :
      (* All joined local_master_port_bundles have the same port size
as
         the size of the flattened signal_bundle. *)
      not_exists(QUERY(joinedLocalMasterPortBundle <*
                        joined_local_master_port_bundles |
             joinedLocalMasterPortBundle.size <> signal_bundle_size));

    valid_joined_master_structure_port_bundle_size :
      (* All joined master_structure_port_bundles have the same port
size as
         the size of the flattened signal_bundle. *)
      not_exists(QUERY(joinedMasterStructurePortBundle <*
           joined_master_structure_port_bundles |
             joinedMasterStructurePortBundle.size <>
signal_bundle_size));

    valid_joined_instance_structure_port_bundles :
      (* All instance_structure_port_bundles are either joined in the
         commoned or fanned out style. *)
      joined_instance_structure_port_bundles =
        commoned_port_bundles + fanned_out_port_bundles;

    valid_structural_connectivity :
      (* The structural connectivity of a connectivity_generic_bus is
         consistent with the logical connectivity of the associated
         signal_bundle. *)
      check_structural_connectivity
       (associated_signal_bundle.flattened_signal_list,
        commoned_port_bundles,
        fanned_out_port_bundles,
        NVL (joined_global_port_bundles, []),
        NVL (joined_local_master_port_bundles, []),
        NVL (joined_master_structure_port_bundles, []));

    valid_bus_slices :
      (* No two bus_slices within a connectivity_generic_bus refer to
the
         same signal_bundle. *)
      not_exists(QUERY(busSlice1 <* bus_slices |
        there_exists(QUERY(busSlice2 <* bus_slices - busSlice1 |
          busSlice1.associated_signal_bundle :=:
          busSlice2.associated_signal_bundle))));
```

IECNORM.COM : Click to view the full PDF of IEC 62016:2003
Withdrawn

```
    valid_sub_busses :
      (* No two sub_busses within a connectivity_generic_bus refer to
the
         same set of ports and rippers. *)
    not_exists(QUERY(bus1 <* sub_busses |
      there_exists(QUERY(bus2 <* sub_busses – bus1 |
        (NVL (bus1.joined_global_port_bundles, []) =
         NVL (bus2.joined_global_port_bundles, []))
    AND (NVL (bus1.joined_instance_structure_port_bundles, []) =
         NVL (bus2.joined_instance_structure_port_bundles, []))
    AND (NVL (bus1.joined_local_master_port_bundles, []) =
         NVL (bus2.joined_local_master_port_bundles, []))
    AND (NVL (bus1.joined_master_structure_port_bundles, []) =
         NVL (bus2.joined_master_structure_port_bundles, []))
    AND (NVL (bus1.joined_connectivity_rippers, []) =
         NVL (bus2.joined_connectivity_rippers, []))))));

    unique_document :
      (* No two documents have the same contents. *)
      value_unique(document);
  END_ENTITY;
```

IECNORM.COM : Click to view the full PDF of IEC 62016:2003

10.1.5.1 Description

A connectivity_generic_bus is either a connectivity_bus, a connectivity_bus_slice or a connectivity_sub_bus. A connectivity_generic_bus is used in the model because of the similarity between a connectivity_bus, a connectivity_bus_slice and a connectivity_sub_bus. A connectivity_bus may be subdivided in two orthogonal ways. It may be divided into connectivity_bus_slices or into connectivity_sub_buses. No two connectivity_bus_slices at a given level are associated with the same signal_bundle. No two connectivity_sub_buses at a given level join to the same set of ports and rippers. All joined global_port_bundles, local_master_port_bundles, master_structure_port_bundles and instance_structure_port_bundles are the same size as the flattened signal_bundle. Each single port of the joined port_bundles is mentioned in the corresponding signal within the flattened signal_bundle. The structural connectivity of a connectivity_generic_bus is consistent with the logical connectivity of the associated signal_bundle.

When a wide bus joins to an instance_structure_port_bundle, the pattern of connectivity is either in the commoned style or in the fanned-out style. For the commoned style, the joined instance_structure_port_bundles reference master_structure_port_bundles whose port size is the same as the size of the flattened signal_bundle. For the fanned-out style, the joined instance_structure_port_bundles are flattened to lists of instance_member_structure_ports. The length of these lists is equal to the size of the flattened signal_bundle.

For example, if a bus structure is associated with a signal_bundle of size 8, it may join to the following ports:

global_port_bundle (port_bundle size = 8) instance_structure_port_bundle in commoned style (port bundle size = 8) instance_structure_port_bundle in fanned out style (port_bundle size = 8) local_master_port_bundle (port_bundle size = 8) master_structure_port_bundle (port_bundle size = 8)

10.1.5.2 Used by

connectivity_bus_slice connectivity_ripper connectivity_sub_bus documentation instance_structure_port_bundle name_information property referenced_interconnect

10.1.6 connectivity_bus

```
  ENTITY connectivity_bus
    SUBTYPE OF (connectivity_generic_bus);
  DERIVE
    SELF\connectivity_generic_bus.containing_connectivity_view
      : internal_connectivity_view                          --
reference
        := containing_view;
  INVERSE
    containing_view : internal_connectivity_view FOR busses;
  WHERE
    valid_associated_signal_bundle :
      (* The signal_bundle associated with a connectivity_bus is
defined in the
         containing internal_connectivity_view. *)
      associated_signal_bundle.containing_view :=: containing_view;

    valid_joined_local_master_port_bundles :
      (* The joined local_master_port_bundles are defined in the
containing
         internal_connectivity_view. *)
      NVL (joined_local_master_port_bundles, []) <=
        NVL (containing_view.local_master_port_bundles, []);

    valid_joined_connectivity_rippers :
      (* Rippers joined by a connectivity_bus are defined in the
         containing internal_connectivity_view. *)
      NVL (joined_connectivity_rippers, []) <=
        NVL (containing_view.rippers, []);
  END_ENTITY;
```

10.1.6.1 Description

A connectivity_bus is defined as a structured representation of the connectivity of a connectivity view. It is in an internal_connectivity_view. Its associated signal_bundle and the joined local_master_port_bundle are defined in the internal_connectivity_view. All joined connectivity_rippers are also defined in the internal_connectivity_view.

10.1.6.2 Used by

internal_connectivity_view

10.1.7 connectivity_bus_slice

```
  ENTITY connectivity_bus_slice
    SUBTYPE OF (connectivity_generic_bus);
  DERIVE
    SELF\connectivity_generic_bus.containing_connectivity_view
      : internal_connectivity_view                        -- reference
        := containing_generic_bus.containing_connectivity_view;
  INVERSE
    containing_generic_bus : connectivity_generic_bus FOR bus_slices;
  WHERE
    valid_associated_signal_bundle :
      (* The signal_bundle associated with a connectivity_bus_slice is
         a member of the signal_bundle associated with its immediately
         containing connectivity_generic_bus.*)
      associated_signal_bundle IN
```

Withdrawn
IECNORM.COM : Click to view the full PDF of IEC 62016:2003

```
      containing_generic_bus.associated_signal_bundle.members;

    valid_joined_local_master_port_bundles :
      (* The joined local_master_port_bundles are defined in the
containing
         internal_connectivity_view. *)
      NVL (joined_local_master_port_bundles, []) <=
        NVL (containing_connectivity_view.local_master_port_bundles,
[]);

    valid_joined_connectivity_rippers :
      (* Rippers joined by a connectivity_bus_slice are defined in the
         containing internal_connectivity_view. *)
      NVL (joined_connectivity_rippers, []) <=
        NVL (containing_connectivity_view.rippers, []);
  END_ENTITY;
```

10.1.7.1 Description

A connectivity_bus_slice defines part of a connectivity_bus. Its associated signal_bundle is a member of the signal_bundle associated with its immediately containing connectivity_generic_bus. All joined connectivity_rippers and local_master_port_bundles are defined in the containing internal_connectivity_view.

10.1.7.2 Used by

connectivity_generic_bus

10.1.8 connectivity_sub_bus

```
  ENTITY connectivity_sub_bus
    SUBTYPE OF (connectivity_generic_bus);
  DERIVE
    SELF\connectivity_generic_bus.containing_connectivity_view
      : internal_connectivity_view                           --
reference
         := containing_generic_bus.containing_connectivity_view;
  INVERSE
    containing_generic_bus : connectivity_generic_bus FOR sub_busses;
  WHERE
    valid_associated_signal_bundle :
      (* The signal_bundle associated with a connectivity_sub_bus is
the same as
         its immediately containing connectivity_generic_bus. *)
      associated_signal_bundle :=:
      containing_generic_bus.associated_signal_bundle;

    valid_joined_global_port_bundles :
      (* The global_port_bundles joined by a connectivity_sub_bus are
         a subset of the global_port_bundles joined by its
         immediately containing connectivity_generic_bus. *)
      NVL (joined_global_port_bundles, []) <=
        NVL (containing_generic_bus.joined_global_port_bundles, []);

    valid_joined_instance_structure_port_bundles :
      (* The instance_structure_port_bundles joined by a
connectivity_sub_bus
         are a subset of the instance_structure_port_bundles joined by
its
         immediately containing connectivity_generic_bus. *)
```

Withdrawn
IECNORM.COM: Click to view the full PDF of IEC 62016:2003

```
      NVL (joined_instance_structure_port_bundles, []) <=
        NVL
(containing_generic_bus.joined_instance_structure_port_bundles, []);

    valid_joined_local_master_port_bundles :
      (* The local_master_port_bundles joined by a
connectivity_sub_bus
          are a subset of the instance_structure_port_bundles joined
by its
          immediately containing connectivity_generic_bus. *)
      NVL (joined_local_master_port_bundles, []) <=
        NVL (containing_generic_bus.joined_local_master_port_bundles,
[]);

    valid_joined_master_structure_port_bundles :
      (* The master_structure_port_bundles joined by a
connectivity_sub_bus
         are a subset of the master_structure_port_bundles joined by
its
         immediately containing connectivity_generic_bus. *)
      NVL (joined_master_structure_port_bundles, []) <=
        NVL
(containing_generic_bus.joined_master_structure_port_bundles, []);

    valid_joined_connectivity_rippers :
      (* The connectivity_rippers joined by a connectivity_sub_bus are
         a subset of the connectivity_rippers joined by its
immediately
         containing connectivity_generic_bus. *)
      NVL (joined_connectivity_rippers, []) <=
        NVL (containing_generic_bus.joined_connectivity_rippers, []);
  END_ENTITY;
```

IECNORM.COM : Click to view the full PDF of IEC 62016:2003
Withdrawn

10.1.8.1 Description

A connectivity_sub_bus has the same size as the connectivity_generic_bus immediately containing it because its associated signal_bundle is the same as that of its immediately containing connectivity_generic_bus. The joint global_port_bundles, instance_structure_port_bundles local_master_port_bundles and master_structure_port_bundles are all joined by a connectivity_sub_bus and are a subset of their respective ports which are joined by their immediately containing connectivity_generic_bus. A connectivity_sub_bus only joins to rippers which are joined by its immediately containing connectivity_generic_bus.

10.1.8.2 Used by

connectivity_generic_bus

10.1.9 check_structural_connectivity

```
  FUNCTION check_structural_connectivity
    (signalList              : LIST OF signal;
     commonedPortBundles     : SET OF instance_structure_port_bundle;
     fannedOutPortBundles    : SET OF instance_structure_port_bundle;
     globalPortBundles       : SET OF global_port_bundle;
     localMasterPortBundles : SET OF local_master_port_bundle;
     masterPortBundles        : SET OF master_structure_port_bundle) :
LOGICAL;
    LOCAL
      instStructPort         : instance_member_structure_port;
      logicalPort          : master_logical_port;
      currentSignal        : signal;
      instanceIndex, portIndex, signalIndex : INTEGER;
      signalGroupSize      : INTEGER := SIZEOF(signalList);
      validConnectivity    : LOGICAL := TRUE;
      imps                 : SET OF instance_member_logical_port;
      gps                  : SET OF global_port;
      mps                  : SET OF master_logical_port;
    END_LOCAL;

    REPEAT signalIndex := 1 TO SIZEOF(signalList);
      currentSignal := signalList[signalIndex];
      imps := NVL (currentSignal.joined_instance_member_logical_ports,
[]);
      gps := NVL (currentSignal.joined_global_ports, []);
      mps := NVL (currentSignal.joined_master_logical_ports, []);

      REPEAT portIndex := 1 TO SIZEOF(commonedPortBundles);
        REPEAT instanceIndex := 0 TO
            commonedPortBundles[portIndex].referenced_instance.width –
1;
          logicalPort
            := commonedPortBundles[portIndex].
               referenced_master_structure_port_bundle.
               associated_logical_port_bundle.

flattened_port_list[signalIndex+signalGroupSize*instanceIndex];

          IF not_exists(QUERY(lp <* imps |
            (commonedPortBundles[portIndex].referenced_instance :=:
             lp.referenced_instance) AND
            (instanceIndex =
             lp.referenced_instance_member_index) AND
            (logicalPort :=:
             lp.referenced_master_port)))
          THEN
            validConnectivity := FALSE;
          END_IF;
        END_REPEAT;
      END_REPEAT; -- check all instance_structure_port_bundles joined
                  -- in commoned style

      REPEAT portIndex := 1 TO SIZEOF(fannedOutPortBundles);
        logicalPort := fannedOutPortBundles[portIndex].
                       referenced_master_structure_port_bundle.
                       associated_logical_port_bundle.
                       flattened_port_list[signalIndex];
        REPEAT instanceIndex := 1 TO fannedOutPortBundles[portIndex].
                                     referenced_instance.
                                     width;
          IF not_exists(QUERY(lp <* imps |
               (fannedOutPortBundles[portIndex].referenced_instance
:=:
                lp.referenced_instance) AND
               (instanceIndex =
                lp.referenced_instance_member_index) AND
               (logicalPort :=:
                lp.referenced_master_port)))
          THEN
            validConnectivity := FALSE;
          END_IF;
        END_REPEAT;
      END_REPEAT; -- check all instance_structure_port_bundles joined
```

Withdrawn
IECNORM.COM : Click to view the full PDF of IEC 62016:2003

```
                  -- in fanned out style

      REPEAT portIndex := 1 TO SIZEOF(globalPortBundles);
        IF NOT (globalPortBundles[portIndex].
                flattened_port_list[signalIndex]
                IN gps) THEN
           validConnectivity := FALSE;
        END_IF;
      END_REPEAT; -- check all global_port_bundles

      REPEAT portIndex := 1 TO SIZEOF(localMasterPortBundles);
        IF NOT (localMasterPortbundles[portIndex].
                flattened_port_list[signalIndex]
                IN mps) THEN
           validConnectivity := FALSE;
        END_IF;
      END_REPEAT; -- check all local_master_port_bundles

      REPEAT portIndex := 1 TO SIZEOF(masterPortBundles);
        IF NOT (masterPortBundles[portIndex].
                associated_logical_port_bundle.
                flattened_port_list[signalIndex]
                IN mps) THEN
           validConnectivity := FALSE;
        END_IF;
      END_REPEAT; -- check all master_structure_port_bundles

    END_REPEAT;  -- check all signals
  END_FUNCTION;
```

10.1.9.1 Description

The check_structural_connectivity function returns true if the structural connectivity of a connectivity_bus or a connectivity_bus_slice or a connectivity_sub_bus conforms with its logical connectivity.

10.1.9.2 Used by

connectivity_generic_bus

10.1.10 connectivity_ripper

```
  ENTITY connectivity_ripper;
  INVERSE
    related_nets : SET OF connectivity_generic_net FOR
joined_connectivity_rippers;
    related_busses : SET [1:?] OF connectivity_generic_bus FOR
joined_connectivity_rippers;
    containing_internal_connectivity_view
      : internal_connectivity_view FOR rippers;
  WHERE
    valid_related_nets :
      (* All related nets belong to the same view as the
         connectivity_ripper. *)
      not_exists(QUERY(net <* related_nets |
            net.containing_connectivity_view :<>:
                containing_internal_connectivity_view));

    valid_related_busses :
```

Withdrawn
IECNORM.COM : Click to view the full PDF of IEC 62016:2003

```
      (* All related busses belong to the same view as the
         connectivity_ripper. *)
      not_exists(QUERY(bus <* related_busses |
            bus.containing_connectivity_view :<>:
                 containing_internal_connectivity_view));

    valid_net_structure :
      (* For each related net, there is at least one other related net
         or bus which shares a common signal. *)
      not_exists (QUERY (net <* related_nets |
        not_exists(QUERY(net1 <* related_nets - net |
                          net1.associated_signal :=:
net.associated_signal))
        AND
        not_exists(QUERY(bus1 <* related_busses |
                         net.associated_signal IN

bus1.associated_signal_bundle.flattened_signal_list))));

    valid_bus_structure :
      (* For each related bus, there is at least one other related net
         or bus which shares a common signal. *)
      not_exists(QUERY(bus <* related_busses |
        not_exists(QUERY(bus1 <* related_busses - bus |
                   there_exists(QUERY(sig <*
                                   bus1.associated_signal_bundle.
                                   flattened_signal_list |
                             sig IN

bus.associated_signal_bundle.flattened_signal_list))))
        AND
        not_exists(QUERY(net1 <* related_nets |
                  net1.associated_signal IN

bus.associated_signal_bundle.flattened_signal_list))));
  END_ENTITY;
```

IECNORM.COM : Click to view the full PDF of IEC 62016:2003

10.1.10.1 Description

A connectivity_ripper defines a relationship between net and bus structures in a connectivity view. All related nets and buses belong to the same view as the connectivity_ripper. For each related bus or net, there is at least one other related bus or net which shares a common signal.

10.1.10.2 Used by

connectivity_generic_bus connectivity_generic_net internal_connectivity_view

10.1.11 connectivity_view_model

Description

The connectivity_view_model schema describes what information can be found in a connectivity view. The connectivity view conveys hierarchical and connectivity information. The hierarchy information describes how a circuit is broken down into a number of sub-circuits. The connectivity information defines how circuits are connected.

10.1.12 connectivity_instance_implementation

```
  ENTITY connectivity_instance_implementation;
    name                  : OPTIONAL name_information;
    referenced_instance : instance;            -- reference
    selected_structure  : port_structure;      -- reference
    overriding_structure_properties
                          : OPTIONAL SET [1:?] OF property_override;
  INVERSE
    containing_connectivity_view
      : internal_connectivity_view FOR implementations;
  UNIQUE
    unique_referenced_instance :
      (* A connectivity_instance_implementation can be associated with
only
         one instance *)
      referenced_instance;
  WHERE
    valid_referenced_instance :
      (* the referenced instance is defined in the containing
         internal_connectivity_view *)
      referenced_instance IN containing_connectivity_view.instances;

    valid_structure_properties :
      (* the overridden properties are defined in the
         port_structure *)
      not_exists(QUERY(pr <* overriding_structure_properties |
                  NOT (pr.overridden_property IN
                       selected_structure.properties)));

    valid_selected_structure :
      (* the selected port_structure is defined in the interface of
the
         instantiated cluster of the referenced_instance *)
      selected_structure IN
referenced_instance.instantiated_cluster.interface.port_structures;
  END_ENTITY;
```

IECNORM.COM : Click to view the full PDF of IEC 62016:2003 Withdrawn

10.1.12.1 Description

A connectivity_instance_implementation describes information associated with the implementation of an instance. Since the only type of cell_representation defined in the current model is the connectivity view, the only implementation information is the identification of the port structure to be used. The referenced instance is defined in the containing internal_connectivity_view, the overridden properties are defined in the port_structure and the selected port_structure is defined in the interface of the instantiated cluster of the referenced instance.

10.1.12.2 Used by

internal_connectivity_view name_information property_override

10.1.13 internal_connectivity_view

```
  ENTITY internal_connectivity_view
    SUBTYPE OF (internal_cell_representation);
    busses              : OPTIONAL SET [1:?] OF connectivity_bus;
    instances           : OPTIONAL SET [1:?] OF instance;
    local_master_port_bundles  : OPTIONAL SET [1:?] OF
local_master_port_bundle;
    nets                : OPTIONAL SET [1:?] OF connectivity_net;
    rippers             : OPTIONAL SET [1:?] OF connectivity_ripper;
    signals             : OPTIONAL SET [1:?] OF signal;
    signal_bundles       : OPTIONAL SET [1:?] OF signal_bundle;
    implementations     : OPTIONAL SET [1:?] OF
                          connectivity_instance_implementation;
    units               : connectivity_units;
  INVERSE
    related_expandable_configurations
      : SET [0:?] OF
expandable_internal_connectivity_view_configuration
          FOR chosen_cell_representation;
  WHERE
    valid_direct_instances :
      (* An internal_connectivity_view does not contain instances of
clusters
         of the containing_cell directly. *)
      not_exists(QUERY(inst <* instances |
        inst.instantiated_cluster.containing_cell :=:
        containing_internal_cluster.containing_internal_cell));

    valid_indirect_instances :
      (* An internal_connectivity_view does not contain instances of
clusters
         of the containing_cell indirectly. *)
      not_exists(QUERY(ecc <* related_expandable_configurations |
         there_exists(QUERY(ic <* ecc.instance_configurations |
           ic.configured_instance.instantiated_cluster.containing_cell
:=:
           containing_internal_cluster.containing_internal_cell))));

    valid_signal_global_ports :
      (* Only one signal connects to a particular global_port. *)
      not_exists (QUERY (sig1 <* signals |
        EXISTS (sig1.joined_global_ports)
        AND
        there_exists (QUERY (sig2 <* signals - sig1 |
                      EXISTS (sig2.joined_global_ports)
                      AND
                      there_exists (sig1.joined_global_ports *
                                    sig2.joined_global_ports)))));

    valid_signal_instance_member_ports :
      (* Only one signal connects to a particular
         instance_member_logical_port. *)
      not_exists (QUERY (sig1 <* signals |
        EXISTS (sig1.joined_instance_member_logical_ports)
        AND
        there_exists (QUERY (sig2 <* signals - sig1 |
                      EXISTS
(sig2.joined_instance_member_logical_ports)
                      AND
                      there_exists
(sig1.joined_instance_member_logical_ports *
```

IECNORM.COM : Click to view the full PDF of IEC 62016:2003 Withdrawn

```
sig2.joined_instance_member_logical_ports)))));

    valid_signal_master_ports :
      (* Only one signal connects to a particular master_logical_port.
*)
      not_exists (QUERY (sig1 <* signals |
        EXISTS (sig1.joined_master_logical_ports)
        AND
        there_exists (QUERY (sig2 <* signals - sig1 |
                      EXISTS (sig2.joined_master_logical_ports)
                      AND
                      there_exists (sig1.joined_master_logical_ports *

sig2.joined_master_logical_ports)))));
  END_ENTITY;
```

10.1.13.1 Description

An internal_connectivity_view is a cell representation which describes connectivity. An internal_connectivity_view does not contain instances of the clusters of the containing cell directly. Only one signal connects to a particular global_port or instance_member_ logical_port or a master_logical_port.

10.1.13.2 Used by

connectivity_bus connectivity_bus_slice connectivity_generic_bus connectivity_generic_net connectivity_instance_implementation connectivity_net connectivity_ripper connectivity_sub_ bus connectivity_sub_net connectivity_units expandable_internal_connectivity_ occurrence_ annotate_expandable_internal_connectivity_view_configuration instance local_master_port_ bundle signal signal_bundle

Withdrawn
IECNORM.COM : Click to view the full PDF of IEC 62016:2003

10.1.14 external_connectivity_view

```
  ENTITY external_connectivity_view
    SUBTYPE OF (external_cell_representation);
    units : connectivity_units;
  END_ENTITY;
```

10.1.14.1 Description

An external_connectivity_view is a connectivity view found in a cluster within an external_cell.

10.1.14.2 Used by

connectivity_units

10.1.15 connectivity_units

```
  ENTITY connectivity_units;
    set_capacitance : capacitance_unit;
  INVERSE
    containing_internal_connectivity_view
      : SET [0:1] OF internal_connectivity_view FOR units;
    containing_external_connectivity_view
      : SET [0:1] OF external_connectivity_view FOR units;
  WHERE
    containment_constraint :
      SIZEOF (containing_internal_connectivity_view) +
      SIZEOF (containing_external_connectivity_view) = 1;
  END_ENTITY;
```

10.1.15.1 Description

A connectivity_units sets the scaling for a connectivity view. In a connectivity view, it is appropriate to set a capacitance scale.

10.1.15.2 Used by

capacitance_unit external_connectivity_view internal_connectivity_view

10.2 design_hierarchy_model

Description

The design_hierarchy_model schema describes the annotation data relating to an occurrence hierarchy. A design_hierarchy is defined by choosing the top cluster_configuration. It is annotated by an occurrence_hierarchy_annotate. Interconnects and master ports within the top occurrence can be annotated by interconnect_annotate and master_port_annotate respectively. If further annotation is to be given for occurrences further down in the occurrence hierarchy, occurrence_annotate specify which occurrence is to be annotated. Similarly, interconnects and instance ports within an occurrence can be annotated by interconnect_annotate and instance_port_annotate respectively.

Withdrawn
IECNORM.COM : Click to view the full PDF of IEC 62016:2003

10.2.1 design

```
  ENTITY design;
    design_hierarchies   : OPTIONAL SET [1:?] OF design_hierarchy;
    document             : OPTIONAL SET [1:?] OF documentation;
    name                 : OPTIONAL name_information;
    properties           : OPTIONAL SET [1:?] OF private_property;
    top_cell             : cell;                              --
reference
    units                : design_units;
    status_of_copyright  : OPTIONAL SET [1:?] OF copyright;
    status_of_written    : OPTIONAL SET [1:?] OF written;
  INVERSE
    containing_information_base : information_base FOR designs;
  WHERE
    unique_document :
      (* There are no two documents with the same contents *)
      value_unique(document);

    unique_status_of_copyright :
      (* The design does not have the same copyright information
         twice *)
      value_unique(status_of_copyright);

    unique_status_of_written :
      (* The design does not have two identical "written"
         information *)
      value_unique(status_of_written);
  END_ENTITY;
```

10.2.1.1 Description

A design identifies the cell at the top level of the hierarchy of a particular design within an information_base. It may have multiple design_hierarchy. There are no two documents with the same content.

10.2.1.2 Used by

copyright design_hierarchy design_units documentation information_base name_information private_property written

10.2.2 design_units

```
  ENTITY design_units;
    set_capacitance : capacitance_unit;
  INVERSE
    containing_design : design FOR units;
  END_ENTITY;
```

10.2.2.1 Description

A design_units sets the scaling for a design. It is appropriate to set a capacitance scale.

10.2.2.2 Used by

capacitance_unit design

IECNORM.COM : Click to view the full PDF of IEC 62016:2003
Withdrawn

10.2.3 design_hierarchy

```
  ENTITY design_hierarchy
    ABSTRACT SUPERTYPE OF (ONEOF(external_design_hierarchy,
                                 internal_design_hierarchy));
    chosen_cluster_configuration  : cluster_configuration;      --
reference
    occurrence_hierarchy_annotate : OPTIONAL
occurrence_hierarchy_annotate;
    name                          : OPTIONAL name_information;
  INVERSE
    containing_design : design FOR design_hierarchies;
  WHERE
      valid_design_hierarchies :
      (* All design_hierarchies start with cluster_configurations
which are
         defined in the clusters of the top_cell. *)
 chosen_cluster_configuration.
        containing_cluster.containing_cell :=:
containing_design.top_cell;
  END_ENTITY;
```

10.2.3.1 Description

A design_hierarchy is either an external or an internal design hierarchy. It specifies an expanded occurrence hierarchy by selecting the top cluster_configuration. The chosen cluster_configuration is defined in the cluster_interface of the top_cell in a design.

10.2.3.2 Used by

design name_information occurrence_hierarchy_annotate

10.2.4 external_design_hierarchy

```
  ENTITY external_design_hierarchy
    ABSTRACT SUPERTYPE OF (ONEOF(expandable_external_design_hierarchy,
                                 leaf_external_design_hierarchy))
    SUBTYPE OF (design_hierarchy);
    SELF\design_hierarchy.chosen_cluster_configuration    -- reference
      : external_cluster_configuration;
    SELF\design_hierarchy.occurrence_hierarchy_annotate
      : OPTIONAL external_occurrence_hierarchy_annotate;
  END_ENTITY;
```

10.2.4.1 Description

An external_design_hierarchy is either an expandable_external_design_hierarchy or a leaf_external_design_hierarchy. It selects an external cluster configuration which is defined in a cluster of an external_cell.

10.2.4.2 Used by

external_occurrence_hierarchy_annotate

Withdrawn
IECNORM.COM : Click to view the full PDF of IEC 62016:2003

10.2.5 expandable_external_design_hierarchy

```
ENTITY expandable_external_design_hierarchy
  SUBTYPE OF (external_design_hierarchy);
  SELF\design_hierarchy.chosen_cluster_configuration     -- reference
    : expandable_external_cluster_configuration;
  SELF\design_hierarchy.occurrence_hierarchy_annotate
    : OPTIONAL expandable_external_occurrence_hierarchy_annotate;
END_ENTITY;
```

10.2.5.1 Description

An expandable_external_design_hierarchy selects an expandable_external_cluster_configuration.

10.2.5.2 Used by

expandable_external_occurrence_hierarchy_annotate

10.2.6 leaf_external_design_hierarchy

```
ENTITY leaf_external_design_hierarchy
  SUBTYPE OF (external_design_hierarchy);
  SELF\design_hierarchy.chosen_cluster_configuration     -- reference
    : leaf_external_cluster_configuration;
  SELF\design_hierarchy.occurrence_hierarchy_annotate
    : OPTIONAL leaf_external_occurrence_hierarchy_annotate;
END_ENTITY;
```

10.2.6.1 Description

A leaf_external_design_hierarchy selects a leaf_external_cluster_configuration.

10.2.6.2 Used by

leaf_external_occurrence_hierarchy_annotate

10.2.7 internal_design_hierarchy

```
ENTITY internal_design_hierarchy
  ABSTRACT SUPERTYPE OF (ONEOF(expandable_internal_design_hierarchy,
                               leaf_internal_design_hierarchy))
  SUBTYPE OF (design_hierarchy);
  SELF\design_hierarchy.chosen_cluster_configuration     -- reference
    : internal_cluster_configuration;
  SELF\design_hierarchy.occurrence_hierarchy_annotate
    : OPTIONAL internal_occurrence_hierarchy_annotate;
END_ENTITY;
```

10.2.7.1 Description

An internal_design_hierarchyis an expandable or a leaf internal design hierarchy. It selects an internal_cluster_configuration.

IECNORM.COM : Click to view the full PDF of IEC 62016:2003
Withdrawn

10.2.7.2 Used by

internal_occurrence_hierarchy_annotate

10.2.8 expandable_internal_design_hierarchy

```
  ENTITY expandable_internal_design_hierarchy
    ABSTRACT SUPERTYPE OF
       (ONEOF (expandable_internal_connectivity_design_hierarchy))
    SUBTYPE OF (internal_design_hierarchy);
    SELF\design_hierarchy.chosen_cluster_configuration     --
reference
      : expandable_internal_cluster_configuration;
    SELF\design_hierarchy.occurrence_hierarchy_annotate
      : OPTIONAL expandable_internal_occurrence_hierarchy_annotate;
  END_ENTITY;
```

10.2.8.1 Description

An expandable_internal_design_hierarchy is an "expandable_internal_ _connectivity_design_ hierarchy". It selects a cluster_configuration which provides information for further expansion.

10.2.8.2 Used by

expandable_internal_occurrence_hierarchy_annotate

10.2.9 expandable_internal_connectivity_design_hierarchy

```
  ENTITY expandable_internal_connectivity_design_hierarchy
    SUBTYPE OF (expandable_internal_design_hierarchy);
    SELF\design_hierarchy.chosen_cluster_configuration    -- reference
      : expandable_internal_connectivity_view_configuration;
    SELF\design_hierarchy.occurrence_hierarchy_annotate
      : OPTIONAL expandable_internal_occurrence_hierarchy_annotate;
  END_ENTITY;
```

10.2.9.1 Description

An expandable_internal_connectivity_design_hierarchy selects a cluster_configuration which provides information for the further expansion starting at an internal_connectivity_view.

10.2.9.2 Used by

expandable_internal_connectivity_occurrence_hierarchy_annotate

10.2.10 leaf_internal_design_hierarchy

```
  ENTITY leaf_internal_design_hierarchy
    SUBTYPE OF (internal_design_hierarchy);
    SELF\design_hierarchy.chosen_cluster_configuration   -- reference
      : leaf_internal_cluster_configuration;
    SELF\design_hierarchy.occurrence_hierarchy_annotate
      : OPTIONAL leaf_internal_occurrence_hierarchy_annotate;
  END_ENTITY;
```

Withdrawn IECNORM.COM: Click to view the full PDF of IEC 62016:2003

10.2.10.1 Description

A leaf_internal_design_hierarchy corresponds to a cluster_configuration which cannot be expanded any further.

10.2.10.2 Used by

leaf_internal_occurrence_hierarchy_annotate

10.2.11 occurrence_hierarchy_annotate

```
  ENTITY occurrence_hierarchy_annotate
    ABSTRACT SUPERTYPE OF
(ONEOF(external_occurrence_hierarchy_annotate,

internal_occurrence_hierarchy_annotate));
    master_port_annotates          : OPTIONAL SET [1:?] OF
master_port_annotate;
    overriding_cell_properties     : OPTIONAL SET [1:?] OF
property_override;
    overriding_cluster_properties : OPTIONAL SET [1:?] OF
property_override;
    overriding_designator          : OPTIONAL string_token;
  INVERSE
    containing_design_hierarchy
      : design_hierarchy FOR occurrence_hierarchy_annotate;
  WHERE
    valid_master_port_annotate_definitions :
      (* The annotated master_logical_ports are defined in the
interface of
         the containing cluster of the chosen_configuration of the
containing
         design_hierarchy. *)
        not_exists(QUERY(portAnnotate <* master_port_annotates |
         portAnnotate.annotated_port.containing_interface :<>:
           containing_design_hierarchy.
           chosen_cluster_configuration.
           containing_cluster.
           interface));

    valid_overriding_cell_properties :
      (* The overridden properties are defined in the containing cell
of the
         chosen cluster_configuration of the containing
design_hierarchy. *)
      not_exists(QUERY(cellPropertyOverride <*
overriding_cell_properties |
        NOT (containing_design_hierarchy.chosen_cluster_configuration.
           containing_cluster.containing_cell IN

cellPropertyOverride.overridden_property.containing_cell)));

    valid_overriding_cluster_properties :
      (* The overridden properties are defined in the containing
cluster of the
         chosen cluster_configuration of the containing
design_hierarchy. *)
      not_exists(QUERY(clusterPropertyOverride <*
                      overriding_cluster_properties |
        NOT (containing_design_hierarchy.chosen_cluster_configuration.
           containing_cluster IN
```

Withdrawn
IECNORM.COM : Click to view the full PDF of IEC 62016:2003

```
clusterPropertyOverride.overridden_property.containing_cluster)));
  END_ENTITY;
```

10.2.11.1 Description

An occurrence_hierarchy_annotate is either an external_occurrence_hierarchy_annotate or an internal_occurrence_hierarchy_annotate. It annotates the top level occurrence of a design_hierarchy. The view chosen for annotation is specified by the cluster_configuration in the containing design_hierarchy.The annotated master_logical_ports are defined in the interface of the containing cluster of the chosen configuration of the containing design_hierarchy. The overridden properties are defined in the containing cell of the chosen cluster_configuration of the containing design_hierarchy.

10.2.11.2 Used by

design_hierarchy master_port_annotate property_override

10.2.12 external_occurrence_hierarchy_annotate

```
  ENTITY external_occurrence_hierarchy_annotate
    ABSTRACT SUPERTYPE OF
      (ONEOF(expandable_external_occurrence_hierarchy_annotate,
             leaf_external_occurrence_hierarchy_annotate))
    SUBTYPE OF (occurrence_hierarchy_annotate);
    SELF\occurrence_hierarchy_annotate.containing_design_hierarchy
      : external_design_hierarchy;                          -- retype of
inverse
  END_ENTITY;
```

10.2.12.1 Description

An external_occurrence_hierarchy_annotate is either an expandable or a leaf external occurrence hierarchy annotate. It annotates the top level occurrence of a design_hierarchy.

10.2.12.2 Used by

external_design_hierarchy

10.2.13 expandable_external_occurrence_hierarchy_annotate

```
  ENTITY expandable_external_occurrence_hierarchy_annotate
    SUBTYPE OF (external_occurrence_hierarchy_annotate);
    overriding_view_properties : OPTIONAL SET [1:?] OF
property_override;
    SELF\occurrence_hierarchy_annotate.containing_design_hierarchy
      : expandable_external_design_hierarchy;               -- retype of
inverse
  DERIVE
    chosen_cell_representation : external_cell_representation
-- reference
      := containing_design_hierarchy.
         chosen_cluster_configuration.
         chosen_cell_representation;
  WHERE
    valid_overriding_view_properties :
```

Withdrawn
IECNORM.COM : Click to view the full PDF of IEC 62016:2003

```
      (* The overridden properties are defined in the
chosen_cell_representation of the top
         occurrence. *)
      not_exists(QUERY(viewPropertyOverride <*
overriding_view_properties |
        NOT (chosen_cell_representation IN

viewPropertyOverride.overridden_property.containing_cell_representatio
n)));
  END_ENTITY;
```

10.2.13.1 Description

An expandable_external_occurrence_hierarchy_annotate annotates the top level occurrence of a design hierarchy. The external view chosen for annotation is specified by the external_cluster_configuration in the containing expandable_external_design_hierarchy. The overridden properties are defined in the chosen_view of the top occurrence.

10.2.13.2 Used by

expandable_external_design_hierarchy property_override

10.2.14 leaf_external_occurrence_hierarchy_annotate

```
  ENTITY leaf_external_occurrence_hierarchy_annotate
    SUBTYPE OF (external_occurrence_hierarchy_annotate);
    SELF\occurrence_hierarchy_annotate.containing_design_hierarchy
      : leaf_external_design_hierarchy;                    -- retype of
inverse
  END_ENTITY;
```

10.2.14.1 Description

A leaf_external_occurrence_hierarchy_annotate annotates a design which contains only a single unexpandable cell.

10.2.14.2 Used by

leaf_external_design_hierarchy

10.2.15 internal_occurrence_hierarchy_annotate

```
  ENTITY internal_occurrence_hierarchy_annotate
    ABSTRACT SUPERTYPE OF
      (ONEOF(expandable_internal_occurrence_hierarchy_annotate,
             leaf_internal_occurrence_hierarchy_annotate))
    SUBTYPE OF (occurrence_hierarchy_annotate);
    SELF\occurrence_hierarchy_annotate.containing_design_hierarchy
      : internal_design_hierarchy;                         -- retype of
inverse
  END_ENTITY;
```

Withdrawn
IECNORM.COM : Click to view the full PDF of IEC 62016:2003

10.2.15.1 Description

An internal_occurrence_hierarchy_annotate is an expandable or a leaf_internal_occurrence_hierarchy_annotate.

10.2.15.2 Used by

internal_design_hierarchy

10.2.16 leaf_internal_occurrence_hierarchy_annotate

```
  ENTITY leaf_internal_occurrence_hierarchy_annotate
    SUBTYPE OF (internal_occurrence_hierarchy_annotate);
    SELF\occurrence_hierarchy_annotate.containing_design_hierarchy
      : leaf_internal_design_hierarchy;                   -- retype of
inverse
  END_ENTITY;
```

10.2.16.1 Description

A leaf_internal_occurrence_hierarchy_annotate annotates a design which contains only an unexpanded cell.

10.2.16.2 Used by

leaf_internal_design_hierarchy

10.2.17 expandable_internal_occurrence_hierarchy_annotate

```
  ENTITY expandable_internal_occurrence_hierarchy_annotate
    ABSTRACT SUPERTYPE OF
      (ONEOF
(expandable_internal_connectivity_occurrence_hierarchy_annotate))
    SUBTYPE OF (internal_occurrence_hierarchy_annotate);
    overriding_view_properties : OPTIONAL SET [1:?] OF
property_override;
    SELF\occurrence_hierarchy_annotate.containing_design_hierarchy
      : expandable_internal_design_hierarchy;              -- retype of
inverse
  DERIVE
    chosen_cell_representation : internal_cell_representation
-- reference
      := containing_design_hierarchy.
        chosen_cluster_configuration.
         chosen_cell_representation;
  WHERE
    valid_overriding_view_properties :
      (* The overridden properties are defined in the
cell_representation_set of the top
         occurrence. *)
      not_exists(QUERY(viewPropertyOverride <*
overriding_view_properties |
        NOT (chosen_cell_representation IN

viewPropertyOverride.overridden_property.containing_cell_representation))
);
  END_ENTITY;
```

Withdrawn
IECNORM.COM : Click to view the full PDF of IEC 62016:2003

10.2.17.1 Description

An expandable_internal_occurrence_hierarchy_annotate annotates the top-level occurrence of an expandable_internal_design_hierarchy. The internal_cell_representation chosen for annotation is specified by the cluster_configuration in the containing expandable_internal_design_hierarchy. The overridden properties are defined in the cell_representation_set of the top occurrence.

10.2.17.2 Used by

expandable_internal_connectivity_design_hierarchy expandable_internal_design_hierarchy property_override

10.2.18 expandable_internal_connectivity_occurrence_hierarchy_annotate

```
  ENTITY
expandable_internal_connectivity_occurrence_hierarchy_annotate
    SUBTYPE OF (expandable_internal_occurrence_hierarchy_annotate);
    occurrence_annotates        : OPTIONAL SET [1:?] OF
occurrence_annotate;
    interconnect_annotates      : OPTIONAL SET [1:?] OF
interconnect_annotate;
    signal_annotates            : OPTIONAL SET [1:?] OF
signal_annotate;
    SELF\occurrence_hierarchy_annotate.containing_design_hierarchy
      : expandable_internal_connectivity_design_hierarchy; -- retype
of inverse
  WHERE
    valid_interconnect_annotate_definition :
      (* The annotated interconnects are defined in the
chosen_cell_representation of the
         top occurrence. *)
      not_exists(QUERY(ia <* interconnect_annotates |
         ia.annotated_interconnect.containing_connectivity_view :<>:
            chosen_cell_representation));

    valid_occurrence_annotates :
      (* The next level occurrence_annotates reference
         instance_configurations which are defined in the chosen
         cluster_configuration of the containing design_hierarchy. *)
      not_exists(QUERY(occurrenceAnnotate <* occurrence_annotates |
                 occurrenceAnnotate.chosen_instance_configuration.
                 containing_expandable_internal_cluster_configuration
                 :<>:

containing_design_hierarchy.chosen_cluster_configuration));

    valid_annotated_signals :
      (* The annotated signals are defined in the
chosen_cell_representation of the top
         occurrence. *)
      not_exists(QUERY(sa <* signal_annotates |
                       sa.annotated_signal.containing_view :<>:
                         chosen_cell_representation));
  END_ENTITY;
```

IECNORM.COM : Click to view the full PDF of IEC 62016:2003

10.2.18.1 Description

An expandable_internal_connectivity_occurrence_hierarchy_annotate annotates the top-level occurrence of an expandable_internal_connectivity_design_hierarchy. The internal_ connectivity_view chosen for annotation is specified by the cluster_configuration in the containing expandable_internal_connectivity_design_hierarchy. The annotated interconnects and the annotated signals are defined in the chosen_cell_representation of the top occurrence. The next level occurrence_annotates reference instance_configurations which are defined in the chosen cluster_configuration of the containing design_hierarchy.

10.2.18.2 Used by

interconnect_annotate occurrence_annotate signal_annotate

10.2.19 occurrence_annotate

```
  ENTITY occurrence_annotate
    ABSTRACT SUPERTYPE OF (ONEOF(external_occurrence_annotate,
                                 internal_occurrence_annotate));
    chosen_instance_configuration : instance_configuration;      --
reference
    instance_member_index           : integer_token;
    instance_port_annotates         : OPTIONAL SET [1:?] OF
                                        instance_port_annotate;
    new_instance_properties         : OPTIONAL SET [1:?] OF property;
    overriding_cell_properties      : OPTIONAL SET [1:?] OF
property_override;
    overriding_cluster_properties   : OPTIONAL SET [1:?] OF
property_override;
    overriding_designator           : OPTIONAL string_token;
    overriding_instance_properties  : OPTIONAL SET [1:?] OF
property_override;
  DERIVE
    annotated_instance : instance -- reference
      := chosen_instance_configuration.configured_instance;
  INVERSE
    containing_expandable_internal_occurrence_hierarchy_annotate
      : SET [0:1] OF
        expandable_internal_connectivity_occurrence_hierarchy_annotate
        FOR occurrence_annotates;
    containing_expandable_internal_connectivity_occurrence_annotate
      : SET [0:1] OF
expandable_internal_connectivity_occurrence_annotate
        FOR occurrence_annotates;
  UNIQUE
    unique_instance :
      (* No two "occurrence_annotates" in the same context annotate
         the same "instance". *)
    containing_expandable_internal_occurrence_hierarchy_annotate,
    containing_expandable_internal_connectivity_occurrence_annotate,
    annotated_instance,
    instance_member_index;
  WHERE
    containment_constraint :
      (* An occurrence_annotate belongs to either one
         occurrence_hierarchy_annotate or one occurrence_annotate . *)

SIZEOF(containing_expandable_internal_occurrence_hierarchy_annotate) +

SIZEOF(containing_expandable_internal_connectivity_occurrence_annotate
) = 1;
```

Withdrawn
IECNORM.COM : Click to view the full PDF of IEC 62016:2003

```
    valid_instance_member_index :
      (* The member index is within the range defined by the width of the
         annotated instance. *)
      {0 <= instance_member_index < annotated_instance.width};

    valid_annotated_instance_ports :
      (* The annotated instance_structure_ports reference the
         annotated_instance. *)
      not_exists(QUERY(portAnnotate1 <* instance_port_annotates |
        there_exists(QUERY(portAnnotate2 <*
                               instance_port_annotates - portAnnotate1 |
          portAnnotate1.annotated_port :=:
portAnnotate2.annotated_port))));

    valid_overriding_cell_properties :
      (* The overridden properties are defined in the containing cell
of the
         instantiated cluster of the annotated_instance. *)
      not_exists(QUERY(cellPropertyOverride <*
                    overriding_cell_properties |
        NOT (annotated_instance.instantiated_cluster.containing_cell
IN

cellPropertyOverride.overridden_property.containing_cell)));

    valid_overriding_cluster_properties :
      (* The overridden properties are defined in the instantiated
cluster of
         the annotated_instance. *)
      not_exists(QUERY(clusterPropertyOverride <*
                    overriding_cluster_properties |
        NOT (annotated_instance.instantiated_cluster IN

clusterPropertyOverride.overridden_property.containing_cluster)));

    valid_overriding_instance_properties :
      (* The overridden properties are defined in the
annotated_instance. *)
      not_exists(QUERY(instancePropertyOverride <*
                    overriding_instance_properties |
        NOT (annotated_instance IN

instancePropertyOverride.overridden_property.containing_instance)));
  END_ENTITY;
```

IECNORM.COM : Click to view the full PDF of IEC 62016:2003
Withdrawn

10.2.19.1 Description

An occurrence_annotate is either an external_occurrence_annotate or an internal_ occurrence_annotate. It annotates an occurrence of any level (apart from the top level and the leaf level occurrences) in an occurrence hierarchy. It has a reference to an instance_configuration which selects the annotated instance and the chosen view for the annotated instance. The member index is within the range defined by the width of the annotated instance. The annotated instance_structure_ports reference the annotated instance.The overridden properties for the cell, cluster and the instance are defined in the containing cell of the instantiated cluster of the annotated instance, the instantiated cluster of the annotated instance, and the annotated instance, respectively.

10.2.19.2 Used by

expandable_internal_connectivity_occurrence_annotate expandable_internal_connectivity_ occurrence_hierarchy_annotate instance_port_annotate property property_override

10.2.20 external_occurrence_annotate

```
  ENTITY external_occurrence_annotate
    ABSTRACT SUPERTYPE OF
(ONEOF(expandable_external_occurrence_annotate,
                                 leaf_external_occurrence_annotate))
    SUBTYPE OF (occurrence_annotate);
    SELF\occurrence_annotate.chosen_instance_configuration
      : external_instance_configuration;                           --
reference
  END_ENTITY;
```

10.2.20.1 Description

An external_occurrence_annotate is either an expandable_external_occurrence_annotate or an leaf_external_occurrence_annotate. It annotates an occurrence of an external_cluster_ instance within a design_hierarchy.

10.2.21 expandable_external_occurrence_annotate

```
  ENTITY expandable_external_occurrence_annotate
    SUBTYPE OF (external_occurrence_annotate);
    overriding_view_properties : OPTIONAL SET [1:?] OF
property_override;
    SELF\occurrence_annotate.chosen_instance_configuration
                 : expandable_external_instance_configuration;     --
reference
  DERIVE
    chosen_cell_representation : external_cell_representation      --
reference
      := chosen_instance_configuration.
         chosen_cluster_configuration.
         chosen_cell_representation;
  WHERE
    valid_overriding_view_properties :
      (* The overridden properties are defined in the
chosen_cell_representation for the
         annotated_instance. *)
      not_exists(QUERY(viewPropertyOverride <*
                       overriding_view_properties |
        NOT (chosen_cell_representation IN

viewPropertyOverride.overridden_property.containing_cell_representatio
n)));
  END_ENTITY;
```

10.2.21.1 Description

An expandable_external_occurrence_annotate annotates an expandable occurrence of an external_cluster_instance within a design_hierarchy. The overridden properties are defined in the chosen_cell_representation for the annotated instance.

10.2.21.2 Used by

property_override

Withdrawn
IECNORM.COM : Click to view the full PDF of IEC 62016:2003

10.2.22 leaf_external_occurrence_annotate

```
  ENTITY leaf_external_occurrence_annotate
    SUBTYPE OF (external_occurrence_annotate);
    SELF\occurrence_annotate.chosen_instance_configuration
      : leaf_external_instance_configuration;                     --
reference
   END_ENTITY;
```

10.2.22.1 Description

A leaf_external_occurrence_annotate annotates a leaf occurrence of an external_cluster_ instance within a design_hierarchy.

10.2.23 internal_occurrence_annotate

```
  ENTITY internal_occurrence_annotate
    ABSTRACT SUPERTYPE OF
(ONEOF(expandable_internal_occurrence_annotate,
                                  leaf_internal_occurrence_annotate))
    SUBTYPE OF (occurrence_annotate);
    SELF\occurrence_annotate.chosen_instance_configuration
      : internal_instance_configuration;                          --
reference
  END_ENTITY;
```

10.2.23.1 Description

An internal_occurrence_annotate is an expandable or a leaf_internal_occurrence_annotate. In both cases it annotates an occurrence of an internal_cluster_instance within a design_ hierarchy.

10.2.24 leaf_internal_occurrence_annotate

```
  ENTITY leaf_internal_occurrence_annotate
    SUBTYPE OF (internal_occurrence_annotate);
    SELF\occurrence_annotate.chosen_instance_configuration
      : leaf_internal_instance_configuration;                    --
reference
   END_ENTITY;
```

Withdrawn
IECNORM.COM : Click to view the full PDF of IEC 62016:2003

10.2.24.1 Description

A leaf_internal_occurrence_annotate annotates a leaf occurrence of an internal_ cluster_instance within a design_hierarchy.

10.2.25 expandable_internal_occurrence_annotate

```
  ENTITY expandable_internal_occurrence_annotate
    ABSTRACT SUPERTYPE OF
      (ONEOF (expandable_internal_connectivity_occurrence_annotate))
    SUBTYPE OF (internal_occurrence_annotate);
    SELF\occurrence_annotate.chosen_instance_configuration
      : expandable_internal_instance_configuration;              --
reference
    overriding_view_properties : OPTIONAL SET [1:?] OF
property_override;
  DERIVE
```

```
    chosen_cell_representation : internal_cell_representation
-- reference
      := chosen_instance_configuration.
         chosen_cluster_configuration.
         chosen_cell_representation;
  WHERE
    valid_overriding_view_properties :
      (* The overridden properties are defined in the
chosen_cell_representation for the
         annotated_instance. *)
      not_exists(QUERY(viewPropertyOverride <*
                       overriding_view_properties |
        NOT (chosen_cell_representation IN

viewPropertyOverride.overridden_property.containing_cell_representatio
n)));
  END_ENTITY;
```

10.2.25.1 Description

An expandable_internal_occurrence_annotate is a expandable_internal_connectivity_occurrence_annotate. It annotates an expandable occurrence of an internal_cluster_instance within a design_hierarchy. The overridden properties are defined in the chosen_cell_representation for the annotated instance.

10.2.25.2 Used by

property_override

10.2.26 expandable_internal_connectivity_occurrence_annotate

```
  ENTITY expandable_internal_connectivity_occurrence_annotate
    SUBTYPE OF (expandable_internal_occurrence_annotate);
    SELF\occurrence_annotate.chosen_instance_configuration
      : expandable_internal_connectivity_instance_configuration;  --
reference
    occurrence_annotates         : OPTIONAL SET [1:?] OF
occurrence_annotate;
    interconnect_annotates       : OPTIONAL SET [1:?] OF
interconnect_annotate;
    signal_annotates             : OPTIONAL SET [1:?] OF
signal_annotate;
  DERIVE
    chosen_connectivity_view : internal_connectivity_view         --
reference
      := chosen_instance_configuration.
         chosen_cluster_configuration.
         chosen_cell_representation;
  WHERE
    valid_interconnect_annotate_definition :
      (* The annotated interconnects are defined in the
chosen_connectivity_view
        of the top occurrence. *)
      not_exists(QUERY(ia <* interconnect_annotates |
         ia.annotated_interconnect.containing_connectivity_view :<>:
               chosen_connectivity_view));

    valid_occurrence_annotates :
      (* The next level occurrence_annotates reference
         instance_configurations defined in the cluster_configuration
         which are chosen by the instance_configuration. *)
```

IECNORM.COM : Click to view the full PDF of IEC 62016:2003
Withdrawn

```
      not_exists(QUERY(occurrenceAnnotate <* occurrence_annotates |
        occurrenceAnnotate.chosen_instance_configuration.
        containing_expandable_internal_cluster_configuration :<>:
        chosen_instance_configuration.chosen_cluster_configuration));

    valid_annotated_signals :
      (* The annotated signals are defined in the
chosen_connectivity_view for the
         annotated_instance. *)
      not_exists(QUERY(sa <* signal_annotates |
         sa.annotated_signal.containing_view :<>:
chosen_connectivity_view));
  END_ENTITY;
```

10.2.26.1 Description

An expandable_internal_connectivity_occurrence_annotate annotates an occurrence of an instance of an internal_connectivity_view within a design_hierarchy. Note that it is expandable. The annotated interconnects and and the annotated signals are defined in the chosen connectivity view of the top occurrence. The next level occurrence_annotates reference instance_configurations defined in the cluster_configuration which are chosen by the instance_configuration.

10.2.26.2 Used by

interconnect_annotate occurrence_annotate signal_annotate

10.2.27 master_port_annotate

```
  ENTITY master_port_annotate
    ABSTRACT SUPERTYPE OF (ONEOF(input_master_port_annotate,
                             output_master_port_annotate,
                             bidirectional_master_port_annotate,

unspecified_direction_master_port_annotate));
    annotated_port          : master_logical_port;          --
reference
    new_properties          : OPTIONAL SET [1:?] OF property;
    overriding_ac_load      : OPTIONAL capacitance_value;
    overriding_designator   : OPTIONAL string_token;
    overriding_properties   : OPTIONAL SET [1:?] OF property_override;
  INVERSE
    containing_occurrence_hierarchy_annotate
      : occurrence_hierarchy_annotate FOR master_port_annotates;
  UNIQUE
    unique_master_port_in_annotate :
      (* No two "master_port_annotates" in a given
         "occurrence_hierarchy_annotate" annotate the same
         "master_logical_port". *)
      containing_occurrence_hierarchy_annotate, annotated_port;
  WHERE
    valid_overriding_properties :
      (* The overridden properties are defined in the annotated
         master_logical_port. *)
      not_exists(QUERY(portPropertyOverride <*
                       overriding_properties |
        NOT (annotated_port IN
             portPropertyOverride.
             overridden_property.
             containing_master_logical_port)));
  END_ENTITY;
```

IECNORM.COM : Click to view the full PDF of IEC 62016:2003
Withdrawn

10.2.27.1 Description

A master_port_annotate is either an input_master_port_annotate, output_master_port_ annotate, bidirectional_master_port_annotate, or an unspecified_direction_master_ port_annotate. A master_port_annotate within an occurrence_hierarchy_annotate attaches or modify properties or attributes which are associated with a master_logical_port occurrence in the top level occurrence. The overridden properties are defined in the annotated master_logical_port.

10.2.27.2 Used by

capacitance_value occurrence_hierarchy_annotate property property_override

10.2.28 input_master_port_annotate

```
  ENTITY input_master_port_annotate
    SUBTYPE OF (master_port_annotate);
    SELF\master_port_annotate.annotated_port                    --
reference
      : input_master_logical_port;
    overriding_dc_fanout_load : OPTIONAL load_value;
    overriding_dc_max_fanin   : OPTIONAL load_value;
  END_ENTITY;
```

10.2.28.1 Description

An input_master_port_annotate attaches or modifies properties and attributes which are associated with an input_master_logical_port occurrence in the top level occurrence.

10.2.29 output_master_port_annotate

```
  ENTITY output_master_port_annotate
    SUBTYPE OF (master_port_annotate);
    SELF\master_port_annotate.annotated_port                  --
reference
      : output_master_logical_port;
    overriding_dc_fanin_load : OPTIONAL load_value;
    overriding_dc_max_fanout : OPTIONAL load_value;
  END_ENTITY;
```

10.2.29.1 Description

An output_master_port_annotate attaches or modifies properties or attributes which are associated with an output_master_logical_port occurrence in the top-level occurrence.

Withdrawn
IECNORM.COM: Click to view the full PDF of IEC 62016:2003

10.2.30 bidirectional_master_port_annotate

```
  ENTITY bidirectional_master_port_annotate
    SUBTYPE OF (master_port_annotate);
    SELF\master_port_annotate.annotated_port                       --
reference
      : bidirectional_master_logical_port;
    overriding_dc_fanin_load  : OPTIONAL load_value;
    overriding_dc_fanout_load : OPTIONAL load_value;
    overriding_dc_max_fanin   : OPTIONAL load_value;
    overriding_dc_max_fanout  : OPTIONAL load_value;
  END_ENTITY;
```

10.2.30.1 Description

The bidirectional_master_port_annotate attaches or modifies properties or attributes which are associated with a bidirectional_master_logical_port occurrence in the top-level occurrence.

10.2.31 unspecified_direction_master_port_annotate

```
  ENTITY unspecified_direction_master_port_annotate
    SUBTYPE OF (master_port_annotate);
    SELF\master_port_annotate.annotated_port                      --
reference
      : unspecified_direction_master_logical_port;
  END_ENTITY;
```

10.2.31.1 Description

An unspecified_direction_master_port_annotate attaches or modifies properties or attributes which are associated with an unspecified_direction_master_logical_port occurrence in the top-level occurrence.

10.2.32 instance_port_annotate

```
  ENTITY instance_port_annotate
    ABSTRACT SUPERTYPE OF (ONEOF(input_instance_port_annotate,
                                 output_instance_port_annotate,
                                 bidirectional_instance_port_annotate,

unspecified_direction_instance_port_annotate));
    annotated_port          : instance_structure_port;              --
reference
    new_properties          : OPTIONAL SET [1:?] OF property;
    overriding_ac_load      : OPTIONAL capacitance_value;
    overriding_designator   : OPTIONAL string_token;
    overriding_properties   : OPTIONAL SET [1:?] OF property_override;
  DERIVE
    port_attributes : instance_port_attributes
      := find_instance_port_attributes(annotated_port.

referenced_master_structure_port.
                                        associated_logical_port,
                                        annotated_port.
                                        referenced_instance);
  INVERSE
    containing_occurrence_annotate
      : occurrence_annotate FOR instance_port_annotates;
  UNIQUE
```

Withdrawn
IECNORM.COM : Click to view the full PDF of IEC 62016:2003

```
    unique_instance_port :
      (* No two instance_port_annotates in a given occurrence_annotate
         annotate the same instance_structure_port. *)
      containing_occurrence_annotate,
      annotated_port;
  WHERE
    valid_overriding_properties :
      (* The overridden properties are defined in the annotated_port
or in the
         definition of the master_logical_port associated with the
         referenced_master_structure_port *)
      not_exists(QUERY(prop <* overriding_properties |
        NOT (prop.overridden_property IN
port_attributes.new_properties)
          OR
          (prop.overridden_property IN annotated_port.

referenced_master_structure_port.
                                       associated_logical_port.
                                       properties)));
  END_ENTITY;
```

10.2.32.1 Description

An instance_port_annotate is either an input_instance_port_annotate, output_instance_port_annotate, bidirectional_instance_port_annotate, or an unspecified_direction_instance_port_annotate. It is within an occurrence_annotate. It attaches or modifies properties or attributes which are associated with a master_logical_port occurrence within the occurrence. The overridden properties are defined in the annotated_port or in the definition of the master_logical_port associated with the referenced master_structure_port.

10.2.32.2 Used by

capacitance_value occurrence_annotate property property_override

10.2.33 input_instance_port_annotate

```
  ENTITY input_instance_port_annotate
    SUBTYPE OF (instance_port_annotate);
    overriding_dc_fanout_load : OPTIONAL load_value;
    overriding_dc_max_fanin   : OPTIONAL load_value;
  WHERE
    valid_annotated_port:
      'HIERARCHY_MODEL.INPUT_MASTER_LOGICAL_PORT' IN
      TYPEOF(annotated_port.referenced_master_structure_port);
  END_ENTITY;
```

10.2.33.1 Description

An input_instance_port_annotate attaches or modifies properties or attributes which are associated with an input_master_logical_port occurrence within the occurrence.

10.2.34 output_instance_port_annotate

```
  ENTITY output_instance_port_annotate
    SUBTYPE OF (instance_port_annotate);
    overriding_dc_fanin_load : OPTIONAL load_value;
    overriding_dc_max_fanout : OPTIONAL load_value;
  WHERE
    valid_annotated_port:
```

Withdrawn
IECNORM.COM: Click to view the full PDF of IEC 62016:2003

```
      'HIERARCHY_MODEL.OUTPUT_MASTER_LOGICAL_PORT' IN
      TYPEOF(annotated_port.referenced_master_structure_port);
  END_ENTITY;
```

10.2.34.1 Description

An output_instance_port_annotate attaches or modifies properties or attributes which are associated with an output_master_logical_port occurrence within the occurrence.

10.2.35 bidirectional_instance_port_annotate

```
  ENTITY bidirectional_instance_port_annotate
    SUBTYPE OF (instance_port_annotate);
    overriding_dc_fanin_load  : OPTIONAL load_value;
    overriding_dc_fanout_load : OPTIONAL load_value;
    overriding_dc_max_fanin   : OPTIONAL load_value;
    overriding_dc_max_fanout  : OPTIONAL load_value;
  WHERE
    valid_annotated_port:
      'HIERARCHY_MODEL.BIDIRECTIONAL_MASTER_LOGICAL_PORT' IN
      TYPEOF(annotated_port.referenced_master_structure_port);
  END_ENTITY;
```

10.2.35.1 Description

The bidirectional_instance_port_annotate attach or modifies properties or attributes which are associated with a bidirectional_master_logical_port occurrence within the occurrence.

10.2.36 unspecified_direction_instance_port_annotate

```
  ENTITY unspecified_direction_instance_port_annotate
    SUBTYPE OF (instance_port_annotate);
  WHERE
    valid_annotated_port:
      'HIERARCHY_MODEL.UNSPECIFIED_DIRECTION_MASTER_LOGICAL_PORT' IN
      TYPEOF(annotated_port.referenced_master_structure_port);
  END_ENTITY;
```

10.2.36.1 Description

An unspecified_direction_instance_port_annotate attaches or modifies properties or attributes which are associated with an unspecified_direction_master_logical_port occurrence within the occurrence.

10.2.37 signal_annotate

```
  ENTITY signal_annotate;
    annotated_signal      : signal;                        --
reference
    new_properties        : OPTIONAL SET [1:?] OF property;
    overriding_properties : OPTIONAL SET [1:?] OF property_override;
  INVERSE
    containing_expandable_internal_occurrence_hierarchy_annotate
      : SET [0:1] OF

expandable_internal_connectivity_occurrence_hierarchy_annotate
          FOR signal_annotates;
    containing_expandable_internal_connectivity_occurrence_annotate
```

Withdrawn
IECNORM.COM : Click to view the full PDF of IEC 62016:2003

```
      : SET [0:1] OF
expandable_internal_connectivity_occurrence_annotate
          FOR signal_annotates;
  UNIQUE
    unique_signal :
      (* No two signal_annotates in a given context annotate the same
         signal. *)
      containing_expandable_internal_occurrence_hierarchy_annotate,
      containing_expandable_internal_connectivity_occurrence_annotate,
      annotated_signal;
  WHERE
    containment_constraint :
      (* A signal_annotate belongs to either one
occurrence_hierarchy_annotate
         or one occurrence_annotate. *)

SIZEOF(containing_expandable_internal_occurrence_hierarchy_annotate) +

SIZEOF(containing_expandable_internal_connectivity_occurrence_annotate
) = 1;

    valid_overriding_properties :
      (* The overridden properties are defined in the
annotated_signal. *)
      not_exists(QUERY(signalPropertyOverride <*
                       overriding_properties |
        NOT (annotated_signal IN

signalPropertyOverride.overridden_property.containing_signal)));
  END_ENTITY;
```

10.2.37.1 Description

A signal_annotate attaches or modifies properties which are associated with a signal occurrence within the occurrence. A signal_annotate belongs to either one occurrence_hierarchy_annotate or one occurrence_annotate. The overridden properties are defined in the annotated signal.

10.2.37.2 Used by

expandable_internal_connectivity_occurrence_annotate expandable_internal_connectivity_occurrence_hierarchy_annotate property property_override

10.2.38 interconnect_annotate

```
  ENTITY interconnect_annotate;
    annotated_interconnect : referenced_interconnect;       --
reference
    new_properties          : OPTIONAL SET [1:?] OF property;
    overriding_criticality : OPTIONAL integer_token;
    overriding_properties   : OPTIONAL SET [1:?] OF property_override;
  INVERSE

containing_expandable_internal_connectivity_occurrence_hierarchy_annot
ate
      : SET [0:1] OF
        expandable_internal_connectivity_occurrence_hierarchy_annotate
        FOR interconnect_annotates;
    containing_expandable_internal_connectivity_occurrence_annotate
      : SET [0:1] OF
expandable_internal_connectivity_occurrence_annotate FOR
```

Withdrawn
IECNORM.COM : Click to view the full PDF of IEC 62016:2003

```
          interconnect_annotates;
  UNIQUE
    unique_interconnect :
      (* No two interconnect_annotates in the same context annotate
         the same interconnect. *)

containing_expandable_internal_connectivity_occurrence_hierarchy_annot
ate,
      containing_expandable_internal_connectivity_occurrence_annotate,
      annotated_interconnect;
  WHERE
    containment_constraint :
      (* An interconnect_annotate belongs to either one
         occurrence_hierarchy_annotate or one occurrence_annotate. *)

SIZEOF(containing_expandable_internal_connectivity_occurrence_hierarch
y_annotate) +

SIZEOF(containing_expandable_internal_connectivity_occurrence_annotate
) = 1;

    valid_overriding_properties :
      (* The overridden properties are defined in the annotated
         interconnect. *)
      not_exists(QUERY(op <* overriding_properties |
        NOT (annotated_interconnect IN

op.overridden_property.containing_connectivity_generic_bus)
        AND
        NOT (annotated_interconnect IN

op.overridden_property.containing_connectivity_generic_net)));
  END_ENTITY;
```

10.2.38.1 Description

An interconnect_annotate attaches or modifies properties or attributes which are associated with the connectivity_generic_net or a connectivity_generic_bus occurrence within the occurrence. An interconnect_annotate belongs to either one occurrence_hierarchy_annotate or one occurrence_annotate. The overridden properties are defined in the annotated interconnect.

10.2.38.2 Used by

expandable_internal_connectivity_occurrence_annotate
expandable_internal_connectivity_occurrence_hierarchy_annotate property property_override

10.2.39 design_management_model

10.2.39.1 Description

The design_management_model schema provides the design management information. It records a history of modifications. It provides the information needed to trace back to the origin or the owner of the data, and also identifies the software or program name which was responsible for creating the data.

10.2.40 copyright

```
  ENTITY copyright;
    strings : OPTIONAL BAG [1:?] OF string_token;
    year    : SET OF positive_integer_token;
```

Withdrawn
IECNORM.COM: Click to view the full PDF of IEC 62016:2003

```
  INVERSE
    containing_documentation
      : SET [0:1] OF documentation FOR status_of_copyright;
    containing_information_base
      : SET [0:1] OF information_base FOR status_of_copyright;
    containing_library
      : SET [0:1] OF library FOR status_of_copyright;
    containing_cell_representation
      : SET [0:1] OF cell_representation FOR status_of_copyright;
    containing_cluster
      : SET [0:1] OF cluster FOR status_of_copyright;
    containing_design
      : SET [0:1] OF design FOR status_of_copyright;
    containing_cell
      : SET [0:1] OF cell FOR status_of_copyright;
  WHERE
    containment_constraint :
      (* Each "copyright" is defined in only one place *)
      SIZEOF(containing_documentation) +
      SIZEOF(containing_information_base) +
      SIZEOF(containing_library) +
      SIZEOF(containing_cell_representation) +
      SIZEOF(containing_cluster) +
      SIZEOF(containing_design) +
      SIZEOF(containing_cell) = 1;

    unique_year :
      (* The copyright does not have identical years *)
      value_unique(year);
  END_ENTITY;
```

10.2.40.1 Description

The copyright indicates the copyright restrictions on the use of the information base, or associated with a particular object within the file.

10.2.40.2 Used by

cell cell_representation cluster design documentation information_base library

10.2.41 written

```
  ENTITY written;
    author       : OPTIONAL string_token;
    date         : time_stamp;
    data_origin  : OPTIONAL version_information;
    program      : OPTIONAL version_information;
  INVERSE
    containing_documentation
      : SET [0:1] OF documentation FOR status_of_written;
    containing_information_base
      : SET [0:1] OF information_base FOR status_of_written;
    containing_library
      : SET [0:1] OF library FOR status_of_written;
    containing_cell_representation
      : SET [0:1] OF cell_representation FOR status_of_written;
    containing_cluster
      : SET [0:1] OF cluster FOR status_of_written;
    containing_cell
      : SET [0:1] OF cell FOR status_of_written;
    containing_design
```

Withdrawn
IECNORM.COM : Click to view the full PDF of IEC 62016:2003

```
      : SET [0:1] OF design FOR status_of_written;
  WHERE
    containment_constraint :
      (* Each "written" information is defined in only one place. *)
      SIZEOF(containing_documentation) +
      SIZEOF(containing_information_base) +
      SIZEOF(containing_library) +
      SIZEOF(containing_cell_representation) +
      SIZEOF(containing_cluster) +
      SIZEOF(containing_cell) +
      SIZEOF(containing_design) = 1;
  END_ENTITY;
```

10.2.41.1 Description

A written includes information relating to the writer or generator of the object with which the written is associated. It includes a time_stamp and may include program identification, human or organization identification, or location information to help the reader trace the origin of a particular part of an information base.

10.2.41.2 Used by

cell cell_representation cluster design documentation information_base library time_stamp version_information

10.2.42 version_information

```
  ENTITY version_information;
    name    : string_token;
    version : OPTIONAL string_token;
  INVERSE
    containing_written_data_origin
      : SET [0:1] OF written FOR data_origin;
    containing_written_program
      : SET [0:1] OF written FOR program;
  WHERE
    containment_constraint :
      (* Each "version_information" is defined in only one place *)
      SIZEOF(containing_written_data_origin) +
      SIZEOF(containing_written_program) = 1;
  END_ENTITY;
```

10.2.42.1 Description

A version_information provides a revision code which can be used to keep track of the source and creator of the written data. It is intended for human interpretation and should serve as an aid in problem analysis.

10.2.42.2 Used by

written

10.2.43 time_stamp

```
  ENTITY time_stamp;
    recorded_date : date;
    recorded_time : time;
  INVERSE
    containing_written : written FOR date;
  END_ENTITY;
```

Withdrawn
IECNORM.COM : Click to view the full PDF of IEC 62016:2003

10.2.43.1 Description

A time_stamp identifies when the data was created or last modified.

10.2.43.2 Used by

date time written

10.2.44 time

```
ENTITY time;
  hour   : integer_token;
  minute : integer_token;
  second : integer_token;
INVERSE
  containing_time_stamp : time_stamp FOR recorded_time;
WHERE
  valid_hour :
    (* Twenty four hours each day. *)
    { 0 <= hour <= 23 };

  valid_minute :
    (* Sixty minutes each hour. *)
    { 0 <= minute <= 59 };

  valid_second :
    (* Sixty seconds each minute, or sixty one if leap second. *)
    { 0 <= second <= 60 };
END_ENTITY;
```

10.2.44.1 Description

A time specifies the time in hours, minutes and seconds. The time is specified in Universal Time Coordinated.

10.2.44.2 Used by

time_stamp

10.2.45 date

```
ENTITY date;
  year  : positive_integer_token;
  month : positive_integer_token;
  day   : positive_integer_token;
INVERSE
  containing_time_stamp : time_stamp FOR recorded_date;
WHERE
  check_valid_date :
    (* The "date" entity contains a valid date *)
    valid_date(year, month, day);
END_ENTITY;
```

10.2.45.1 Description

A date specifies the date as three integers representing the year, month and day. These are consistent with each other to form a valid date.

Withdrawn
IECNORM.COM : Click to view the full PDF of IEC 62016:2003

10.2.45.2 Used by

time_stamp

10.2.46 valid_date

```
  FUNCTION valid_date
    (yy, mm, dd : positive_integer_token) : BOOLEAN;
    LOCAL
      leap_year : BOOLEAN;
      valid     : BOOLEAN;
    END_LOCAL;
    leap_year := (yy mod 4 = 0) AND ((yy mod 100 <> 0) OR (yy mod 400
= 0));
    CASE mm OF
      1, 3, 5, 7, 8, 10, 12 : valid := dd <= 31;
      4, 6, 9, 11           : valid := dd <= 30;
      2                     : IF leap_year THEN
                                 valid := dd <= 29;
                              ELSE
                                 valid := dd <= 28;
                              END_IF;
      OTHERWISE             : valid := false;
    END_CASE;
    RETURN(valid);
  END_FUNCTION;
```

10.2.46.1 Description

The valid_date function checks every instance of date. It is used to ensure that the day, month and year fields constitute a valid date.

10.2.46.2 Used by

date

10.3 documentation_model

Description

The documentation_model schema describes the documentation provided for an object. A documentation may consists of several sections. Each section may have text and nested sections.

10.3.1 documentation

```
  ENTITY documentation;
    sections            : OPTIONAL LIST [1:?] OF section;
    status_of_copyright : OPTIONAL SET [1:?] OF copyright;
    status_of_written   : OPTIONAL SET [1:?] OF written;
  INVERSE
    containing_connectivity_generic_bus
      : SET [0:1] OF connectivity_generic_bus FOR document;
    containing_connectivity_generic_net
      : SET [0:1] OF connectivity_generic_net FOR document;
    containing_information_base
      : SET [0:1] OF information_base FOR document;
    containing_cell
      : SET [0:1] OF cell FOR document;
    containing_cell_representation
```

Withdrawn
IECNORM.COM : Click to view the full PDF of IEC 62016:2003

```
      : SET [0:1] OF cell_representation FOR document;
    containing_cluster
      : SET [0:1] OF cluster FOR document;
    containing_cell_representation_set
      : SET [0:1] OF cell_representation_set FOR document;
    containing_library
      : SET [0:1] OF library FOR document;
    containing_design
      : SET [0:1] OF design FOR document;
  WHERE
    containment_constraint :
      (* Each "documentation" is defined in only one place *)
      SIZEOF(containing_connectivity_generic_bus) +
      SIZEOF(containing_connectivity_generic_net) +
      SIZEOF(containing_information_base) +
      SIZEOF(containing_cell) +
      SIZEOF(containing_cell_representation) +
      SIZEOF(containing_cluster) +
      SIZEOF(containing_cell_representation_set) +
      SIZEOF(containing_library) +
      SIZEOF(containing_design) = 1;

    unique_status_of_copyright :
      (* The document does not have the same copyright information
         twice *)
      value_unique(status_of_copyright);

    unique_status_of_written :
      (* The document does not have two identical "written"
         information *)
      value_unique(status_of_written);
  END_ENTITY;
```

10.3.1.1 Description

A documentation provides documentation for an object. It may consist of several sections. Each section may have text and nested sections.

10.3.1.2 Used by

cell cell_representation cell_representation_set cluster connectivity_generic_bus connectivity_generic_net copyright design information_base library section written

10.3.2 section

```
  ENTITY section;
    contents : OPTIONAL LIST [1:?] OF section_element;
    title    : string_token;
  INVERSE
    containing_documentation : documentation FOR sections;
  END_ENTITY;
```

10.3.2.1 Description

A section divides a document into section_elements. A section element may be a section or a string_token. A string_token gives a title to section. A section may be nested with other sections.

10.3.2.2 Used by

documentation section_element

Withdrawn
IECNORM.COM: Click to view the full PDF of IEC 62016:2003

10.3.3 section_element

```
TYPE section_element
  = SELECT (section, string_token);
END_TYPE;
```

10.3.3.1 Description

A section_element constitutes the body of a section. It can be a string_token or a nested section.

10.3.3.2 Used by

section

10.4 hierarchy_model

Description

The hierarchy_model schema describes the hierarchical information of a cell which is the basic unit of design. A cell can have multiple views of the same type. Views which share the same interface are grouped into a cluster. Views may also be grouped into a cell_representation_set to indicate a particularly close relationship to each other. In the information base clusters are instantiated within other views.

10.4.1 cell

```
  ENTITY cell
    ABSTRACT SUPERTYPE OF (ONEOF(internal_cell,
                                 external_cell));
    clusters                    : OPTIONAL SET [1:?] OF cluster;
    document                    : OPTIONAL SET [1:?] OF documentation;
    name                        : OPTIONAL name_information;
    properties                  : OPTIONAL SET [1:?] OF property;
    status_of_copyright         : OPTIONAL SET [1:?] OF copyright;
    status_of_written           : OPTIONAL SET [1:?] OF written;
    cell_representation_sets      : OPTIONAL SET [1:?] OF
cell_representation_set;
  INVERSE
    containing_library : library FOR cells;
  WHERE
    unique_status_of_copyright :
      (* A "cell" does not contain identical copyrights *)
      value_unique(status_of_copyright);

    unique_status_of_written :
      (* A "cell" does not contain identical "written"
         information *)
      value_unique(status_of_written);

    unique_document :
      (* A "cell" does not contain identical documents *)
      value_unique(document);
  END_ENTITY;
```

Withdrawn
IECNORM.COM: Click to view the full PDF of IEC 62016:2003

10.4.1.1 Description

A cell is classified into an internal_cell and an external_cell. A cell may contain clusters which may be instantiated later in another cell. Views which share the same interface are grouped into a cluster. Views may also be grouped into a cell_representation_set to indicate a particularly close relationship to each other.

10.4.1.2 Used by

cell_representation_set cluster copyright design documentation library name_information property written

10.4.2 internal_cell

```
  ENTITY internal_cell
    SUBTYPE OF (cell);
    SELF\cell.clusters     : OPTIONAL SET [1:?] OF internal_cluster;
   INVERSE
    containing_internal_library : internal_library FOR cells;
  END_ENTITY;
```

10.4.2.1 Description

An internal_cell is a cell which can be found in an internal_library. An internal_cell may contain implementation information.

10.4.2.2 Used by

internal_cluster internal_library

10.4.3 external_cell

```
  ENTITY external_cell
    SUBTYPE OF (cell);
    SELF\cell.clusters     : OPTIONAL SET [1:?] OF external_cluster;
   INVERSE
    containing_external_library : external_library FOR cells;
  END_ENTITY;
```

10.4.3.1 Description

An external_cell is a cell found in an external_library. An external_cell does not have implementation information, i.e. only external_cell_representations may be specified in an external_cell.

10.4.3.2 Used by

external_cluster external_library

10.4.4 cluster

```
  ENTITY cluster
    ABSTRACT SUPERTYPE OF (ONEOF(internal_cluster,
                                 external_cluster));
    cell_representations    : OPTIONAL SET [1:?] OF
cell_representation;
    cluster_configurations : OPTIONAL SET [1:?] OF
cluster_configuration;
```

Withdrawn
IECNORM.COM : Click to view the full PDF of IEC 62016:2003

```
    document                : OPTIONAL SET [1:?] OF documentation;
    name                    : OPTIONAL name_information;
    interface               : cluster_interface;
    properties              : OPTIONAL SET [1:?] OF property;
    status_of_copyright     : OPTIONAL SET [1:?] OF copyright;
    status_of_written       : OPTIONAL SET [1:?] OF written;
  INVERSE
    containing_cell : cell FOR clusters;
  WHERE
    unique_status_of_copyright :
      (* A "cluster" does not contain two identical copyrights *)
      value_unique(status_of_copyright);

    unique_status_of_written :
      (* A "cluster" does not contain two identical "written"
         information *)
      value_unique(status_of_written);

    unique_document :
      (* A "cluster" does not contain two identical documents *)
      value_unique(document);
  END_ENTITY;
```

10.4.4.1 Description

A cluster is either an internal_cluster or a external_cluster. It is a group of connectivity views which have the same cluster_interface.

10.4.4.2 Used by

cell cell_representation cluster_configuration cluster_interface copyright documentation instance name_information property signal written

10.4.5 internal_cluster

```
  ENTITY internal_cluster
    SUBTYPE OF (cluster);
    SELF\cluster.cell_representations
      : OPTIONAL SET [1:?] OF internal_cell_representation;
    SELF\cluster.cluster_configurations
      : OPTIONAL SET [1:?] OF internal_cluster_configuration;
  INVERSE
    containing_internal_cell : internal_cell FOR clusters;
  END_ENTITY;
```

10.4.5.1 Description

An internal_cluster is a cluster found in an internal_cell. Internal views which share the same interface are grouped into an internal_cluster.

10.4.5.2 Used by

internal_cell internal_cell_representation internal_cluster_instance

10.4.6 external_cluster

```
  ENTITY external_cluster
    SUBTYPE OF (cluster);
    SELF\cluster.cell_representations
      : OPTIONAL SET [1:?] OF external_cell_representation;
```

Withdrawn
IECNORM.COM: Click to view the full PDF of IEC 62016:2003

```
    SELF\cluster.cluster_configurations
      : OPTIONAL SET [1:?] OF external_cluster_configuration;
  INVERSE
    containing_external_cell : external_cell FOR clusters;
  END_ENTITY;
```

10.4.6.1 Description

An external_cluster is a cluster found in an external_cell. External views which share the same interface are grouped into an external_cluster.

10.4.6.2 Used by

external_cell external_cell_representation external_cluster_instance

10.4.7 cluster_configuration

```
  ENTITY cluster_configuration
    ABSTRACT SUPERTYPE OF (ONEOF(external_cluster_configuration,
                                 internal_cluster_configuration));
    name       : OPTIONAL name_information;
    properties : OPTIONAL SET [1:?] OF property;
  INVERSE
    containing_cluster : cluster FOR cluster_configurations;
  END_ENTITY;
```

10.4.7.1 Description

A cluster_configuration is either an internal_cluster_configuration or an external_cluster_configuration.

10.4.7.2 Used by

cluster design_hierarchy instance_configuration name_information property

10.4.8 external_cluster_configuration

```
  ENTITY external_cluster_configuration
    ABSTRACT SUPERTYPE OF
(ONEOF(expandable_external_cluster_configuration,
                                 leaf_external_cluster_configuration))
    SUBTYPE OF (cluster_configuration);
  END_ENTITY;
```

10.4.8.1 Description

An external_cluster_configuration is an expandable or a leaf external cluster configuration.

10.4.8.2 Used by

external_cluster external_design_hierarchy external_instance_configuration

10.4.9 expandable_external_cluster_configuration

```
  ENTITY expandable_external_cluster_configuration
    SUBTYPE OF (external_cluster_configuration);
    chosen_cell_representation  : external_cell_representation; --
reference
```

Withdrawn
IECNORM.COM : Click to view the full PDF of IEC 62016:2003

```
    scoped_global_ports         : OPTIONAL SET [1:?] OF         --
reference
                                    global_port;
  WHERE
    valid_scoped_global_ports :
      (* The referenced global_ports are defined in the containing
         information_base. *)
      scoped_global_ports  <= NVL (containing_cluster.
                                  containing_cell.
                                  containing_library.
                                  containing_information_base.
                                  global_ports, []);

    valid_cell_representation :
      (* The chosen cell representation is in the cluster which is
         configured. *)
      chosen_cell_representation.containing_cluster :=:
containing_cluster;
  END_ENTITY;
```

10.4.9.1 Description

An expandable_external_cluster_configuration selects a cell representation for an external cluster. The referenced global_ports are defined in the containing information_base.

10.4.9.2 Used by

expandable_external_design_hierarchy expandable_external_instance_configuration

10.4.10 leaf_external_cluster_configuration

```
  ENTITY leaf_external_cluster_configuration
    SUBTYPE OF (external_cluster_configuration);
  END_ENTITY;
```

10.4.10.1 Description

A leaf_external_cluster_configuration specifies that there is no further substructure in an expanded design. If an instance is given an instance_configuration which points to a leaf_external_cluster_configuration, it means that there is no further structure in the design_hierarchy below that instance.

10.4.10.2 Used by

leaf_external_design_hierarchy leaf_external_instance_configuration

10.4.11 internal_cluster_configuration

```
  ENTITY internal_cluster_configuration
    ABSTRACT SUPERTYPE OF
(ONEOF(expandable_internal_cluster_configuration,
                                  leaf_internal_cluster_configuration))
    SUBTYPE OF (cluster_configuration);
  END_ENTITY;
```

10.4.11.1 Description

An internal_cluster_configuration may be an expandable or a leaf internal cluster configuration.

IECNORM.COM : Click to view the full PDF of IEC 62016:2003 Withdrawn

10.4.11.2 Used by

internal_cluster internal_design_hierarchy internal_instance_configuration

10.4.12 expandable_internal_cluster_configuration

```
  ENTITY expandable_internal_cluster_configuration
    ABSTRACT SUPERTYPE OF
      (ONEOF (expandable_internal_connectivity_view_configuration))
    SUBTYPE OF (internal_cluster_configuration);
    chosen_cell_representation : internal_cell_representation; --
reference
    scoped_global_ports        : OPTIONAL SET [1:?] OF          --
reference
                                   global_port;
  WHERE
    valid_scoped_global_ports :
      (* The referenced global_ports are defined in the containing
         information_base. *)
      scoped_global_ports  <= NVL (containing_cluster.
                                  containing_cell.
                                  containing_library.
                                  containing_information_base.
                                  global_ports, []);

    valid_cell_representation :
      (* The chosen cell representation is in the cluster which is
         being configured. *)
      chosen_cell_representation.containing_cluster :=:
containing_cluster;
  END_ENTITY;
```

10.4.12.1 Description

An expandable_internal_cluster_configuration is classified into an expandable_internal_connectivity_view_configuration. It gives enough information to expand the hierarchy from that point onwards. It chooses which cell_representation to look at. “Global_port” scopes are established on global_ports in an occurrence hierarchy. If a global_port is scoped, this indicates that any references to that global_port within the occurrence hierarchy are connected together, except in any lower branch that scopes the same global_port. The referenced global_ports are defined in the containing information_base. The chosen cell_representation is in the cluster which is being configured.

IECNORM.COM : Click to view the full PDF of IEC 62016:2003
Withdrawn

10.4.12.2 Used by

expandable_internal_design_hierarchy expandable_internal_instance_configuration

10.4.13 expandable_internal_connectivity_view_configuration

```
  ENTITY expandable_internal_connectivity_view_configuration
    SUBTYPE OF (expandable_internal_cluster_configuration);

SELF\expandable_internal_cluster_configuration.chosen_cell_representat
ion
      : internal_connectivity_view;                      -- reference
    instance_configurations    : OPTIONAL SET [1:?] OF
instance_configuration;
    unconfigured_instances     : OPTIONAL SET [1:?] OF
unconfigured_instance;
  WHERE
```

```
    valid_instance_configurations :
      (* Each instance defined within the chosen cell representation
         is either given an instance_configuration or chosen to
         be unconfigured. *)
      SIZEOF (NVL (instance_configurations, [])) +
      SIZEOF (NVL (unconfigured_instances, [])) =
         SIZEOF (NVL (chosen_cell_representation.instances, []));
  END_ENTITY;
```

10.4.13.1 Description

An expandable_internal_connectivity_view_configuration gives enough information to expand the hierarchy down an internal_connectivity_view. For each instance in that view, it gives an instance_configuration or defines that instance as unconfigured.

10.4.13.2 Used by

expandable_internal_connectivity_design_hierarchy expandable_internal_connectivity_instance_configuration instance_configuration internal_connectivity_view unconfigured_instance

10.4.14 leaf_internal_cluster_configuration

```
  ENTITY leaf_internal_cluster_configuration
    SUBTYPE OF (internal_cluster_configuration);
  END_ENTITY;
```

10.4.14.1 Description

A leaf_internal_cluster_configuration specifies that there is no further substructure in an expanded design_hierarchy. If an instance is given an instance_configuration which points to a leaf_internal_cluster_configuration, it means that there is no further structure in the design_hierarchy below that instance.

10.4.14.2 Used by

leaf_internal_design_hierarchy leaf_internal_instance_configuration

10.4.15 instance_configuration

```
  ENTITY instance_configuration
    ABSTRACT SUPERTYPE OF (ONEOF (external_instance_configuration,
                                  internal_instance_configuration));
    configured_instance          : instance;              -- reference
    chosen_cluster_configuration : cluster_configuration; -- reference
  INVERSE
    containing_expandable_internal_cluster_configuration
      : expandable_internal_connectivity_view_configuration
        FOR instance_configurations;
  UNIQUE
    unique_instance :
      (* There is at most one instance_configuration for a given
         instance within an
         expandable_internal_connectivity_view_configuration. *)
      containing_expandable_internal_cluster_configuration,
      configured_instance;
  WHERE
    valid_chosen_cluster_configuration :
      (* The chosen cluster configuration is defined in the
instantiated
         cluster of the chosen instance. *)
```

Withdrawn
IECNORM.COM : Click to view the full PDF of IEC 62016:2003

```
      chosen_cluster_configuration.containing_cluster :=:
      configured_instance.instantiated_cluster;
  END_ENTITY;
```

10.4.15.1 Description

An instance_configuration is either an external_instance_configuration or an internal_ instance_configuration. It selects a cluster_configuration for an instance. The chosen cluster_configuration is defined in the instantiated cluster of the configured instance.The chosen cluster_configuration is defined in the instantiated cluster of the chosen instance.

10.4.15.2 Used by

expandable_internal_connectivity_view_configuration occurrence_annotate

10.4.16 external_instance_configuration

```
  ENTITY external_instance_configuration
    ABSTRACT SUPERTYPE OF
      (ONEOF (expandable_external_instance_configuration,
              leaf_external_instance_configuration))
    SUBTYPE OF (instance_configuration);
    SELF\instance_configuration.chosen_cluster_configuration :
             external_cluster_configuration;  -- reference
  END_ENTITY;
```

10.4.16.1 Description

An external_instance_configuration is either an expandable_external_instance_configuration or a leaf_external_instance_configuration. It selects an external_cluster_configuration for an instance.

10.4.16.2 Used by

external_occurrence_annotate

10.4.17 expandable_external_instance_configuration

```
  ENTITY expandable_external_instance_configuration
    SUBTYPE OF (external_instance_configuration);
    SELF\instance_configuration.chosen_cluster_configuration :
             expandable_external_cluster_configuration;  -- reference
  END_ENTITY;
```

10.4.17.1 Description

An expandable_external_instance_configuration selects an expandable_external_cluster_ configuration for an instance.

10.4.17.2 Used by

expandable_external_occurrence_annotate

10.4.18 leaf_external_instance_configuration

```
  ENTITY leaf_external_instance_configuration
    SUBTYPE OF (external_instance_configuration);
    SELF\instance_configuration.chosen_cluster_configuration :
```

Withdrawn
IECNORM.COM : Click to view the full PDF of IEC 62016:2003

```
              leaf_external_cluster_configuration;  -- reference
  END_ENTITY;
```

10.4.18.1 Description

A leaf_external_instance_configuration selects a expandable_external_cluster_configuration for an instance.

10.4.18.2 Used by

leaf_external_occurrence_annotate

10.4.19 internal_instance_configuration

```
  ENTITY internal_instance_configuration
    ABSTRACT SUPERTYPE OF
      (ONEOF (expandable_internal_instance_configuration,
              leaf_internal_instance_configuration))
    SUBTYPE OF (instance_configuration);
    SELF\instance_configuration.chosen_cluster_configuration :
              internal_cluster_configuration;  -- reference
  END_ENTITY;
```

10.4.19.1 Description

An internal_instance_configuration is either an expandable_internal_instance_configuration or a leaf_internal_instance_configuration. It selects an internal_cluster_configuration for an instance.

10.4.19.2 Used by

internal_occurrence_annotate

10.4.20 expandable_internal_instance_configuration

```
  ENTITY expandable_internal_instance_configuration
    ABSTRACT SUPERTYPE OF
      (ONEOF
(expandable_internal_connectivity_instance_configuration))
    SUBTYPE OF (internal_instance_configuration);
    SELF\instance_configuration.chosen_cluster_configuration :
              expandable_internal_cluster_configuration;  -- reference
  END_ENTITY;
```

10.4.20.1 Description

An expandable_internal_instance_configuration selects an expandable_internal_cluster_configuration for an instance.

10.4.20.2 Used by

expandable_internal_occurrence_annotate

10.4.21 expandable_internal_connectivity_instance_configuration

```
  ENTITY expandable_internal_connectivity_instance_configuration
    SUBTYPE OF (expandable_internal_instance_configuration);
    SELF\instance_configuration.chosen_cluster_configuration :
```

Withdrawn
IECNORM.COM : Click to view the full PDF of IEC 62016:2003

```
           expandable_internal_connectivity_view_configuration;  --
reference
  END_ENTITY;
```

10.4.21.1 Description

An expandable_internal_connectivity_instance_configuration selects an expandable_internal_ connectivity_view_configuration for an instance.

10.4.21.2 Used by

expandable_internal_connectivity_occurrence_annotate

10.4.22 leaf_internal_instance_configuration

```
  ENTITY leaf_internal_instance_configuration
    SUBTYPE OF (internal_instance_configuration);
    SELF\instance_configuration.chosen_cluster_configuration :
             leaf_internal_cluster_configuration;  -- reference
  END_ENTITY;
```

10.4.22.1 Description

A leaf_internal_instance_configuration selects a leaf_internal_cluster_configuration for an instance.

10.4.22.2 Used by

leaf_internal_occurrence_annotate

10.4.23 unconfigured_instance

```
  ENTITY unconfigured_instance;
    referenced_instance : instance;                     -- reference
  INVERSE
    containing_expandable_internal_connectivity_view_configuration
      : expandable_internal_connectivity_view_configuration
        FOR unconfigured_instances;
  UNIQUE
    (* An instance is only unconfigured at most once in a given
       expandable_internal_connectivity_view_configuration. *)
    unique_instance :
      containing_expandable_internal_connectivity_view_configuration,
      referenced_instance;
  END_ENTITY;
```

10.4.23.1 Description

An unconfigured_instance specifies that an instance is deliberately specified to be unconfigured. It is not possible to do a complete expansion of a design_hierarchy containing one or more unconfigured_instances.

10.4.23.2 Used by

expandable_internal_connectivity_view_configuration

Withdrawn
IECNORM.COM : Click to view the full PDF of IEC 62016:2003

10.4.24 cell_representation_set

```
  ENTITY cell_representation_set;
     cell_representations : OPTIONAL SET [1:?] OF                 --
reference
                             cell_representation;
    document               : OPTIONAL SET [1:?] OF documentation;
    name                   : OPTIONAL name_information;
    member_cell_representation_sets
                          : OPTIONAL SET [1:?] OF
cell_representation_set; -- reference
    properties             : OPTIONAL SET [1:?] OF private_property;
    reason                 : OPTIONAL string_token;
  INVERSE
    containing_cell : cell FOR cell_representation_sets;
  WHERE
     valid_cell_representations :
      (* All cell_represenattions are defined in the containing cell
of the
          cell_representation_set. *)
      not_exists(QUERY(memberCellRepresentation <*
cell_representations |
         memberCellRepresentation.containing_cluster.containing_cell
:<>:
         containing_cell));

    valid_member_cell_representation_sets :
      (* All member_cell_representation_sets belong to the
          containing cell of the cell_representation_set. *)
      not_exists(QUERY(ncrs <* member_cell_representation_sets |
        ncrs.containing_cell :<>: containing_cell));

    non_recursive_member_cell_representation_sets :
      (* Non of the member_cell_representation_sets have a current
        cell_representation_set as a member either directly or
indirectly *)
      check_non_recursive_member_cell_representation_sets(SELF,
                                      NVL
(member_cell_representation_sets, []));

    unique_document :
      (* A "cell_representation_set" cannot contain two identical
documents *)
      value_unique(document);
  END_ENTITY;
```

IECNORM.COM: Click to view the full PDF of IEC 62016:2003 Withdrawn

10.4.24.1 Description

A cell_representation_set is a group of cell_representations which belong to the same cell. There are various reasons for grouping, for example, a flattened netlist and a hierarchy netlist. It is used to express a relationship between any number of cell_representations of any type.

10.4.24.2 Used by

cell cell_representation_set documentation name_information private_property

10.4.25 check_non_recursive_member_cell_representation_sets

```
  FUNCTION check_non_recursive_member_cell_representation_sets
    (crs  : cell_representation_set;
     ncrs : SET OF cell_representation_set) : BOOLEAN;
    REPEAT i := 1 TO SIZEOF(ncrs);
      IF crs :=: ncrs[i] THEN
        RETURN(FALSE);
      ELSE
        IF NOT check_non_recursive_member_cell_representation_sets
                 (crs, NVL (ncrs[i].member_cell_representation_sets,
[])) THEN
          RETURN(FALSE);
        END_IF;
      END_IF;
    END_REPEAT;
    RETURN(TRUE);
  END_FUNCTION;
```

10.4.25.1 Description

The check_non_recursive_member_cell_representation_sets function checks that a cell_representation_set is not contained in its “member_cell_representation_sets” set either directly or indirectly.

10.4.25.2 Used by

cell_representation_set

10.4.26 cluster_interface

```
  ENTITY cluster_interface;
    designator                 : OPTIONAL string_token;
    joined_master_logical_port_sets : OPTIONAL SET [1:?] OF
joined_master_logical_port_set;
    must_join_port_sets        : OPTIONAL SET [1:?] OF must_join_port_set;
    permutables                : OPTIONAL SET [1:?] OF
permutable_relationship;
    structure_port_bundles     : OPTIONAL SET [1:?] OF
                                   master_structure_port_bundle;
    logical_port_bundles       : OPTIONAL SET [1:?] OF
master_logical_port_bundle;
    structure_ports            : OPTIONAL SET [1:?] OF
master_structure_port;
    logical_ports              : OPTIONAL SET [1:?] OF
master_logical_port;
    port_structures            : OPTIONAL SET [1:?] OF port_structure;
    units                      : cluster_interface_units;
    weak_joined_port_sets      : OPTIONAL SET [1:?] OF
weak_joined_port_set;
  INVERSE
    containing_cluster : cluster FOR interface;
  END_ENTITY;
```

10.4.26.1 Description

A cluster_interface defines objects which can be seen and relationships which hold for all views within a cluster. It includes definitions of master_logical_ports, master_structure_ports, master_logical_port_bundles and master_structure_port_bundles.

10.4.26.2 Used by

cell_representation cluster cluster_interface_units external_cell_representation internal_cell_representation joined_master_logical_port_set master_logical_port master_logical_port_

Withdrawn
IECNORM.COM : Click to view the full PDF of IEC 62016:2003

bundle master_structure_port master_structure_port_bundle must_join_port_set permutable_relationship port_structure weak_joined_port_set

10.4.27 port_structure

```
  ENTITY port_structure
    ABSTRACT SUPERTYPE OF (ONEOF (ordered_port_structure,
                                  unordered_port_structure));
    name       : OPTIONAL name_information;
    properties : OPTIONAL SET [1:?] OF property;
    members    : SET [1:?] OF                                  --
reference

master_structure_port_or_master_structure_port_bundle;
  INVERSE
    containing_cluster_interface
      : cluster_interface FOR port_structures;
  WHERE
    valid_ports_and_port_bundles :
      (* All the structure ports and port bundles referenced by
         "port_structure" are defined in the containing
cluster_interface *)
      not_exists(QUERY(mp <* members |
                 NOT ((('HIERARCHY_MODEL.MASTER_STRUCTURE_PORT' IN
TYPEOF(mp))
                       AND (mp IN

containing_cluster_interface.structure_ports)) OR
                     (('HIERARCHY_MODEL.MASTER_STRUCTURE_PORT_BUNDLE'
IN
                        TYPEOF(mp))
                       AND (mp IN

containing_cluster_interface.structure_port_bundles)))));

    valid_members :
      (* The same master_logical_port cannot be associated with more
that one
         master_structure_port referenced by the port_structure
either
         directly or indirectly. *)
      check_valid_port_structure_members(SELF);
  END_ENTITY;
```

IECNORM.COM : Click to view the full PDF of IEC 62016:2003 Withdrawn

10.4.27.1 Description

A port_structure can be either an ordered_port_structure or an unordered_port_structure. It describes a possible structuring of the structure ports and port bundles of a cluster_interface. The same master_logical_port cannot be associated with more than one master_structure_port referenced by the port_structure either directly or indirectly.

10.4.27.2 Used by

cluster_interface connectivity_instance_implementation name_information property

10.4.28 ordered_port_structure

```
  ENTITY ordered_port_structure
    SUBTYPE OF (port_structure);
    SELF\port_structure.members                               --
reference
      : LIST [1:?] OF UNIQUE
          master_structure_port_or_master_structure_port_bundle;
  END_ENTITY;
```

10.4.28.1 Description

An ordered_port_structure describes a possible structuring of the structure ports and port_bundles of a cluster_interface and defines an ordering for the structured port.

10.4.29 unordered_port_structure

```
  ENTITY unordered_port_structure
    SUBTYPE OF (port_structure);
  END_ENTITY;
```

10.4.29.1 Description

An unordered_port_structure describes a possible structuring of the structure ports and port_bundles of a cluster interface without defining an ordering for the structured port.

10.4.30 check_valid_port_structure_members

```
  FUNCTION check_valid_port_structure_members(ps : port_structure) :
BOOLEAN;
  LOCAL
    port_set : SET [0:?] OF master_logical_port := [];
    tmp      : LIST [0:?] OF master_logical_port;
  END_LOCAL;
  REPEAT i:=1 TO SIZEOF(ps.members);
    IF 'HIERARCHY_MODEL.MASTER_STRUCTURE_PORT' IN
TYPEOF(ps.members[i])
    THEN
      IF ps.members[i].associated_logical_port IN port_set
      THEN
        RETURN(FALSE);
      ELSE
        port_set := port_set + ps.members[i].associated_logical_port;
      END_IF;
    ELSE
      tmp :=
ps.members[i].associated_logical_port_bundle.flattened_port_list;
      REPEAT j:=1 TO SIZEOF(tmp);
        IF tmp[j] IN port_set
        THEN
          RETURN(FALSE);
        ELSE
          port_set := port_set + tmp[j];
        END_IF;
      END_REPEAT;
    END_IF;
  END_REPEAT;
  RETURN(TRUE);
  END_FUNCTION;
```

Withdrawn IECNORM.COM : Click to view the full PDF of IEC 62016:2003

10.4.30.1 Description

The check_valid_port_structure_members function returns TRUE if no master_logical_port is associated more than once with the master_structure_ports that are referenced by a port_structure, directly or indirectly.

10.4.30.2 Used by

port_structure

10.4.31 cluster_interface_units

```
  ENTITY cluster_interface_units;
    set_capacitance : capacitance_unit;
  INVERSE
    containing_cluster_interface : cluster_interface FOR units;
  END_ENTITY;
```

10.4.31.1 Description

A cluster_interface_units sets the scaling for the interface of a cluster. In a cluster_interface, it is appropriate to set a capacitance scale.

10.4.31.2 Used by

capacitance_unit cluster_interface

10.4.32 master_logical_port

```
  ENTITY master_logical_port
    ABSTRACT SUPERTYPE OF (ONEOF(input_master_logical_port,
                                 output_master_logical_port,
                                 bidirectional_master_logical_port,

unspecified_direction_master_logical_port));
    default_connection : OPTIONAL global_port;        -- reference
    ac_load            : OPTIONAL capacitance_value;
    unused_internally  : boolean_token;
    name               : OPTIONAL name_information;
    properties         : OPTIONAL SET [1:?] OF property;
  DERIVE
    size : INTEGER := 1;
  INVERSE
    containing_interface : cluster_interface FOR logical_ports;
    related_permutable_master_port_sets : SET [0:1] OF
permutable_master_port_set
                                   FOR members;
    related_non_permutable_master_logical_port_sets :
                               SET [0:1] OF
non_permutable_master_logical_port_set
                                   FOR members;
    related_joined_master_logical_port_sets : SET [0:1] OF
joined_master_logical_port_set
                                   FOR member_ports;
    related_must_join_port_sets : SET [0:1] OF must_join_port_set
                                   FOR member_ports;
    related_weak_joined_port_sets : SET [0:1] OF weak_joined_port_set
                                   FOR member_ports;
  WHERE
    valid_default_connection :
```

Withdrawn
IECNORM.COM : Click to view the full PDF of IEC 62016:2003

```
      (* The referenced global_port is defined in the containing
          information_base. *)
      default_connection.containing_information_base :=:
        containing_interface.containing_cluster.containing_cell.
        containing_library.containing_information_base;

    valid_unused :
      (* If a master_logical_port is unused internally, it does not
have a
         default_connection. *)
      NOT (unused_internally AND EXISTS(default_connection));

    valid_permutables :
      (* A port is referenced at most once within a permutable or
         non_permutable in an interface. *)
      SIZEOF (related_permutable_master_port_sets) +
      SIZEOF (related_non_permutable_master_logical_port_sets) <= 1;

    valid_port_relationships :
      (* A port is referenced at most once in joined, must_join and
         weak_joined. *)
      SIZEOF (related_joined_master_logical_port_sets) +
      SIZEOF (related_must_join_port_sets) +
      SIZEOF (related_weak_joined_port_sets) <= 1;
  END_ENTITY;
```

10.4.32.1 Description

A master_logical_port is defined within the interface of a cluster of a cell. The size of a master_logical_port is always one. The unused_internally attribute is a boolean flag indicating whether or not the master_logical_port is being used by the views. The default_connection indicates a connection to a global_port which is to be made if no other connection is specified for the port when the cluster is instantiated. If a master_logical_port is defined to be unused, it does not have a default_connection. The referenced global_port is defined in the containing information_base. A port is referenced at most once within a permutable or non_permutable in an interface, in joined, must_join and weak_join.

10.4.32.2 Used by

capacitance_value cluster_interface instance_member_logical_port instance_port_attributes joined_master_logical_port_set local_master_port_bundle master_logical_port_bundle master_logical_port_or_master_logical_port_bundle master_port_annotate master_structure_port must_join_port_set name_information non_permutable_master_logical_port_set permutable_master_port_set property signal weak_joined_port_set

10.4.33 input_master_logical_port

```
  ENTITY input_master_logical_port
    SUBTYPE OF (master_logical_port);
    dc_fanout_load : OPTIONAL load_value;
    dc_max_fanin   : OPTIONAL load_value;
  END_ENTITY;
```

10.4.33.1 Description

An input_master_logical_port is an input port which is defined in the interface. The signal information flows into the cell at an input_master_logical_port.

IECNORM.COM : Click to view the full PDF of IEC 62016:2003
Withdrawn

10.4.33.2 Used by

input_instance_port_attributes input_master_port_annotate

10.4.34 output_master_logical_port

```
ENTITY output_master_logical_port
  SUBTYPE OF (master_logical_port);
  dc_fanin_load : OPTIONAL load_value;
  dc_max_fanout : OPTIONAL load_value;
END_ENTITY;
```

10.4.34.1 Description

An output_master_logical_port describes an output port which is defined in the interface. The signal information flows out of the cell at an output_master_logical_port.

10.4.34.2 Used by

output_instance_port_attributes output_master_port_annotate

10.4.35 bidirectional_master_logical_port

```
ENTITY bidirectional_master_logical_port
  SUBTYPE OF (master_logical_port);
  dc_fanin_load  : OPTIONAL load_value;
  dc_fanout_load : OPTIONAL load_value;
  dc_max_fanin   : OPTIONAL load_value;
  dc_max_fanout  : OPTIONAL load_value;
END_ENTITY;
```

10.4.35.1 Description

A bidirectional_master_logical_port describes a bidirectional port which is defined in the interface. The signal information flows both ways through the port.

10.4.35.2 Used by

bidirectional_instance_port_attributes bidirectional_master_port_annotate

10.4.36 unspecified_direction_master_logical_port

```
ENTITY unspecified_direction_master_logical_port
  SUBTYPE OF (master_logical_port);
END_ENTITY;
```

10.4.36.1 Description

An unspecified_direction_master_logical_port describes a master_logical_port for which the direction of the signal information flow is not defined.

10.4.36.2 Used by

unspecified_direction_instance_port_attributes unspecified_direction_master_port_annotate

Withdrawn
IECNORM.COM: Click to view the full PDF of IEC 62016:2003

10.4.37 master_logical_port_bundle

```
  ENTITY master_logical_port_bundle;
    member_ports : LIST [1:?] OF UNIQUE                      --
reference

master_logical_port_or_master_logical_port_bundle;
    name          : OPTIONAL name_information;
    properties    : OPTIONAL SET [1:?] OF property;
  DERIVE
    flattened_port_list : LIST OF UNIQUE master_logical_port  --
reference
      := flatten_master_logical_port_bundle(member_ports);
    size                : INTEGER
      := SIZEOF(flattened_port_list);
  INVERSE
    containing_interface : cluster_interface FOR logical_port_bundles;
  WHERE
    valid_member_ports :
      (* All member master_logical_ports or
master_logical_port_bundles are
         defined in the containing cluster_interface. *)
      not_exists(QUERY(msp <* member_ports |
        msp.containing_interface :<>: containing_interface));

    valid_size :
      (* The size of the flattened master_logical_port list is greater
than
         or equal to one. *)
      size >= 1;
  END_ENTITY;
```

10.4.37.1 Description

A master_logical_port_bundle is defined within the interface of a cluster of a cell. It provides a grouping for master_logical_ports and other master_logical_port_bundles. The size of a master_logical_port_bundle is greater than, or equal to, one. All member master_logical_ports or master_logical_port_bundles are defined in the containing cluster_interface.

10.4.37.2 Used by

cluster_interface instance_port_bundle_attributes master_logical_port_or_master_logical_port_bundle master_structure_port_bundle name_information property

10.4.38 master_structure_port

```
  ENTITY master_structure_port;
    associated_logical_port : master_logical_port;           --
reference
    designator              : OPTIONAL string_token;
    name                    : OPTIONAL name_information;
  DERIVE
    size : INTEGER := 1;
  INVERSE
    containing_interface : cluster_interface FOR structure_ports;
  WHERE
    valid_associated_logical_port :
      (* The associated_logical_port is defined in the containing
         cluster interface *)
```

IECNORM.COM : Click to view the full PDF of IEC 62016:2003
Withdrawn

```
      associated_logical_port.containing_interface :=:
containing_interface;
  END_ENTITY;
```

10.4.38.1 Description

A master_structure_port is defined within the interface of a cluster of a cell.The size of a master_structure_port is always one. It has information about its designator. The associated master_logical_port is defined in the containing cluster_interface. It indicates the availability of the associated master_logical_port for structural connectivity.

10.4.38.2 Used by

cluster_interface connectivity_generic_net instance_member_structure_port instance_structure_port master_structure_port_or_master_structure_port_bundle name_information

10.4.39 master_structure_port_bundle

```
  ENTITY master_structure_port_bundle;
    associated_logical_port_bundle : master_logical_port_bundle;  --
reference
    name                           : OPTIONAL name_information;
  DERIVE
    size : INTEGER := associated_logical_port_bundle.size;
  INVERSE
    containing_interface : cluster_interface FOR
structure_port_bundles;
  WHERE
    valid_associated_logical_port_bundle :
      (* The associated signal port bundle is defined in the
containing
         cluster interface *)
      associated_logical_port_bundle.containing_interface :=:
      containing_interface;
  END_ENTITY;
```

10.4.39.1 Description

A master_structure_port_bundle is defined within the interface of a cluster of a cell. It indicates the availability of the associated master_logical_port_bundle for structural connectivity. The associated signal port group is defined in the containing cluster_interface

10.4.39.2 Used by

cluster_interface connectivity_generic_bus instance_structure_port_bundle master_structure_port or_master_structure_port_bundle name_information

10.4.40 master_structure_port_or_master_structure_port_bundle

```
  TYPE master_structure_port_or_master_structure_port_bundle
    = SELECT(master_structure_port, master_structure_port_bundle);
  END_TYPE;
```

10.4.40.1 Description

A master_structure_port_or_master_structure_port_bundle is either a master_structure_port or a master_structure_port_bundle.

IECNORM.COM : Click to view the full PDF of IEC 62016:2003
Withdrawn

10.4.40.2 Used by

ordered_port_structure port_structure

10.4.41 local_master_port_bundle

```
  ENTITY local_master_port_bundle;
    member_ports : LIST [2:?] OF UNIQUE                         --
reference

master_logical_port_or_master_logical_port_bundle;
    name          : OPTIONAL name_information;
    properties    : OPTIONAL SET [1:?] OF property;
  DERIVE
    flattened_port_list : LIST OF UNIQUE master_logical_port    --
reference
      := flatten_local_master_port_bundle(member_ports);
    size                : INTEGER
      := SIZEOF(flattened_port_list);
  INVERSE
    containing_internal_connectivity_view
      : internal_connectivity_view FOR local_master_port_bundles;
  WHERE
    valid_member_ports :
      (* All member master_logical_ports or
master_logical_port_bundles are
         defined in the containing cluster_interface. *)
      not_exists(QUERY(mp <* member_ports |
        mp.containing_interface :<>:
          containing_internal_connectivity_view.interface));

    valid_size :
      (* The size of the flattened master_logical_port list is greater
than
         two. *)
      size >= 2;
  END_ENTITY;
```

Withdrawn IECNORM.COM : Click to view the full PDF of IEC 62016:2003

10.4.41.1 Description

A local_master_port_bundle is defined within an internal_connectivity_view. It provides a grouping of master_logical_ports and master_logical_port_bundles which can be used to support structural connectivity. All “member master_logical_ports” or master_logical_port_bundles are defined in the containing cluster_interface. The size of the flattened master_logical_port list is greater than or equal to two.

10.4.41.2 Used by

connectivity_generic_bus internal_connectivity_view name_information property

10.4.42 flatten_local_master_port_bundle

```
  FUNCTION flatten_local_master_port_bundle
     (bundle_members :
        LIST OF UNIQUE
master_logical_port_or_master_logical_port_bundle)
        : LIST OF UNIQUE master_logical_port;
     LOCAL
       i, j         : INTEGER;
       position     : INTEGER := 1;
```

```
        port_list   : LIST OF master_logical_port := [ ];
        member_list : LIST OF master_logical_port := [ ];
      END_LOCAL;

      REPEAT i := 1 TO SIZEOF(bundle_members);
        IF 'HIERARCHY_MODEL.MASTER_LOGICAL_PORT' IN
TYPEOF(bundle_members[i])
        THEN
          INSERT(port_list, bundle_members[i], position);
          position := position + 1;
        ELSE
          member_list :=

flatten_local_master_port_bundle(bundle_members[i].member_ports);
          REPEAT j := 1 TO SIZEOF(member_list);
            INSERT(port_list, member_list[j], position);
            position := position + 1;
          END_REPEAT;
        END_IF;
      END_REPEAT;
      RETURN(port_list);
    END_FUNCTION;
```

10.4.42.1 Description

The flatten_local_master_port_bundle function returns the flattened list of master_logical_ports of a local_master_port_bundle.

10.4.42.2 Used by

local_master_port_bundle

10.4.43 master_logical_port_or_master_logical_port_bundle

```
  TYPE master_logical_port_or_master_logical_port_bundle
    = SELECT (master_logical_port, master_logical_port_bundle);
  END_TYPE;
```

10.4.43.1 Description

A master_logical_port_or_master_logical_port_bundle is either a master_logical_port or a master_logical_port_bundle.

10.4.43.2 Used by

local_master_port_bundle master_logical_port_bundle

10.4.44 flatten_master_logical_port_bundle

```
  FUNCTION flatten_master_logical_port_bundle
    (bundle_members :
        LIST OF UNIQUE
master_logical_port_or_master_logical_port_bundle)
        : LIST OF UNIQUE master_logical_port;
    LOCAL
      i, j        : INTEGER;
      position    : INTEGER := 1;
      port_list   : LIST OF master_logical_port := [ ];
      member_list : LIST OF master_logical_port := [ ];
    END_LOCAL;
```

Withdrawn
IECNORM.COM: Click to view the full PDF of IEC 62016:2003

```
    REPEAT i := 1 TO SIZEOF(bundle_members);
      IF 'HIERARCHY_MODEL.MASTER_LOGICAL_PORT' IN
TYPEOF(bundle_members[i])
      THEN
        INSERT(port_list, bundle_members[i], position);
        position := position + 1;
      ELSE
        member_list :=

flatten_master_logical_port_bundle(bundle_members[i].member_ports);
        REPEAT j := 1 TO SIZEOF(member_list);
          INSERT(port_list, member_list[j], position);
          position := position + 1;
        END_REPEAT;
      END_IF;
    END_REPEAT;
    RETURN(port_list);
  END_FUNCTION;
```

10.4.44.1 Description

The flatten_master_logical_port_bundle function returns the flattened list of master_logical_ports of a master_logical_port_bundle.

10.4.44.2 Used by

master_logical_port_bundle

10.4.45 permutable_relationship

```
  ENTITY permutable_relationship
    ABSTRACT SUPERTYPE OF (ONEOF(permutable_structure,
                                 permutable_master_port_set));
    size : INTEGER;
  INVERSE
    containing_cluster_interface
      : SET [0:1] OF cluster_interface FOR permutables;
    containing_permutable_structure
      : SET [0:1] OF permutable_structure FOR permutable_members;
    containing_non_permutable_structure
      : SET [0:1] OF non_permutable_structure FOR permutable_members;
  WHERE
    containment_constraint :
      (* A permutable_relationship belongs to either one
cluster_interface, one
         permutable_structure or one non_permutable_structure. *)
      SIZEOF(containing_cluster_interface) +
      SIZEOF(containing_permutable_structure) +
      SIZEOF(containing_non_permutable_structure) = 1;
  END_ENTITY;
```

10.4.45.1 Description

A permutable_relationship is either a permutable_structure or a permutable_master_port_set. It describes master_logical_ports or groups of master_logical_ports which can be interchanged. A permutable_relationship belongs to either one cluster_interface, one permutable_structure or one non_permutable_structure.

Withdrawn
IECNORM.COM : Click to view the full PDF of IEC 62016:2003

10.4.45.2 Used by

cluster_interface non_permutable_structure permutable_structure

10.4.46 permutable_master_port_set

```
  ENTITY permutable_master_port_set
    SUBTYPE OF (permutable_relationship);
    members : OPTIONAL SET [1:?] OF master_logical_port;      --
reference
  DERIVE
    SELF\permutable_relationship.size : INTEGER := SIZEOF(NVL
(members, []));
  END_ENTITY;
```

10.4.46.1 Description

A permutable_master_port_set describes a relationship between a set of master_logical_ports in which each member of the set is interchangeable.

10.4.46.2 Used by

master_logical_port

10.4.47 permutable_structure

```
  ENTITY permutable_structure
    SUBTYPE OF (permutable_relationship);
    non_permutable_members : OPTIONAL SET [1:?] OF
non_permutable_relationship;
    permutable_members     : OPTIONAL SET [1:?] OF
permutable_relationship;
  DERIVE
    SELF\permutable_relationship.size : INTEGER
      := permutable_structure_size(NVL (non_permutable_members, []),
                                   NVL (permutable_members, []));
   WHERE
    same_member_size :
      (* Each permutable_member and non_permutable_member within a
         permutable_structure is the same size. *)
      permutable_structure_equal_member_size(NVL
(non_permutable_members, []),
                                              NVL (permutable_members,
[]));
  END_ENTITY;
```

10.4.47.1 Description

A permutable_structure indicates that its member structures may be exchanged as a whole but not partially. Each permutable_member and non-permutable member within a permutable_structure is the same size.

10.4.47.2 Used by

non_permutable_relationship permutable_relationship

Withdrawn
IECNORM.COM : Click to view the full PDF of IEC 62016:2003

10.4.48 permutable_structure_size

```
  FUNCTION permutable_structure_size
    (non_permutable_members : SET OF non_permutable_relationship;
     permutable_members     : SET OF permutable_relationship) :
INTEGER;
    LOCAL
      i         : INTEGER;
      temp_size : INTEGER := 0;
    END_LOCAL;

    REPEAT i := 1 TO HIINDEX(permutable_members);
      temp_size := temp_size + permutable_members[i].size;
    END_REPEAT;

    REPEAT i := 1 TO HIINDEX(non_permutable_members);
      temp_size := temp_size + non_permutable_members[i].size;
    END_REPEAT;
    RETURN(temp_size);
  END_FUNCTION;
```

10.4.48.1 Description

The permutable_structure_size function returns the size of a permutable_structure. The size is the sum of all its member structure sizes.

10.4.48.2 Used by

permutable_structure

10.4.49 permutable_structure_equal_member_size

```
  FUNCTION permutable_structure_equal_member_size
    (non_permutable_members : SET OF non_permutable_relationship;
     permutable_members     : SET OF permutable_relationship) :
BOOLEAN;
    LOCAL
      i          : INTEGER;
      valid_size : INTEGER;
    END_LOCAL;

    IF there_exists(non_permutable_members) THEN
      valid_size := non_permutable_members[1].size;
    ELSE
      IF there_exists(permutable_members) THEN
         valid_size := permutable_members[1].size;
      END_IF;
    END_IF;
    RETURN(not_exists(QUERY(non_permutable_member <*
non_permutable_members |
             non_permutable_member.size <> valid_size))
               AND
           not_exists(QUERY(permutable_member <* permutable_members |
             permutable_member.size <> valid_size)));
  END_FUNCTION;
```

10.4.49.1 Description

The permutable_structure_equal_member_size function returns true if all member structures of a permutable_structure has the same size.

Withdrawn
IECNORM.COM : Click to view the full PDF of IEC 62016:2003

10.4.49.2 Used by

permutable_structure

10.4.50 non_permutable_relationship

```
  ENTITY non_permutable_relationship
    ABSTRACT SUPERTYPE OF (ONEOF(non_permutable_structure,

non_permutable_master_logical_port_set));
    size : INTEGER;
  INVERSE
    containing_permutable_structure
      : permutable_structure FOR non_permutable_members;
  END_ENTITY;
```

10.4.50.1 Description

A non_permutable_relationship is either a non_permutable_structure or a non_permutable_master_logical_port_set. It describes master_logical_ports or groups of master_logical_ports which cannot be interchanged.

10.4.50.2 Used by

permutable_structure

10.4.51 non_permutable_master_logical_port_set

```
  ENTITY non_permutable_master_logical_port_set
    SUBTYPE OF (non_permutable_relationship);
    members : OPTIONAL SET [1:?] OF master_logical_port;      --
reference
  DERIVE
    SELF\non_permutable_relationship.size : INTEGER
      := SIZEOF(NVL (members, []));
  END_ENTITY;
```

10.4.51.1 Description

A non_permutable_master_logical_port_set describes a relationship in which a set of master_logical_ports cannot be interchanged.

10.4.51.2 Used by

master_logical_port

10.4.52 non_permutable_structure

```
  ENTITY non_permutable_structure
    SUBTYPE OF (non_permutable_relationship);
    permutable_members : OPTIONAL SET [1:?] OF
permutable_relationship;
  DERIVE
    SELF\non_permutable_relationship.size : INTEGER
      := non_permutable_structure_size(NVL (permutable_members, []));
  END_ENTITY;
```

Withdrawn
IECNORM.COM : Click to view the full PDF of IEC 62016:2003

10.4.52.1 Description

A non_permutable_structure indicates that its member structures may not be exchanged.

10.4.52.2 Used by

permutable_relationship

10.4.53 non_permutable_structure_size

```
  FUNCTION non_permutable_structure_size
    (permutable_members : SET OF permutable_relationship) : INTEGER;
    LOCAL
      i         : INTEGER;
      temp_size : INTEGER := 0;
    END_LOCAL;

    REPEAT i := 1 TO HIINDEX(permutable_members);
      temp_size := temp_size + permutable_members[i].size;
    END_REPEAT;
    RETURN(temp_size);
  END_FUNCTION;
```

10.4.53.1 Description

The non_permutable_structure_size function returns the size of a non_permutable_structure. The size is the sum of all its member structure sizes.

10.4.53.2 Used by

non_permutable_structure

10.4.54 joined_master_logical_port_set

```
  ENTITY joined_master_logical_port_set;
    member_ports : OPTIONAL SET [1:?] OF master_logical_port;   --
reference
  INVERSE
    containing_cluster_interface
      : SET [0:1] OF cluster_interface FOR
joined_master_logical_port_sets;
    containing_must_join_port_set
      : SET [0:1] OF must_join_port_set FOR nested_joined_sets;
    containing_weak_joined_port_set
      : SET [0:1] OF weak_joined_port_set FOR nested_joined_sets;
  WHERE
    containment_constraint :
      (* Each "joined_master_logical_port_set" is defined in only one
place *)
      SIZEOF(containing_cluster_interface) +
      SIZEOF(containing_must_join_port_set) +
      SIZEOF(containing_weak_joined_port_set) = 1;
  END_ENTITY;
```

10.4.54.1 Description

A joined_master_logical_port_set specifies that certain master_logical_ports are shorted together. Since the master_logical_ports are shorted internally, a connection to one of them implies that a connection has been made to all of the other master_logical_ports. Each joined_master_logical_port_set is defined in only one place.

Withdrawn
IECNORM.COM : Click to view the full PDF of IEC 62016:2003

10.4.54.2 Used by

cluster_interface master_logical_port must_join_port_set weak_joined_port_set

10.4.55 weak_joined_port_set

```
  ENTITY weak_joined_port_set;
    member_ports                                                     --
reference
                       : OPTIONAL SET [1:?] OF master_logical_port;
    nested_joined_sets : OPTIONAL SET [1:?] OF
joined_master_logical_port_set;
  INVERSE
    containing_cluster_interface
      : SET [0:1] OF cluster_interface FOR weak_joined_port_sets;
    containing_must_join_port_set
      : SET [0:1] OF must_join_port_set FOR nested_weak_joined_sets;
  WHERE
    containment_constraint :
      (* Each "weak_joined_port_set" is defined in only one place *)
      SIZEOF(containing_cluster_interface) +
      SIZEOF(containing_must_join_port_set) = 1;
  END_ENTITY;
```

10.4.55.1 Description

A weak_joined_port_set specifies that certain master_logical_ports are shorted internally so that an external connection can be made to any one of them, but they cannot be used as feedthroughs. Each weak_joined_port_set is defined in only one place.

10.4.55.2 Used by

cluster_interface joined_master_logical_port_set master_logical_port must_join_port_set

10.4.56 must_join_port_set

```
  ENTITY must_join_port_set;
    member_ports                                                      --
reference
                            : OPTIONAL SET [1:?] OF
master_logical_port;
    nested_joined_sets      : OPTIONAL SET [1:?] OF
joined_master_logical_port_set;
    nested_weak_joined_sets : OPTIONAL SET [1:?] OF
weak_joined_port_set;
  INVERSE
    containing_cluster_interface : cluster_interface FOR
must_join_port_sets;
  END_ENTITY;
```

10.4.56.1 Description

A must_join_port_set specifies that certain master_logical_ports must be connected externally for correct operation.

10.4.56.2 Used by

cluster_interface joined_master_logical_port_set master_logical_port weak_joined_port_set

Withdrawn
IECNORM.COM : Click to view the full PDF of IEC 62016:2003

10.4.57 cell_representation

```
  ENTITY cell_representation
    ABSTRACT SUPERTYPE OF (ONEOF(internal_cell_representation,
                                 external_cell_representation));
    document             : OPTIONAL SET [1:?] OF documentation;
    name                 : OPTIONAL name_information;
    properties           : OPTIONAL SET [1:?] OF property;
    status_of_copyright  : OPTIONAL SET [1:?] OF copyright;
    status_of_written    : OPTIONAL SET [1:?] OF written;
  DERIVE
    (* The interface is inherited from the containing cluster. *)
    interface : cluster_interface := containing_cluster.interface; --
reference
  INVERSE
    containing_cluster : cluster FOR cell_representations;
   WHERE
    unique_status_of_copyright :
      (* A "cell_representation" does not contain two identical
copyrights *)
      value_unique(status_of_copyright);

    unique_status_of_written :
      (* A "cell_representation" does not contain two identical
"written"
         information  *)
      value_unique(status_of_written);

    unique_document :
      (* A "cell_representation" does not contain two identical
documents *)
      value_unique(document);
  END_ENTITY;
```

10.4.57.1 Description

A cell_representation specifies a representation, or perspective of a cell. It is classified into internal_cell_representation and external_cell_representation. An internal_cell_representation belongs to one internal_cluster and may include implementation details. An external_cell_representation belongs to one external_cell and does not have implementation information. A cell_representation may be derived from another cell_representation. It may also be a new version of another cell_representation.

10.4.57.2 Used by

cell_representation_set cluster copyright documentation name_information property related_cell_representation written

10.4.58 check_derived_recursion

```
  FUNCTION check_derived_recursion
    (cr  : internal_cell_representation;
     rel : related_cell_representation) : BOOLEAN;
    IF EXISTS(rel) THEN
      IF cr :=: rel.related_representation THEN
        RETURN(FALSE);
      ELSE
        IF 'HIERARCHY_MODEL.EXTERNAL_CELL_REPRESENTATION' IN
                 TYPEOF(rel.related_representation) THEN
          RETURN(TRUE);
        END_IF;
```

Withdrawn
IECNORM.COM : Click to view the full PDF of IEC 62016:2003

```
      IF NOT check_derived_recursion
               (cr, rel.related_representation.derived_from) THEN
        RETURN(FALSE);
      END_IF;
    END_IF;
  END_IF;
  RETURN(TRUE);
END_FUNCTION;
```

10.4.58.1 Description

The check_derived_recursion function checks that an internal_cell_representation does not derive from itself either directly or indirectly.

10.4.58.2 Used by

internal_cell_representation

10.4.59 check_previous_recursion

```
  FUNCTION check_previous_recursion
    (cr  : internal_cell_representation;
     rel : related_cell_representation) : BOOLEAN;
    IF EXISTS(rel) THEN
      IF cr :=: rel.related_representation THEN
        RETURN(FALSE);
      ELSE
        IF 'HIERARCHY_MODEL.EXTERNAL_CELL_REPRESENTATION' IN
                 TYPEOF(rel.related_representation) THEN
          RETURN(TRUE);
        END_IF;
        IF NOT check_previous_recursion
                 (cr, rel.related_representation.previous_version)
THEN
          RETURN(FALSE);
        END_IF;
      END_IF;
    END_IF;
    RETURN(TRUE);
  END_FUNCTION;
```

10.4.59.1 Description

The check_previous_recursion function checks that a internal_cell_representation is not a new version of itself either directly or indirectly.

10.4.59.2 Used by

internal_cell_representation

10.4.60 check_valid_status_in_previous_version

```
  FUNCTION check_valid_status_in_previous_version
    (cr : internal_cell_representation) : BOOLEAN;
    LOCAL
      min_time_stamp : time_stamp;
      max_time_stamp : time_stamp;
      pv             : cell_representation;
    END_LOCAL;
    IF NOT EXISTS(cr.status_of_written) OR
```

Withdrawn
IECNORM.COM : Click to view the full PDF of IEC 62016:2003

```
       NOT EXISTS(cr.previous_version) OR
       NOT EXISTS(cr.previous_version.
                  related_representation.status_of_written) THEN
      RETURN(TRUE);
    END_IF;
    min_time_stamp := cr.status_of_written[1].date;
    REPEAT i := 2 TO SIZEOF(cr.status_of_written);
      IF compare_date(min_time_stamp, cr.status_of_written[i].date) >
0 THEN
        min_time_stamp := cr.status_of_written[i].date;
      END_IF;
    END_REPEAT;
    pv := cr.previous_version.related_representation;
    max_time_stamp := pv.status_of_written[1].date;
    REPEAT i := 2 TO SIZEOF(pv.status_of_written);
      IF compare_date(max_time_stamp, pv.status_of_written[i].date) <
0 THEN
        max_time_stamp := pv.status_of_written[i].date;
      END_IF;
    END_REPEAT;
    RETURN(compare_date(min_time_stamp, max_time_stamp) >= 0);
  END_FUNCTION;
```

10.4.60.1 Description

The check_valid_status_in_previous_version function checks that the earliest time_stamp of the internal_cell_representation is later than the latest time_stamp of its previous version.

10.4.60.2 Used by

internal_cell_representation

10.4.61 compare_date

```
  FUNCTION compare_date
    (d1, d2 : time_stamp) : REAL;
    RETURN((d1.recorded_date.year - d2.recorded_date.year) * 1.E10 +
           (d1.recorded_date.month - d2.recorded_date.month) * 1.E8 +
           (d1.recorded_date.day - d2.recorded_date.day) * 1.E6 +
           (d1.recorded_time.hour - d2.recorded_time.hour) * 1.E4 +
           (d1.recorded_time.minute - d2.recorded_time.minute) * 1.E2
+
           (d1.recorded_time.second - d2.recorded_time.second));
  END_FUNCTION;
```

10.4.61.1 Description

compare_date compares two time stamps “d1” and “d2”. It returns a negative number if d2 > d1, 0 if d1 = d2 and a positive number if d2 < d1.

10.4.62 internal_cell_representation

```
  ENTITY internal_cell_representation
    ABSTRACT SUPERTYPE OF (ONEOF(internal_connectivity_view))
    SUBTYPE OF (cell_representation);
    derived_from        : OPTIONAL related_cell_representation;
    previous_version    : OPTIONAL related_cell_representation;
  DERIVE
    (* The interface is inherited from the containing cluster. *)
```

Withdrawn
IECNORM.COM: Click to view the full PDF of IEC 62016:2003

```
    internal_cluster_interface : cluster_interface             --
reference
      := containing_internal_cluster.interface;
  INVERSE
    containing_internal_cluster : internal_cluster FOR
cell_representations;
  WHERE
    non_recursive_derive_from :
      (* An internal_cell_representation does not derive from itself
         either directly or indirectly. *)
      check_derived_recursion(SELF, derived_from);

    non_recursive_previous_version :
      (* An internal_cell_representation is not a new version of
itself
         either directly or indirectly. *)
      check_previous_recursion(SELF, previous_version);

    valid_derive_from :
      (* The internal_cell_representation and its derived
cell_representation
         are defined in the same cell. *)
      containing_cluster.containing_cell :=:

derived_from.related_representation.containing_cluster.containing_cell;

    valid_previous_version :
      (* The internal_cell_representation and its previous version are
         defined in the same cell and they are of the same type. *)
      (containing_cluster.containing_cell :=:
       previous_version.
       related_representation.
       containing_cluster.
       containing_cell)
      AND (TYPEOF(SELF) = TYPEOF(previous_version));

    valid_status_in_previous_version :
      (* The earliest timestamp of the internal_cell_representation is
         later than the latest timestamp of its previous version. *)
      check_valid_status_in_previous_version(SELF);
  END_ENTITY;
```

Withdrawn
IECNORM.COM: Click to view the full PDF of IEC 62016:2003

10.4.62.1 Description

An internal_cell_representation is classified into an internal_connectivity_view. It is a cell_representation in an internal_cell. An internal_cell_representation may contain implementation information. An internal_cell_representation does not derive from, and is not a new version of, itself either directly or indirectly. The internal_cell_representation and its derived cell_representation are defined in the same cell. The internal_cell_representation and its previous version are defined in the same cell and they are of the same type.

10.4.62.2 Used by

expandable_internal_cluster_configuration expandable_internal_occurrence_annotate expandable_internal_occurrence_hierarchy_annotate internal_cluster related_cell_ representation

10.4.63 external_cell_representation

```
  ENTITY external_cell_representation
    ABSTRACT SUPERTYPE OF (ONEOF(external_connectivity_view))
    SUBTYPE OF (cell_representation);
  DERIVE
    (* The interface is inherited from the containing cluster. *)
    external_cluster_interface : cluster_interface          --
reference
      := containing_external_cluster.interface;
  INVERSE
    containing_external_cluster : external_cluster FOR
cell_representations;
  END_ENTITY;
```

10.4.63.1 Description

An external_cell_representation is classified into an external_connectivity_view. It is a cell_representation in an external_cell. An external_cell_representation does not contain implementation information.

10.4.63.2 Used by

expandable_external_cluster_configuration expandable_external_occurrence_annotate expandable_external_occurrence_hierarchy_annotate external_cluster

10.4.64 related_cell_representation

```
  ENTITY related_cell_representation;
    reason                 : OPTIONAL string_token;
    related_representation : cell_representation;           --
reference
  INVERSE
    containing_internal_cell_representation_from
      : SET [0:1] OF internal_cell_representation FOR derived_from;
    containing_internal_cell_representation_previous
      : SET [0:1] OF internal_cell_representation FOR
previous_version;
  WHERE
    containment_constraint :
      (* Each "related_cell_representation" is defined in
         only one place *)
      SIZEOF(containing_internal_cell_representation_from) +
      SIZEOF(containing_internal_cell_representation_previous) = 1;
  END_ENTITY;
```

10.4.64.1 Description

A related_cell_representation specifies a related cell representation. This is used in defining the relationship between two cell representations where one is a new version of another, or where one is derived from another. A reason can be specified for the relationship by a string token.

10.4.64.2 Used by

internal_cell_representation

Withdrawn
IECNORM.COM : Click to view the full PDF of IEC 62016:2003

10.4.65 instance

```
  ENTITY instance
    ABSTRACT SUPERTYPE OF (ONEOF(external_cluster_instance,
                                 internal_cluster_instance));
    instantiated_cluster          : cluster;                 --
reference
    port_attributes               : OPTIONAL SET [1:?] OF
                                    instance_port_attributes;
    port_bundle_attributes        : OPTIONAL SET [1:?] OF
                                    instance_port_bundle_attributes;
    name                          : OPTIONAL name_information;
    overriding_cell_properties    : OPTIONAL SET [1:?] OF
property_override;
    overriding_cluster_properties : OPTIONAL SET [1:?] OF
property_override;
    overriding_designator         : OPTIONAL string_token;
    properties                    : OPTIONAL SET [1:?] OF property;
    width                         : positive_integer_token;
  INVERSE
    containing_internal_connectivity_view
      : internal_connectivity_view FOR instances;
  WHERE
    valid_instance_ports_attributes :
      (* Each instance_port_attributes references a
master_logical_port
         defined in the interface of the instantiated_cluster. *)
      not_exists(QUERY(port <* port_attributes |
           (port.referenced_master_port.containing_interface :<>:
                 instantiated_cluster.interface)
           OR (port.containing_instance :<>: SELF)));

    valid_instance_port_bundles_attributes :
      (* Each instance_port_bundle_attributes references a
         master_logical_port_bundle defined in the interface of the
         instantiated_cluster. *)
      not_exists(QUERY(portBundle <* port_bundle_attributes |
        (portBundle.referenced_master_port_bundle.containing_interface
         :<>: instantiated_cluster.interface)
        OR (portBundle.containing_instance :<>: SELF)));

    valid_overriding_cell_properties :
      (* The overridden properties are defined in the containing cell
of the
         instantiated_cluster. *)
      not_exists(QUERY(cellPropertyOverride <*
overriding_cell_properties |
        NOT (instantiated_cluster.containing_cell IN

cellPropertyOverride.overridden_property.containing_cell)));

    valid_overriding_cluster_properties :
      (* The overridden properties are defined in the
instantiated_cluster. *)
      not_exists(QUERY(clusterPropertyOverride <*
                       overriding_cluster_properties |
        NOT (instantiated_cluster IN

clusterPropertyOverride.overridden_property.containing_cluster)));
  END_ENTITY;
```

IECNORM.COM : Click to view the full PDF of IEC 62016:2003 Withdrawn

10.4.65.1 Description

An instance can either be an external_cluster_instance or an internal_cluster_instance. It allows a cluster to be referenced within internal_connectivity_view to create the design instance hierarchy. If the width of an instance is greater than 1, an implicit set of instances is created, in this case, all attributes of the instance apply equally to each member of the set. However, the name_information is only applied to the instance but not its members. Each instance_port_attributes and instance_port_bundle_attributes reference a master_logical_port and master_logical_port_bundle repectively and are defined in the interface of the instantiated cluster.

10.4.65.2 Used by

connectivity_instance_implementation instance_configuration instance_member_logical_port instance_member_structure_port instance_port_attributes instance_port_bundle_attributes instance_structure_port instance_structure_port_bundle internal_connectivity_view name_information occurrence_annotate property property_override unconfigured_instance

10.4.66 find_instance_port_attributes

```
  FUNCTION find_instance_port_attributes
    (port : master_logical_port;
     inst : instance) : instance_port_attributes;

    REPEAT i := 1 TO SIZEOF(inst.port_attributes);
      IF port :=: inst.port_attributes[i].
                  referenced_master_port
      THEN
        RETURN(inst.port_attributes[i]);
      END_IF;
    END_REPEAT;
    RETURN(?);
  END_FUNCTION;
```

10.4.66.1 Description

The find_instance_port_attributes function returns the instance_port_attributes given the referenced master_logical_port and the referenced instance.

10.4.66.2 Used by

instance_port_annotate

10.4.67 external_cluster_instance

```
  ENTITY external_cluster_instance
    SUBTYPE OF (instance);
    SELF\instance.instantiated_cluster : external_cluster;        --
reference
  END_ENTITY;
```

10.4.67.1 Description:

An external_cluster_instance describes an instance of an external_cluster.

10.4.68 internal_cluster_instance

```
  ENTITY internal_cluster_instance
    SUBTYPE OF (instance);
```

Withdrawn
IECNORM.COM : Click to view the full PDF of IEC 62016:2003

```
    SELF\instance.instantiated_cluster : internal_cluster;        --
reference
  END_ENTITY;
```

10.4.68.1 Description

An internal_cluster_instance describes an instance of an internal_cluster.

10.4.69 instance_port_attributes

```
  ENTITY instance_port_attributes
    ABSTRACT SUPERTYPE OF (ONEOF(
      input_instance_port_attributes,
      output_instance_port_attributes,
      bidirectional_instance_port_attributes,
      unspecified_direction_instance_port_attributes));
    referenced_master_port : master_logical_port;              --
reference
    new_properties          : OPTIONAL SET [1:?] OF property;
    overriding_ac_load      : OPTIONAL capacitance_value;
    overriding_designator   : OPTIONAL string_token;
    overriding_properties   : OPTIONAL SET [1:?] OF property_override;
    unused_externally       : boolean_token;
  INVERSE
    containing_instance : instance FOR port_attributes;
  UNIQUE
    uniqueness_constraint:
      (* There is only one instance_port_attributes for a particular
         master_logical_port on an instance *)
    referenced_master_port, containing_instance;
  WHERE
    valid_port_and_instance :
      (* The referenced master_logical_port is defined in the
interface of
         the instantiated cluster of the referenced instance. *)
      referenced_master_port IN

containing_instance.instantiated_cluster.interface.logical_ports;

     valid_overriding_properties:
      (* The overridden properties are defined in the referenced
         master_logical_port *)
      not_exists(QUERY(prop <* overriding_properties |
        NOT (prop.overridden_property IN
             referenced_master_port.properties)));
  END_ENTITY;
```

IECNORM.COM : Click to view the full PDF of IEC 62016:2003

10.4.69.1 Description

An instance_port_attributes is either an input_instance_port_attributes, output_instance_port_attributes bidirectional_instance_port_attributes or unspecified_direction_instance_port_attributes. It is used to attach or modify the properties or attributes of a port within an instance. The referenced master_logical_port must be defined in the interface of the cluster which the containing instance instantiates. The unused externally attribute is a boolean flag indicating whether or not the port is connected to any signal in the instantiating view. The overridden properties are defined in the referenced master_logical_port.

10.4.69.2 Used by

capacitance_value instance instance_port_annotate property property_override

10.4.70 input_instance_port_attributes

```
  ENTITY input_instance_port_attributes
    SUBTYPE OF (instance_port_attributes);
    SELF\instance_port_attributes.referenced_master_port       --
reference
      : input_master_logical_port;
    overriding_dc_fanout_load : OPTIONAL load_value;
    overriding_dc_max_fanin   : OPTIONAL load_value;
  END_ENTITY;
```

10.4.70.1 Description

An input_instance_port_attributes is used as the point of attachment for new properties and attributes and as the point at which properties and attributes inherited from the input_master_logical_port may be modified within an instance.

10.4.71 output_instance_port_attributes

```
  ENTITY output_instance_port_attributes
    SUBTYPE OF (instance_port_attributes);
    SELF\instance_port_attributes.referenced_master_port        --
reference
      : output_master_logical_port;
    overriding_dc_fanin_load : OPTIONAL load_value;
    overriding_dc_max_fanout : OPTIONAL load_value;
  END_ENTITY;
```

10.4.71.1 Description

An output_instance_port_attributes is used as the point of attachment for new properties and attributes and as the point at which properties and attributes inherited from the output_master_logical_port may be modified within an instance.

10.4.72 bidirectional_instance_port_attributes

```
  ENTITY bidirectional_instance_port_attributes
    SUBTYPE OF (instance_port_attributes);
    SELF\instance_port_attributes.referenced_master_port        --
reference
      : bidirectional_master_logical_port;
    overriding_dc_fanin_load  : OPTIONAL load_value;
    overriding_dc_fanout_load : OPTIONAL load_value;
    overriding_dc_max_fanin   : OPTIONAL load_value;
    overriding_dc_max_fanout  : OPTIONAL load_value;
  END_ENTITY;
```

10.4.72.1 Description

A bidirectional_instance_port_attributes is used as the point of attachment for new properties and attributes and as the point at which properties and attributes inherited from the bidirectional_master_logical_port may be modified within an instance.

Withdrawn
IECNORM.COM: Click to view the full PDF of IEC 62016:2003

10.4.73 unspecified_direction_instance_port_attributes

```
  ENTITY unspecified_direction_instance_port_attributes
    SUBTYPE OF (instance_port_attributes);
    SELF\instance_port_attributes.referenced_master_port        --
reference
      : unspecified_direction_master_logical_port;
  END_ENTITY;
```

10.4.73.1 Description

An unspecified_direction_instance_port_attributes is used as the point of attachment for new properties and attributes and as the point at which properties and attributes inherited from the unspecified_direction_master_logical_port may be modified within an instance.

10.4.74 instance_structure_port

```
  ENTITY instance_structure_port;
    referenced_instance    : instance;                          --
reference
    referenced_master_structure_port : master_structure_port;
-- reference
  DERIVE
    flattened_port_list : LIST [1:?] OF UNIQUE
instance_member_structure_port
      := flatten_instance_structure_port(SELF);
    size                : INTEGER
      := SIZEOF(flattened_port_list);
  INVERSE
    containing_generic_net:
      SET [0:1] OF connectivity_generic_net FOR
joined_instance_structure_ports;
  WHERE
    valid_port_and_instance :
      (* The referenced master_structure_port is defined in the
interface of
         the instantiated cluster of the referenced instance. *)
    referenced_master_structure_port.containing_interface :=:
    referenced_instance.instantiated_cluster.interface;
  END_ENTITY;
```

10.4.74.1 Description

An instance_structure_port is used to reference a master_structure_port on an instance.The referenced master_structure_port is defined in the interface of the instantiated cluster of the referenced instance.

10.4.74.2 Used by

connectivity_generic_net instance_port_annotate

10.4.75 flatten_instance_structure_port

```
  FUNCTION flatten_instance_structure_port
    (instancePort : instance_structure_port)
      : LIST [1:?] OF UNIQUE instance_member_structure_port;
    LOCAL
      it         : integer_token := 0;
      memberSize : INTEGER := instancePort.referenced_instance.width;
```

Withdrawn
IECNORM.COM : Click to view the full PDF of IEC 62016:2003

```
      members    : LIST [0:?] OF UNIQUE instance_member_structure_port
:= [ ];
    END_LOCAL;

    REPEAT i := 1 TO memberSize;
      it := it + 1;
      INSERT (members,
              instance_member_structure_port
(instancePort.referenced_instance,
                                                 it,
                                                 instancePort.

referenced_master_structure_port),
              i - 1);
    END_REPEAT;
    RETURN(members);
  END_FUNCTION;
```

10.4.75.1 Description

The flatten_instance_structure_port function returns the flattened list of instance_member_structure_ports of an instance_structure_port. The size of the list equals the width of the instance.

10.4.75.2 Used by

instance_structure_port

10.4.76 instance_member_logical_port

```
  ENTITY instance_member_logical_port;
    referenced_instance              : instance;            --
reference
    referenced_instance_member_index : integer_token;       --
reference
    referenced_master_port           : master_logical_port; --
reference
  DERIVE
    size  : INTEGER := 1;
  INVERSE
    containing_signal : signal FOR
joined_instance_member_logical_ports;
  WHERE
    valid_referenced_instance_member_index :
      (* The member_index is within the range defined by the width of
the
         referenced_instance. *)
      {0 <= referenced_instance_member_index <
referenced_instance.width};

    valid_port_and_instance :
      (* The referenced master_logical_port is defined in the
interface of the
         instantiated cluster of the referenced instance. *)
      referenced_master_port.containing_interface :=:
      referenced_instance.instantiated_cluster.interface;
  END_ENTITY;
```

Withdrawn
IECNORM.COM: Click to view the full PDF of IEC 62016:2003

10.4.76.1 Description

An instance_member_logical_port describes a port on an instance member. A member of an instance is accessed by giving the member index value. The size of the instance_ member_logical_portis always one. The member index is within the range defined by the width of the referenced instance. The referenced master_logical_port is defined in the interface of the instantiated cluster of the referenced instance.

10.4.76.2 Used by

signal

10.4.77 instance_member_structure_port

```
  ENTITY instance_member_structure_port;
    referenced_instance              : instance;              --
reference
    referenced_instance_member_index : integer_token;         --
reference
    referenced_master_port           : master_structure_port;   --
reference
  DERIVE
    size  : INTEGER := 1;
  INVERSE
    containing_generic_net:
      SET [0:1] OF connectivity_generic_net FOR
joined_instance_member_structure_ports;
  WHERE
    valid_referenced_instance_member_index :
      (* The member_index is within the range defined by the width of
the
         referenced_instance. *)
      {0 <= referenced_instance_member_index <
referenced_instance.width};

    valid_port_and_instance :
      (* The referenced master_structure_port is defined in the
interface of
         the instantiated cluster of the referenced instance. *)
      referenced_master_port.containing_interface :=:
      referenced_instance.instantiated_cluster.interface;
  END_ENTITY;
```

IECNORM.COM : Click to view the full PDF of IEC 62016:2003
Withdrawn

10.4.77.1 Description

An instance_member_structure_port describes a master_structure_port on an instance member. A member of an instance is accessed by giving the member index value. The member_index is within the range defined by the width of the referenced instance. the size of the instance_member_structure_port is always one. The referenced master_structure_port is defined in the interface of the instantiated cluster of the referenced instance.

10.4.77.2 Used by

connectivity_generic_net instance_structure_port

10.4.78 instance_structure_port_bundle

```
  ENTITY instance_structure_port_bundle;
    referenced_instance           : instance;                    --
reference
```