Proposed recommended practice for the representation of schematic symbols in STEP AP 210 (ISO 10303-210)

June 30, 2010

Revised April 5,2011

Introduction

The purpose of this document is to propose a recommended practice for the representation of schematic symbols and the integration of schematic symbols with functional and physical representations of packaged electronic component models in ISO 10303-210 (STEP AP 210). While the representation of AP 210 packages and packaged parts has been previously validated through existing implementations, the representation of component functional models and associated schematic symbols has been less thoroughly exercised.

AP 210 has dedicated concepts for the representation of a functional element, as well as the association of a functional element with a physical realization through a packaged part and associated package model. The intent in the development of this recommended practice was to leverage existing draughting capabilities within the STEP standards for the presentation of a schematic symbol in AP 210. The schematic symbol is a presentation of the functional model for human visualization, interpretation, and manipulation - the schematic symbol should be a draughting representation of a functional element.

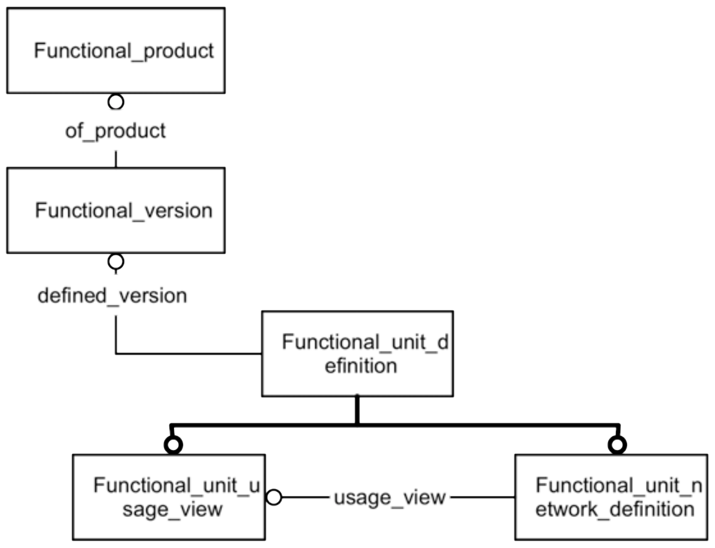
Functional_product
of_product
Functional_version
defined_version
Functional_unit_d efinition
Functional_unit_u sage_view
usage_view
Functional_unit_n etwork_definition

Figure 1 An AP 210 functional_element_definition is either an interface or a network

In this document, “functional element” represents the generic domain concept instead of “functional unit” in order to avoid confusion between the domain concept and the AP 210 information model description. The Application Object (AO) Functional_unit_definition represents the domain concept “functional element” in the AP 210 Application Reference Model (ARM). Functional_unit_usage_view represents the domain concept “interface”. Functional_network_definition represents the domain concept “netlist”.

In order to ensure that the schematic symbol can be unambiguously related to the interface, several new concepts have been proposed as extensions to the existing AP 210 standard. These proposed extensions have been kept to a minimum, in order to ensure the greatest degree of interoperability with existing implementations of STEP draughting.

In the remainder of this document, the key AOs and corresponding Modular Integrated Model (MIM) entities and relationships to be used in the representation of a functional model, schematic symbol, and packaged part are discussed and illustrated.

Notation

AOs will be denoted with a leading uppercase letter in Arial font (i.e. Functional_unit) while a MIM entity will be displayed in all lowercase notation (i.e. component_functional_unit).

Functional Model

Figure 1 shows the top-level AOs relevant in the definition of a functional element. Because a functional element may be configuration managed, the representation in AP 210 for a functional element includes Functional_product, Functional_version and Functional_unit_definition. A Functional_product is classified in STEP as an information product because it is not physically realized. In the context of electronic design, a Functional_product would correspond to the identification information related to any functional element desired by the organization. The Functional_product may have multiple versions (Functional_version). Approval, release, ownership, etc., of functional elements are supported by AP 210 but are not in the scope of this recommended practice.

The interface to a functional element is the Functional_unit_usage_view. The Functional_unit_usage_view will as a minum contain the functional terminals that define the interface structure. Figure 2 illustrates the key AOs needed to represent a simple interface (Functional_unit_usage_view) for a nand gate consisting of three terminals.

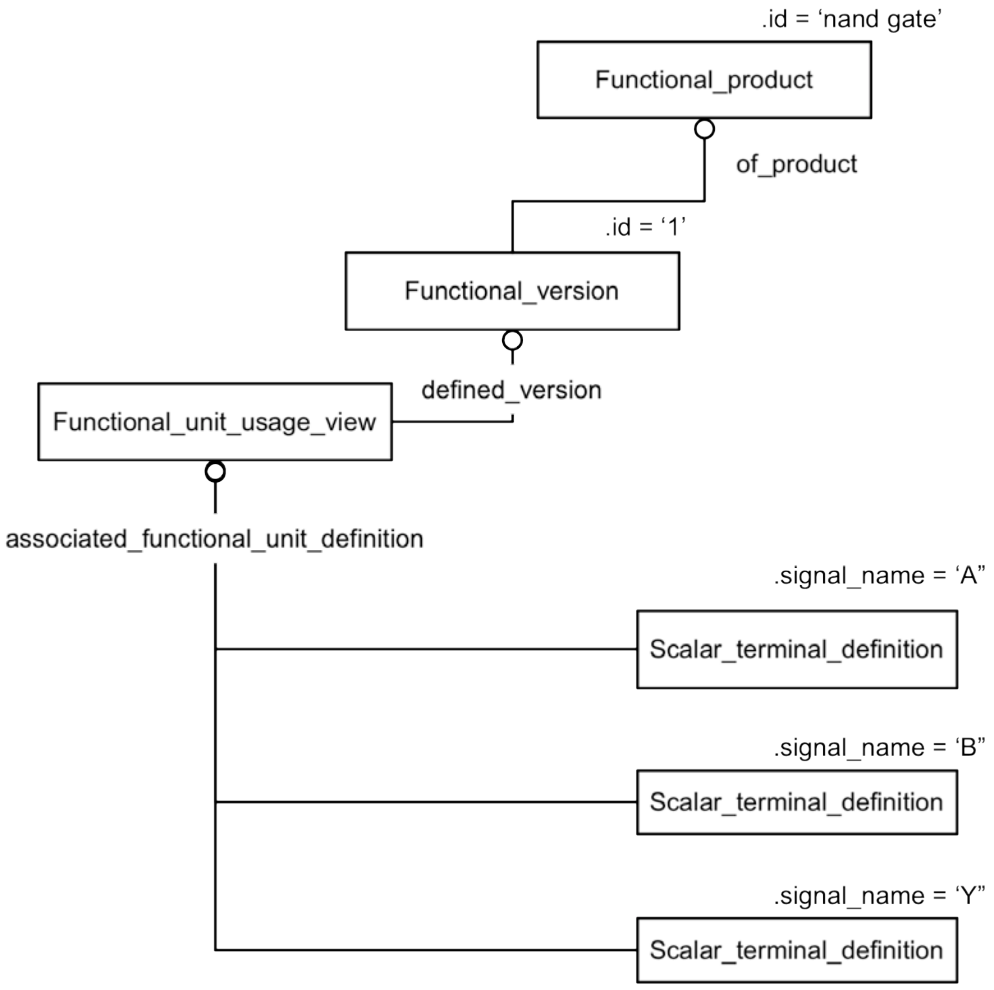
.id = ‘nand gate’
Functional_product
of_product
.id = ‘1’
Functional_version
defined_version
Functional_unit_usage_view
associated_functional_unit_definition
.signal_name = ‘A”
Scalar_terminal_definition
.signal_name = ‘B”
Scalar_terminal_definition
.signal_name = ‘Y”
Scalar_terminal_definition

Figure 2 A minimal ARM interface definition of a two input nand

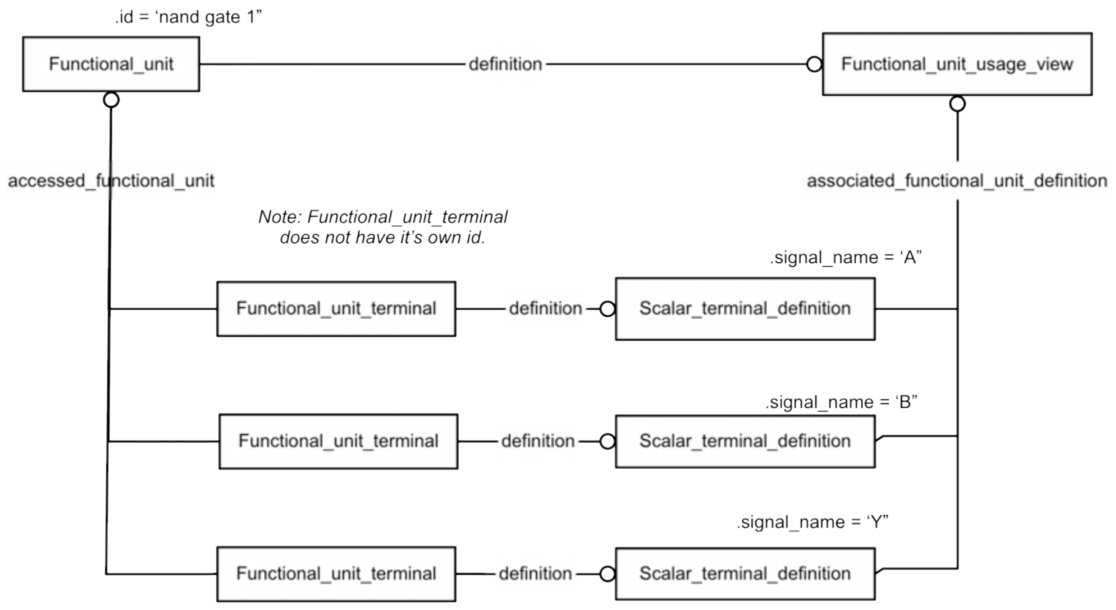

Figure 3 An instance of a functional element where the definition is an interface.

The network definition of a functional element is composed of instances of other functional elements. Each instance of a functional element will be represented by an instance of Functional_unit that will have its own definition (interface only or further decomposition and interface) and its own set of terminals (Functional_unit_terminal).

Functional_unit_terminal instances are uniquely identified by the combination of the Functional_unit.id and the Scalar_terminal_definition.signal_name specified by the Functional_unit.definition and therefore introducing a separate slot for a name of a Functional_unit_terminal would be redundant and AP 210 does not provide that separate slot. Figure 3 illustrates the use case where an instance of a functional element only references an interface as its definition.

In the event that further decomposition of the functional element is to be provided, the Functional_unit.definition would reference that decomposition (an instance of Functional_unit_network_definition). The interface would be indirectly referenced (by the Functional_unit_network_definition.usage_view) instead of directly by the Functional_unit.definition. The terminal references would not change.

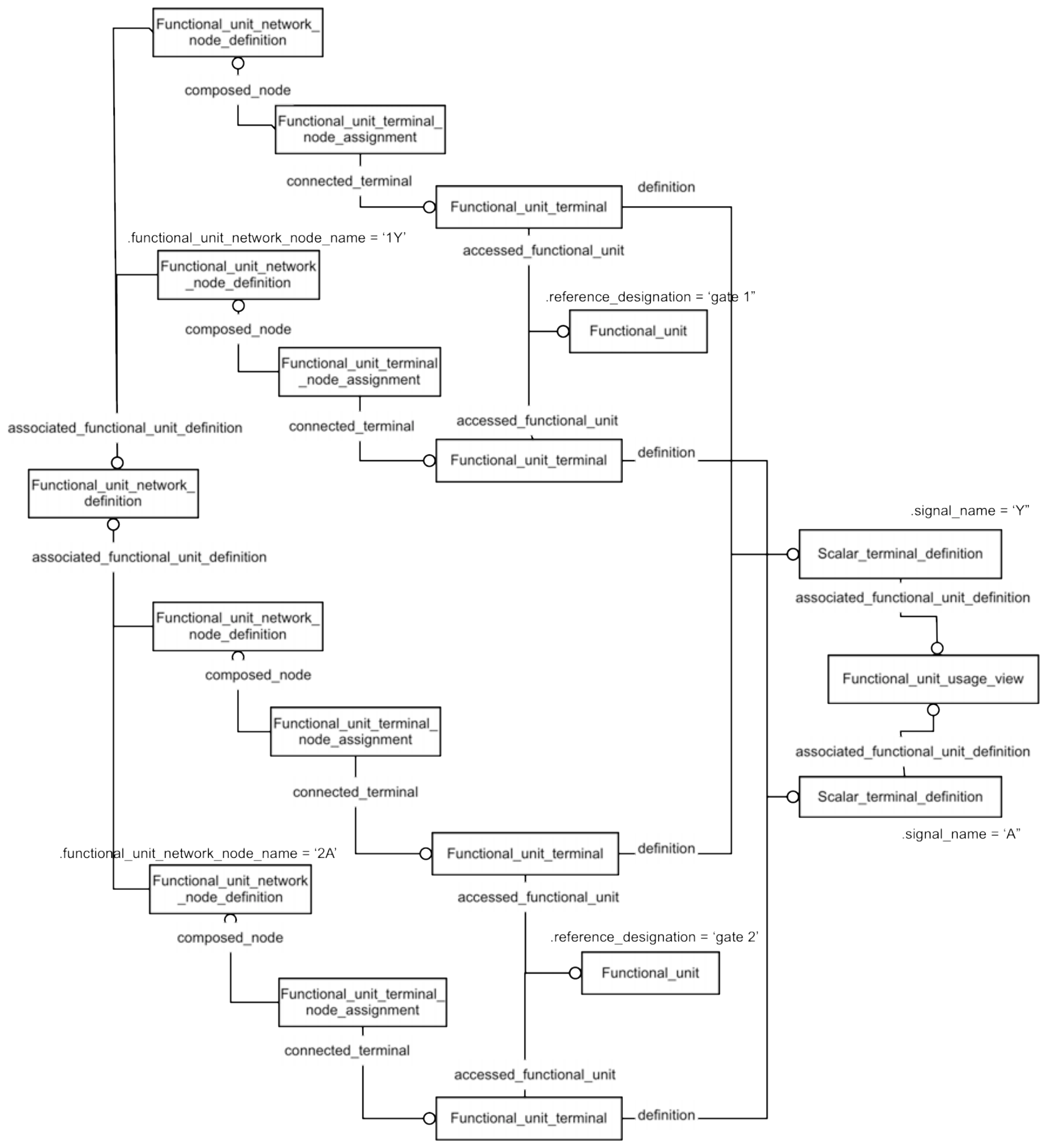
Functional_unit_network_node_definition
composed_node
Functional_unit_terminal_node_assignment
connected_terminal
Functional_unit_terminal
definition
.functional_unit_network_node_name = '1Y'
Functional_unit_network_node_definition
accessed_functional_unit
.reference_designation = 'gate 1"
Functional_unit
composed_node
Functional_unit_terminal_node_assignment
associated_functional_unit_definition
connected_terminal
accessed_functional_unit
Functional_unit_terminal
definition
Functional_unit_network_definition
.signal_name = 'Y"
Scalar_terminal_definition
associated_functional_unit_definition
associated_functional_unit_definition
Functional_unit_network_node_definition
Functional_unit_usage_view
composed_node
Functional_unit_terminal_node_assignment
associated_functional_unit_definition
Scalar_terminal_definition
connected_terminal
.signal_name = 'A"
.functional_unit_network_node_name = '2A'
Functional_unit_terminal
definition
Functional_unit_network_node_definition
accessed_functional_unit
composed_node
.reference_designation = 'gate 2'
Functional_unit
Functional_unit_terminal_node_assignment
connected_terminal
accessed_functional_unit
Functional_unit_terminal
definition

Figure 4. A Functional_unit_network_definition composed of two instances of a common functional element.

Figure 4 illustrates the AO instances needed to represent a network (Functional_unit_network_definition) composed of two instances of a common Functional_unit_definition. Each node in the network definition will be associated with one or more individual Functional_unit_terminal. Each association is accomplished through a Functional_unit_terminal_node_assignment. In general, there are more than one Functional_unit_terminal associated with a node. In the particular example of Figure 4, each node is associated with only one terminal because each node will later be used to define a terminal for the interface of the network. This pattern will continue up through the functional hierarchy to the top level interface, with the result being the availability of access to all primitive functional elements included in a component by the mapping of the top level functional interface to the pins of the Packaged_part.

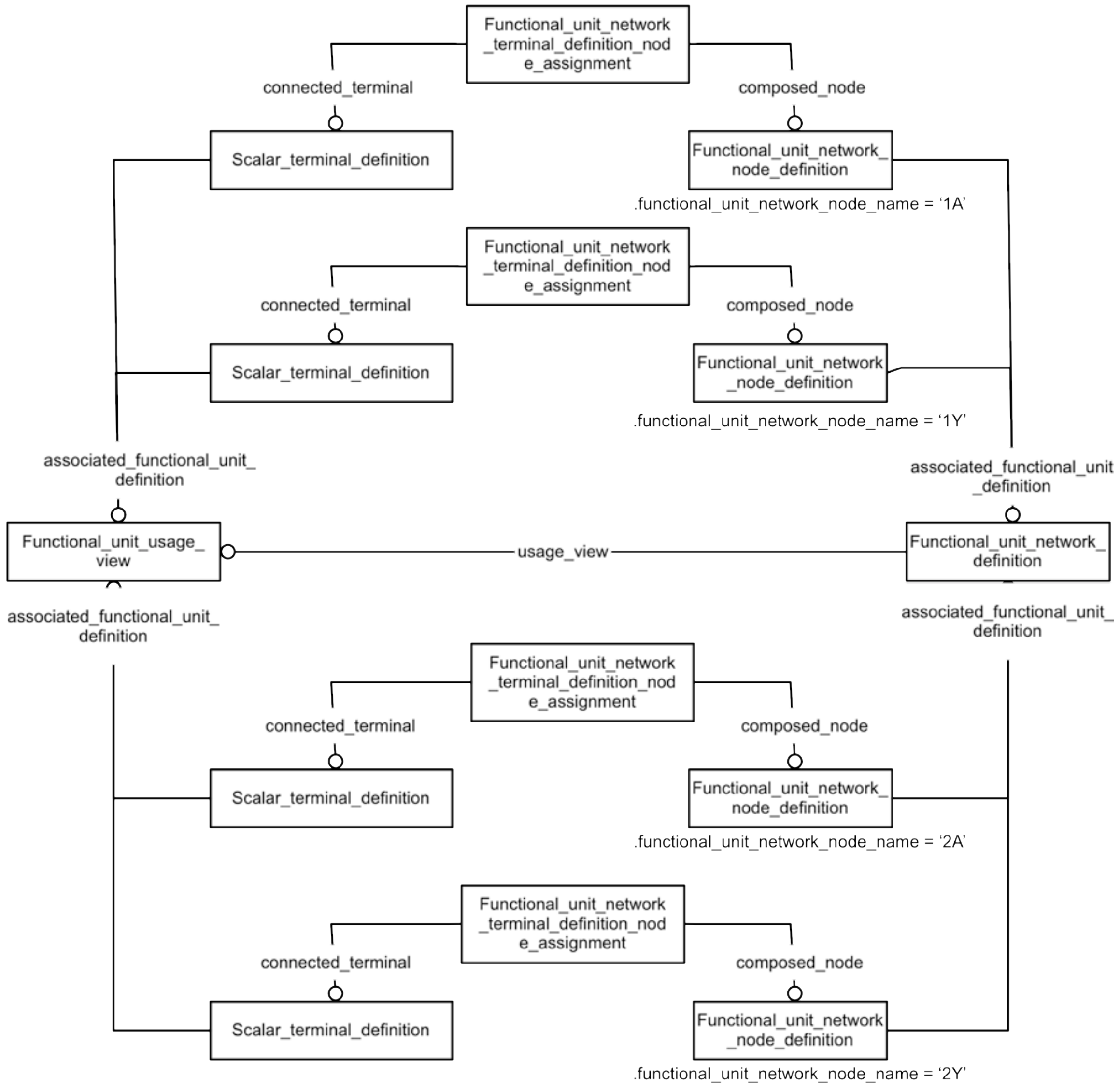
Functional_unit_network_terminal_definition_node_assignment
connected_terminal
composed_node
Scalar_terminal_definition
Functional_unit_network_node_definition
.functional_unit_network_node_name = '1A'
Functional_unit_network_terminal_definition_node_assignment
connected_terminal
composed_node
Scalar_terminal_definition
Functional_unit_network_node_definition
.functional_unit_network_node_name = '1Y'
associated_functional_unit_definition
associated_functional_unit_definition
Functional_unit_usage_view
usage_view
Functional_unit_network_definition
associated_functional_unit_definition
associated_functional_unit_definition
Functional_unit_network_terminal_definition_node_assignment
connected_terminal
composed_node
Scalar_terminal_definition
Functional_unit_network_node_definition
.functional_unit_network_node_name = '2A'
Functional_unit_network_terminal_definition_node_assignment
connected_terminal
composed_node
Scalar_terminal_definition
Functional_unit_network_node_definition
.functional_unit_network_node_name = '2Y'

Figure 5. A Functional_unit_usage_view for a Functional_unit_network_definition.

The downward decomposition terminates when all Functional_units are defined by an interface. The AO instances that would be used in the creation of a Functional_unit_usage_view for the network definition of Figure 4 are illustrated in Figure 5. A Functional_unit_network_terminal_definition_node_assignment is used to provide a definition for a terminal in the interface.

The MIM mappings of the ARM concepts and relationships related to the functional element are illustrated graphically in Figures 6 through 8. In the figures, key AOs are represented in blue overlapping with the corresponding mapped MIM entity. Figure 6 illustrates the mappings for Functional_product, Functional_unit_usage_view and Functional_unit_network_definition. Functional_unit, Functional_unit_usage_view and Functional_unit_usage_view map to the entity functional_unit; Functional_unit is

ABSTRACT and not instantiable. Consequently the product_definition_context specified by product_definition_context_association.frame_of_reference must be used to discriminate between Functional_unit_usage_view and Functional_unit_network_definition. The pre-processor is required to populate the product_definition_context_role.name value of ‘part definition type’ for the AOs Functional_unit_usage_view and Functional_unit_network_definition. The post-processor shall ignore the product_definition_context_role.name string. The product corresponding to a Functional_product should be associated with both ‘functional’ and ‘information’ product_related_product_categories. Figure 7 illustrates the mappings for key relationships between Functional_unit_network_definition, Functional_unit_network_node_definition, Functional_unit_terminal, Functional_unit, and Scalar_terminal_definition. In Figure 7, the name attribute of component_functional_terminal is provided. There is no mapping that specifies this value but a concatenation of the Scalar_teminal_definition.name and of the Functional_unit. reference_designation may prove useful for debugging. In Figure 8, mappings corresponding to the relationship between the Functional_unit_usage_view and Functional_unit_network_definition are illustrated.

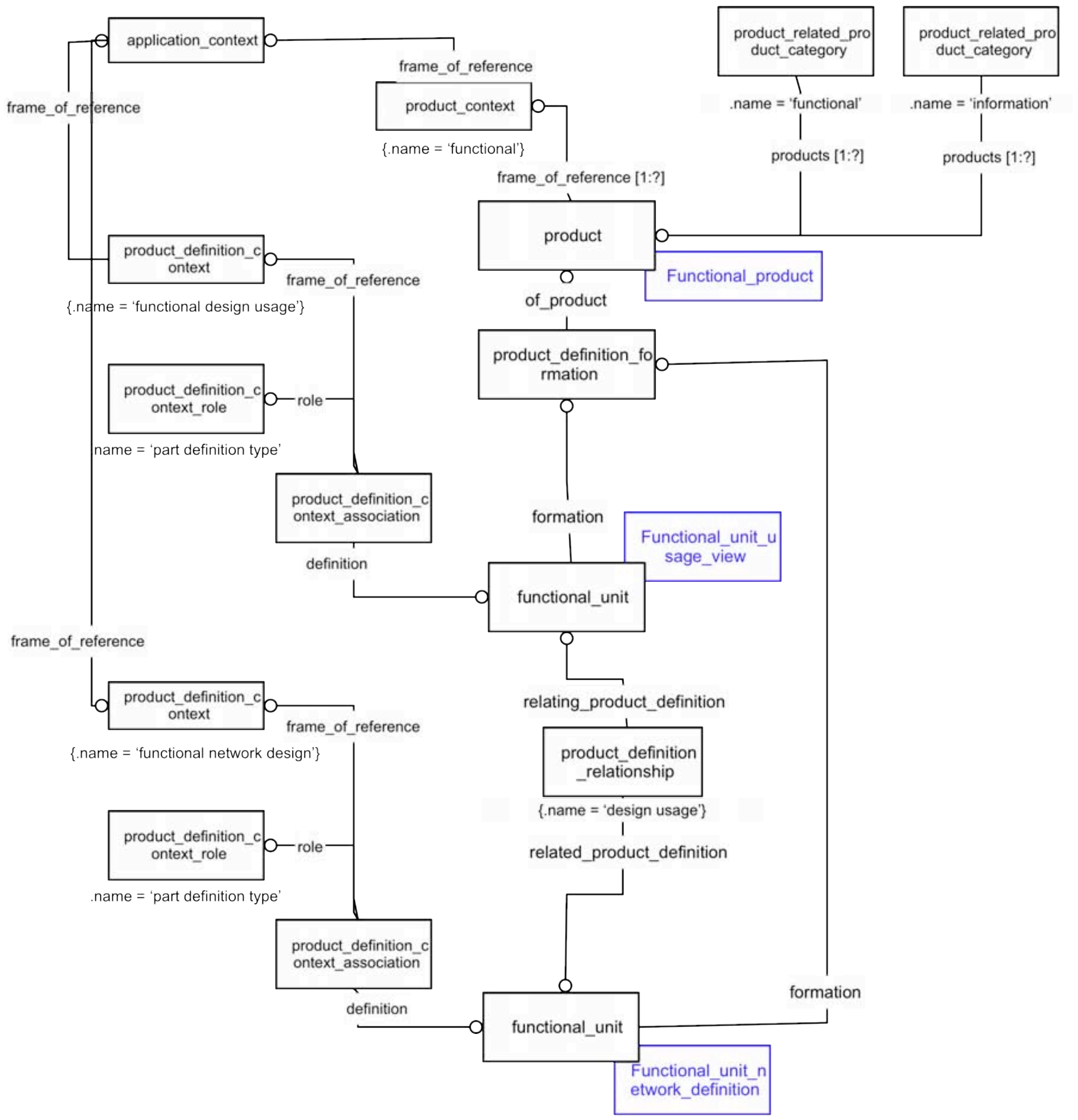
application_context
frame_of_reference
frame_of_reference
product_context
{.name = 'functional'}
product_related_product_category
.name = 'functional'
products [1:?]
product_related_product_category
.name = 'information'
products [1:?]
frame_of_reference [1:?]
product
Functional_product
product_definition_context
{.name = 'functional design usage'}
frame_of_reference
of_product
product_definition_formation
product_definition_context_role
role
name = 'part definition type'
product_definition_context_association
formation
Functional_unit_usage_view
definition
functional_unit
frame_of_reference
relating_product_definition
product_definition_context
{.name = 'functional network design'}
frame_of_reference
product_definition_relationship
{.name = 'design usage'}
product_definition_context_role
role
.name = 'part definition type'
related_product_definition
product_definition_context_association
definition
functional_unit
formation
Functional_unit_network_definition

Figure 6 MIM mapping of top-level Functional_product AOs

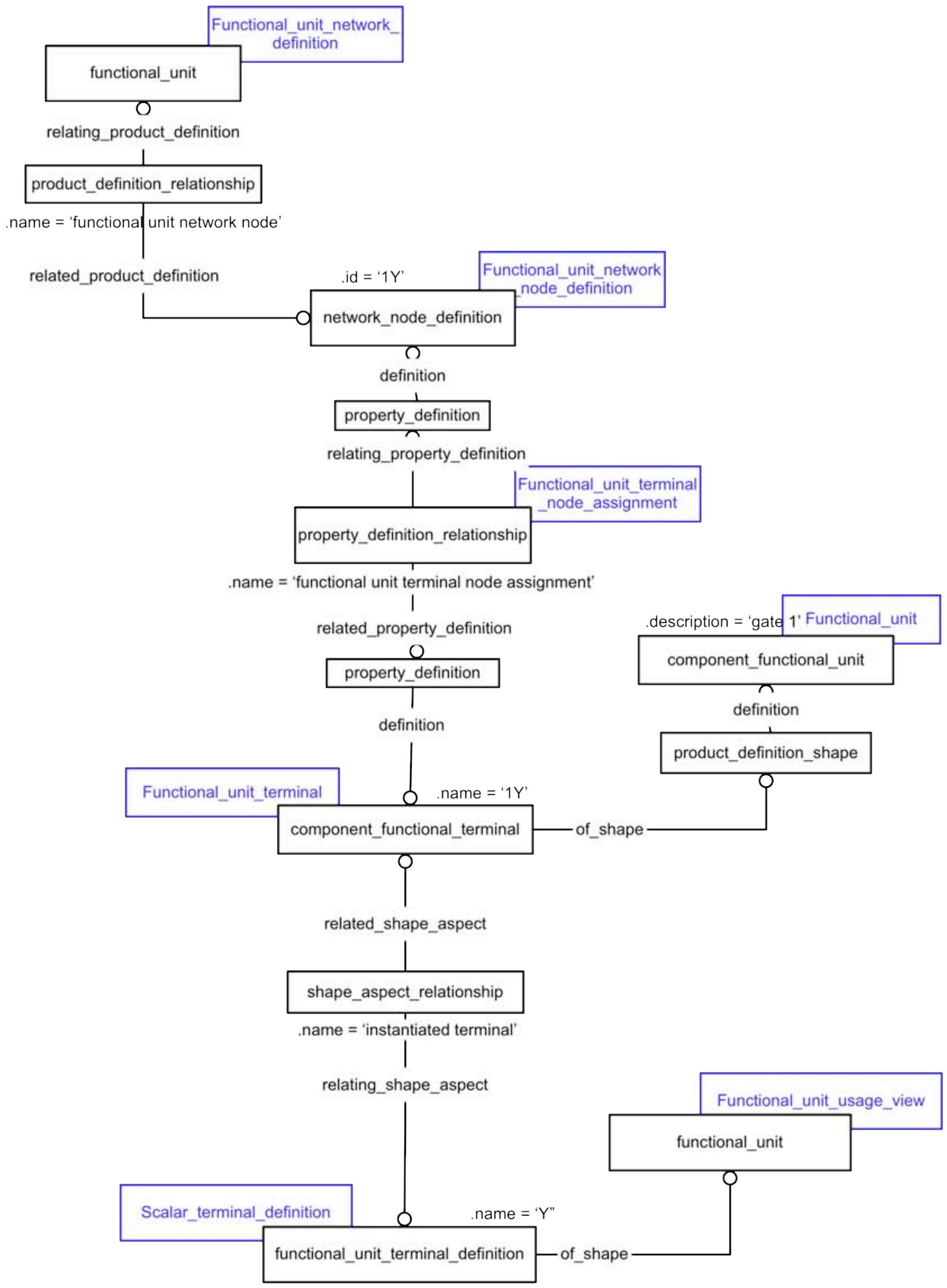
Functional_unit_network_ definition
functional_unit
relating_product_definition
product_definition_relationship
.name = 'functional unit network node'
related_product_definition
.id = '1Y'
Functional_unit_network _node_definition
network_node_definition
definition
property_definition
relating_property_definition
Functional_unit_terminal _node_assignment
property_definition_relationship
.name = 'functional unit terminal node assignment'
related_property_definition
.description = 'gate 1'
Functional_unit
component_functional_unit
property_definition
definition
definition
product_definition_shape
Functional_unit_terminal
.name = '1Y'
component_functional_terminal
of_shape
related_shape_aspect
shape_aspect_relationship
.name = 'instantiated terminal'
relating_shape_aspect
Functional_unit_usage_view
functional_unit
Scalar_terminal_definition
.name = 'Y"
functional_unit_terminal_definition
of_shape

Figure 7. MIM mapping of key AOs and relationships in the Functional_unit_network_definition.

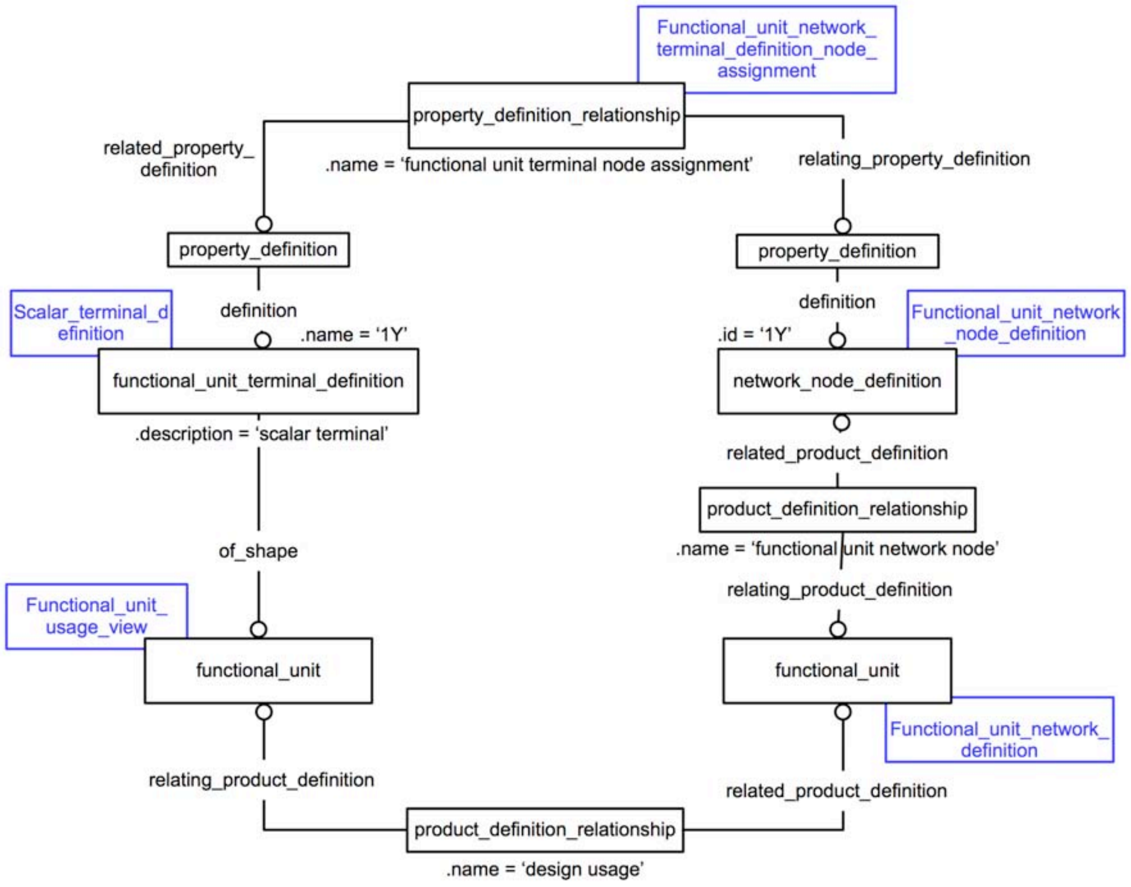
Functional_unit_network_
terminal_definition_node_
assignment
property_definition_relationship
related_property_
definition
.name = 'functional unit terminal node assignment'
relating_property_definition
property_definition
property_definition
Scalar_terminal_d
efinition
definition
.name = '1Y'
functional_unit_terminal_definition
.description = 'scalar terminal'
definition
.id = '1Y'
Functional_unit_network
_node_definition
network_node_definition
related_product_definition
product_definition_relationship
.name = 'functional unit network node'
relating_product_definition
of_shape
Functional_unit_
usage_view
functional_unit
functional_unit
Functional_unit_network_
definition
relating_product_definition
related_product_definition
product_definition_relationship
.name = 'design usage'

Figure 8. MIM mapping of Functional_unit_usage_view to Functional_unit_network_definition relationships

Gate and Pin Swapping

In the functional hierarchy each Functional_unit is a node in a path from a leaf element to the top-level block. In the process of mapping a functional element to a Packaged_part, the AOs Reference_composition_path and Reference_functional_unit_assignment_to_part are populated in order to capture the name for the relative location of the functional element in the component. This is necessary because the hierarchical definition may be more than one level deep, and thus the Functional_unit.reference_designation cannot be relied upon to disambiguate relative location because the reference_designation is unique only within the context of a single hierarchical block (Functional_unit_network_definition). Instead, the attribute Reference_functional_unit_assignment_to_part.path_alias is used to convey the name that designates the relative location in the part. In a NAND gate example where there are four NAND gates in a component, the relative locations are denoted by {"A", "B", "C", "D"}. In a design after packaging, the reference designators for individual gates would be for example {"U1A", "U1B", "U1C", "U1D"}. The path_alias attribute is not dependent on the particular hierarchical definition chosen and therefore facilitates synchronizing different CAD libraries. It is important to note that even in the case that the functional element definition is a single layer of hierarchy, the above-mentioned AOs must be populated in order to support the library to design mapping that occurs during the packaging operation. In the process of realizing a functional design, each path through the functional design hierarchy (Design_composition_path) is associated with a physical component[1]. There may be more than one Design_composition_path associated with the same physical component. An AP 210 model design constraint is that one design functional element is associated with exactly one physical component for a specific design definition. At the packaging step, a mapping between functional and physical terminals is specified. One of the important mechanisms through which improvements in a physical layout can be realized is through gate and pin swapping. AP 210 provides a mechanism to explicitly indicate that a set of functional terminals is equivalent through the AO Equivalent_functional_terminals_assignment.

Population of that AO does not necessarily mean that pins can be swapped in a particular library component or on a particular component instance in a particular design because part and component level properties may disallow certain swaps. This additional information is conveyed separately through user interaction with the design tools. The act of pin swapping occurs during physical Printed Circuit Board (PCB) layout. At this time, connections between part terminals and functional terminals may be swapped (if permitted). It is generally the case that the swap take place only on an instance of a component, rather than in the library definition of a component. The AO Terminal_swap_specification documents a particular terminal swap as a design change. The notion of whether two functional elements (i.e gates) are functionally equivalent will often be explicit in the definition of the functional object. In the example of Figure 4, the top-level network definition is composed of two functional element instances. The two functional element instances share a common definition (the same functional element). This means that they are functionally interchangeable. The interchangeability is asserted because the same Functional_product_version is referenced (indirectly) by each functional element instance. AP 210 provides an additional mechanism for explicitly stating equivalents between functional element definitions via the AO Equivalent_functional_unit_definition_assignment (typically used when synchronizing different libraries that use different naming conventions). The Component_swap_specification supports changing to a different structural representation (e.g., from a NAND representation to a NOR representation) which requires a different library hierachical functional model. Swapping a gate from one quad nand component to another quad nand component can be accomplished with the Gate_path_swap_specification.

Schematic Symbol

The functional model discussed previously includes the strutural concepts that are to be presented in a schematic symbol. In the context of an AP 210 model, a schematic symbol is a visual presentation of a functional interface (Functional_unit_usage_view).

[1] The association is accomplished through population of AOs Design_composition_path, Design_functional_unit_allocation_to_assembly_component, Design_functional_unit_allocation_to_reference_functional_unit, Reference_composition_path and Reference_functional_unit_assignment_to_part.

A drawing_sheet_revision is a representation that includes in its items a collection of annotation elements that will be visually depicted on a drawing page of specified size. The drawing_sheet_revision is explicitly related to the represented content through a presented_item_representation. It is proposed that two new specialized subtypes of presented_item_representation be defined, tentatively named schematic_symbol_representation and part_level_schematic_symbol_representation. The schematic_symbol_representation subtype is needed to constrain the size of the applied_presented_item.items attribute to be exactly one and to constrain the represented_item to be of type Functional_unit_usage_view (functional_unit). The schematic_symbol_representation will be declared in a module[2] MIM EXPRESS file. Figure 9 illustrates the relationship between a Functional unit usage view (functional_unit) and a drawing_sheet_revision through a schematic_symbol_representation.

The part_level_schematic_symbol_representation (not illustrated in Figure 9) provides additional constraints to ensure consistency between the number of terminal symbols on the symbol and the number of terminals on the part.

The drawing sheet has a defined size (presentation_size) and may be configuration controlled through the drawing_revision and drawing_definition. The drawing_sheet_revision should be associated with a global unit context as shown in the figure. The schematic_symbol_representation is represented in red to indicate its status as a proposed entity in the schematic_symbol module. The new SELECT type scms_presented_item_select is also in the schematic_symbol module but is not illustrated.

[2] The module is tentatively titled schematic_symbol. A document number is to be assigned.

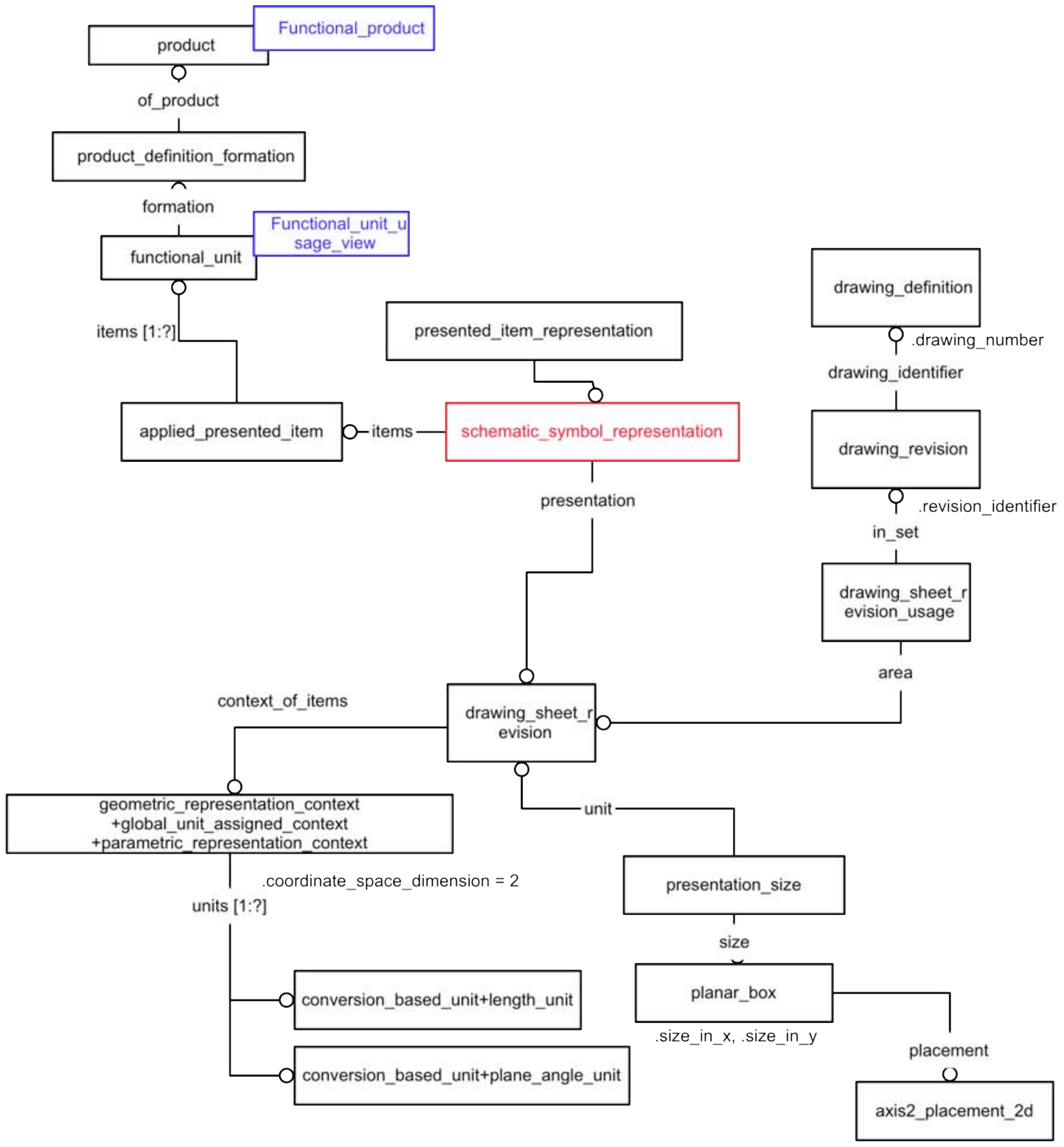
Functional_product
product
of_product
product_definition_formation
formation
Functional_unit_usage_view
functional_unit
items [1:?]
drawing_definition
.drawing_number
drawing_identifier
presented_item_representation
applied_presented_item
items
schematic_symbol_representation
drawing_revision
presentation
.revision_identifier
in_set
drawing_sheet_revision_usage
area
context_of_items
drawing_sheet_revision
geometric_representation_context
+global_unit_assigned_context
+parametric_representation_context
unit
.coordinate_space_dimension = 2
units [1:?]
presentation_size
size
conversion_based_unit+length_unit
planar_box
.size_in_x, .size_in_y
placement
conversion_based_unit+plane_angle_unit
axis2_placement_2d

Figure 9. Entities and associations in a schematic symbol drawing.

Annotation_occurrence

The primary mechanism for placing annotations within the drawing_sheet_revision is through the annotation_text_occurrence, annotation_symbol_occurrence, and annotation_curve_occurrence subtypes of annotation_occurrence. These annotations may be added directly as items of the drawing_sheet_revision that is a subtype through intermediate entities of representation. Usage of the annotation_occurrence within a schematic_symbol_representation is identical to present practice in draughting implementations. The relevant entities and relationships for population of an annotation_curve_occurrence and annotation_symbol_occurrence are illustrated in Figure 10.

Annotation text will be discussed in the following section.

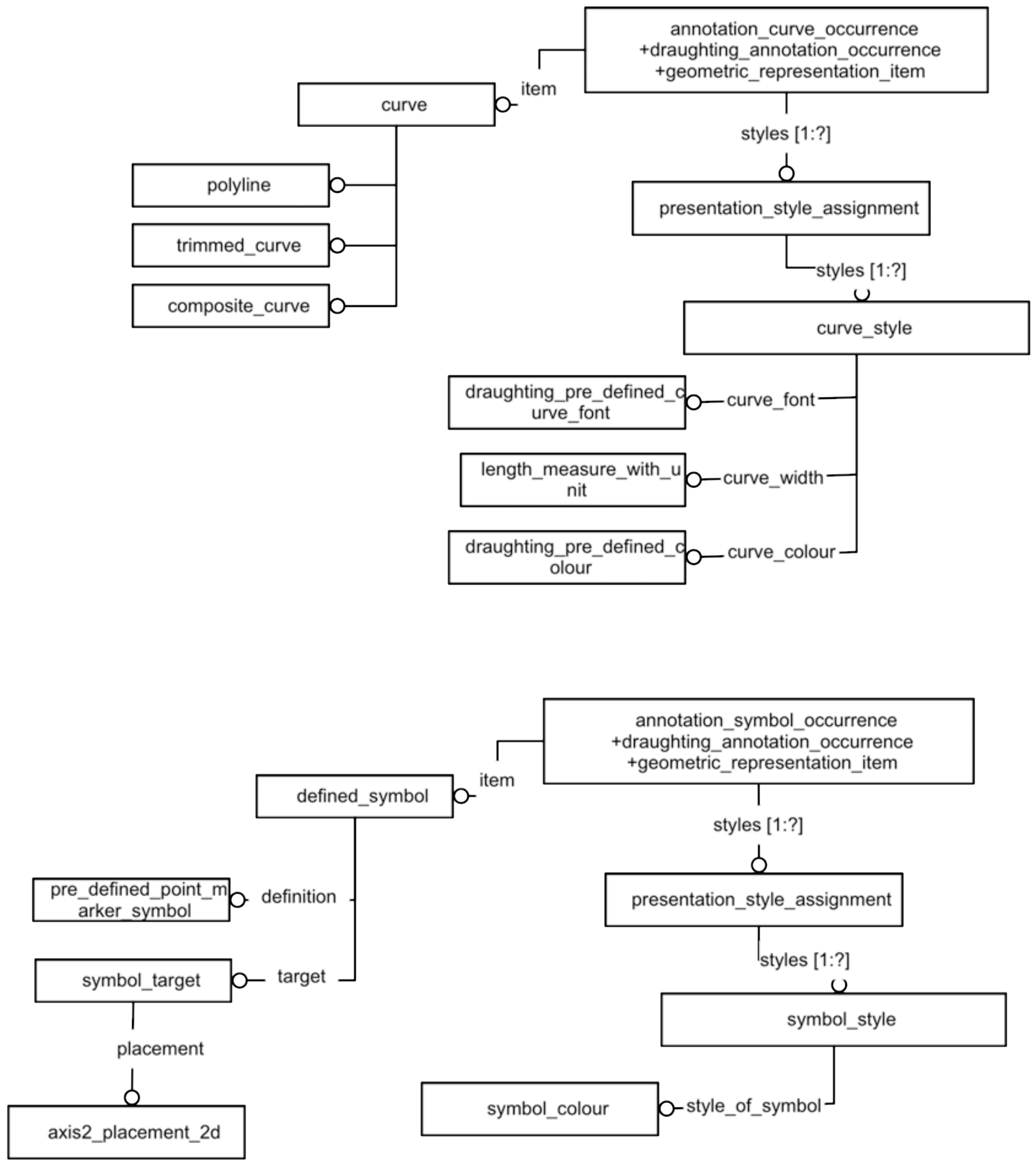
annotation_curve_occurrence
+draughting_annotation_occurrence
+geometric_representation_item
item
curve
styles [1:?]
polyline
presentation_style_assignment
trimmed_curve
styles [1:?]
composite_curve
curve_style
draughting_pre_defined_curve_font
curve_font
length_measure_with_unit
curve_width
draughting_pre_defined_colour
curve_colour
annotation_symbol_occurrence
+draughting_annotation_occurrence
+geometric_representation_item
item
defined_symbol
styles [1:?]
pre_defined_point_marker_symbol
definition
presentation_style_assignment
styles [1:?]
symbol_target
target
symbol_style
placement
symbol_colour
style_of_symbol
axis2_placement_2d

Figure 10. Annotation_curve_occurrence and annotation_symbol_occurrence.

Annotation text

An annotation_text_occurrence is typically coupled with a text_literal through the item attribute inherited from styled_item. A text_literal enables literal textual content to be added to a drawing sheet as an annotation. An additional requirement for the schematic symbol is to accurately reflect the functional model interface, and to provide the capability for defining presentation properties (including font and placment information among others) for textual data that is to be populated during application of the functional element to either a specific part or to a specific design. This additional annotation is not literal text but is a type reference. In industrial usage this concept is usually referred to as text block or text slot.

There are several scenarios considered:

1. The schematic symbol is a symbol for a functional element that is to be displayed as a separate graphic on a schematic. The value of the text string is known and contained in the functional model.
    a. An example is a 2 input NAND symbol.
    b. The value of the text string may be overridden in the design schematic after the packaging operation.
    c. This scenario occurs in the context of a primitive element in a CAD symbol library.
    d. Sub-cases:
        i. It is desired to display the name attribute of a functional_unit_terminal_definition.
        ii. It is desired to display the name of the functional element (contained in the product.id attribute) as annotation in the symbol.
2. The schematic symbol is a symbol for a part level functional model.
    a. There is a special requirement in this case in that there are two coordinated strings: the functional model terminal name and the part pin number. The two are bound with the functional allocation to part capability.
    b. This scenario occurs in the context of a part level library defintion.[3]
3. The schematic symbol is created in the design. A schematic symbol template is defined in the library and includes basic geometric properties but does not include the final symbol shape nor does it include the terminals nor does it include the name. It usually includes an initial template shape (e.g., a rectangle) that may be scaled in the design schematic. The instance of the schematic symbol defines an explicit shape, a name, and terminals. This scenario occurs during hierarchical electrical design where one or several of the top-level decompositions may use dynamic symbol creation techniques (which imply that the creation of the Functional_unit_usage_view is also dynamic). Because there is no fixed dependency on a library context in AP 210, this scenario appears to require schematic symbol template and instance extensions. Thes extensions should be the subject of a future recommended practice.
4. The value of the text string is undefined, but an attribute predefined in the AP 210 schema defines the data type for population in a design schematic.

[3] There is also the case where there is no separate part level schematic symbol created. This would occur if the part is not intended to ever be placed as a whole function in the schematic.

   a. Sub-case 1: It is desired to maintain a visual placeholder for the reference designator (e.g., U1, R1) of the component in a particular design realization. In this example, the location, size, font, etc., are maintained in the schematic symbol, and a final text value is provided in the instantiated symbol in the design schematic after the packaging activity is performed. This sub-case occurs for both scenario 1 and scenario 2. It may occur for scenario 3.
   b. Sub-case 2: It is desired to maintain a visual placeholder for the pin number of the AO Packaged_part.[4] This sub-case only occurs for scenario 1, and in this case in most schematic systems the pin number of the Packaged_part will be diaplayed after packaging instead of displaying the Functional_unit_terminal_definition.name.
   c. Sub-case 3: It is desired to maintain a visual placeholder for the name denoting the relative location of the functional element in the part (e.g., “A”, “B”, “C”, “D”). Note that the placement of this placeholder will need to be coordinated with the placement of the placeholder for the reference designator of the component so as to appear as one string in the schematic design.

5. The value of the text string is undefined, but a type declaration for the combination of a user defined property name and value is necessary.
   a. For example, a schematic may be used to display intended, simulated, or measured power dissipation values for a component.
      i. The symbol should have a text slot for the property name.
      ii. The symbol should have a text slot for the property value.
6. The value of the text string is undefined, but a type declaration for terms defined in IEC 81714-2:2006 is necessary. IEC 81714-2:2006 is based on STEP draughting concepts and provides process details from the perspective of a CAD tool user interface about how to build exchangeable schematic symbols. IEC 81714-2:2006 is not a data exchange standard. IEC 81714-2:2006 specifies four cases to be supported in a symbol model:
   a. Identifying block (e.g., REF_DES_N),
   b. Descriptive block (e.g., COUNTRY_CODE),
   c. Connect node block (e.g., PROD_TERM_DES).
   d. Directions for drawing connecting lines onto schematic connect nodes. The following is extracted from the document.

   In order to facilitate computer-aided tools supporting automatic routing of connecting lines in diagrams, each schematic connect node of a reference symbol should be provided with information providing the permitted directions for drawing connecting lines onto the schematic connect node. The permitted directions shall be defined as one or more sectors under which

[4] In the design application there are two use cases for schematic presentation: pre-packaged and post-packaged. Pre-packaged names are those directly from the symbol library (e.g., IN1 and NAND) and are not cognizant of the design hiearchy. That is, Functional_unit reference designators are local to a node in the hierarchical definition. Post-packaged names are dependent on the component reference designator assigned and the relative placement in the component of the particular primitive (e.g., pin 1 and U1-A). Refer to scenario number (4) for the post-packaged case.

the connecting lines may be drawn onto the schematic connect node. … Due to drawing practice, the sectors are defined in multiple steps of 45° based on the depicted reference system. Each sector shall be described by the start angle and end angle based on a counter-clockwise direction.

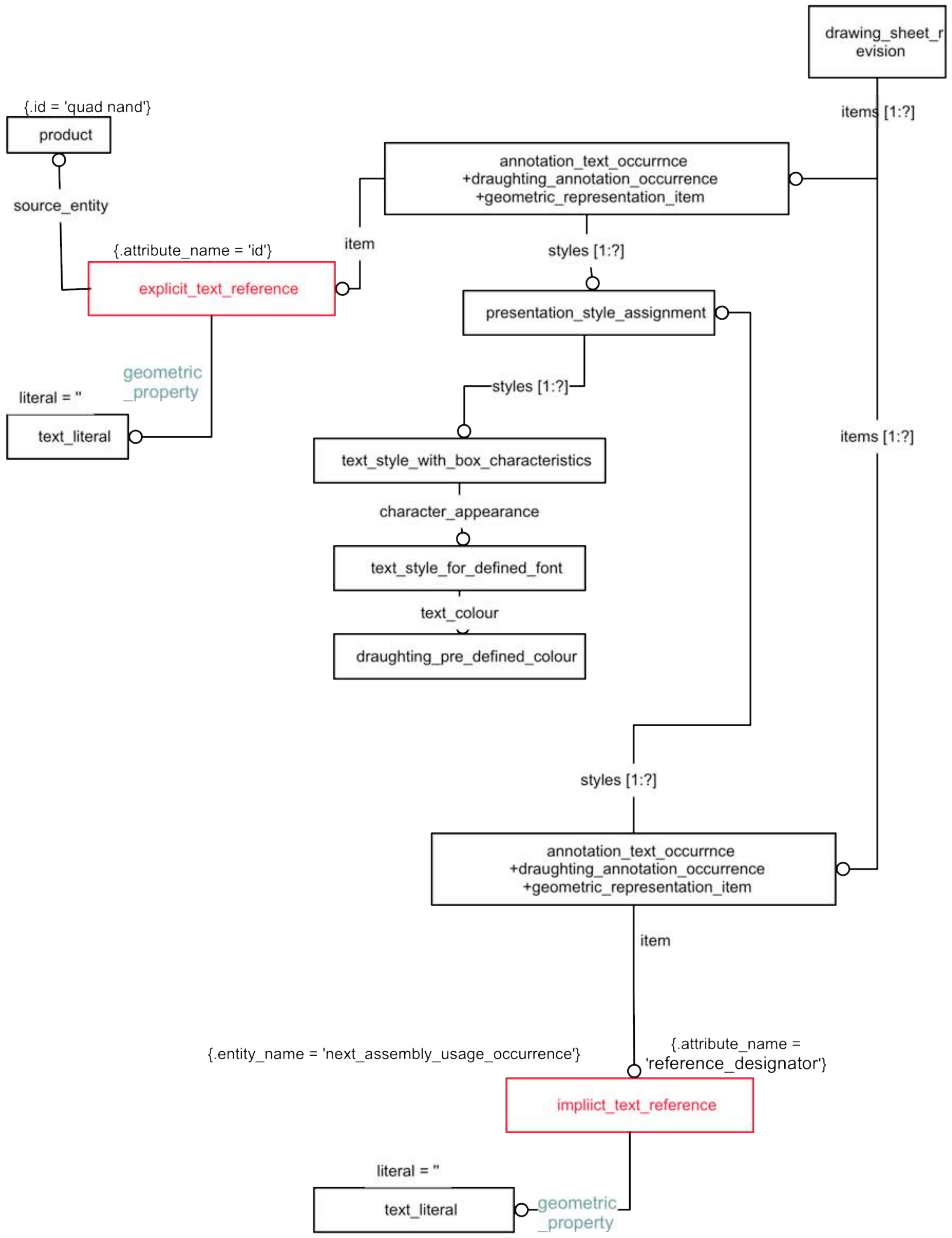
drawing_sheet_r evision
items [1:?]
{.id = 'quad nand'}
product
source_entity
annotation_text_occurrnce +draughting_annotation_occurrence +geometric_representation_item
item
{.attribute_name = 'id'}
explicit_text_reference
styles [1:?]
presentation_style_assignment
geometric _property
literal = ''
text_literal
styles [1:?]
text_style_with_box_characteristics
items [1:?]
character_appearance
text_style_for_defined_font
text_colour
draughting_pre_defined_colour
styles [1:?]
annotation_text_occurrnce +draughting_annotation_occurrence +geometric_representation_item
item
{.entity_name = 'next_assembly_usage_occurrence'}
{.attribute_name = 'reference_designator'}
impliict_text_reference
literal = ''
text_literal
geometric _property

Figure 11. Annotation_text_occurrence using explicit_text_reference and implicit_text_reference.

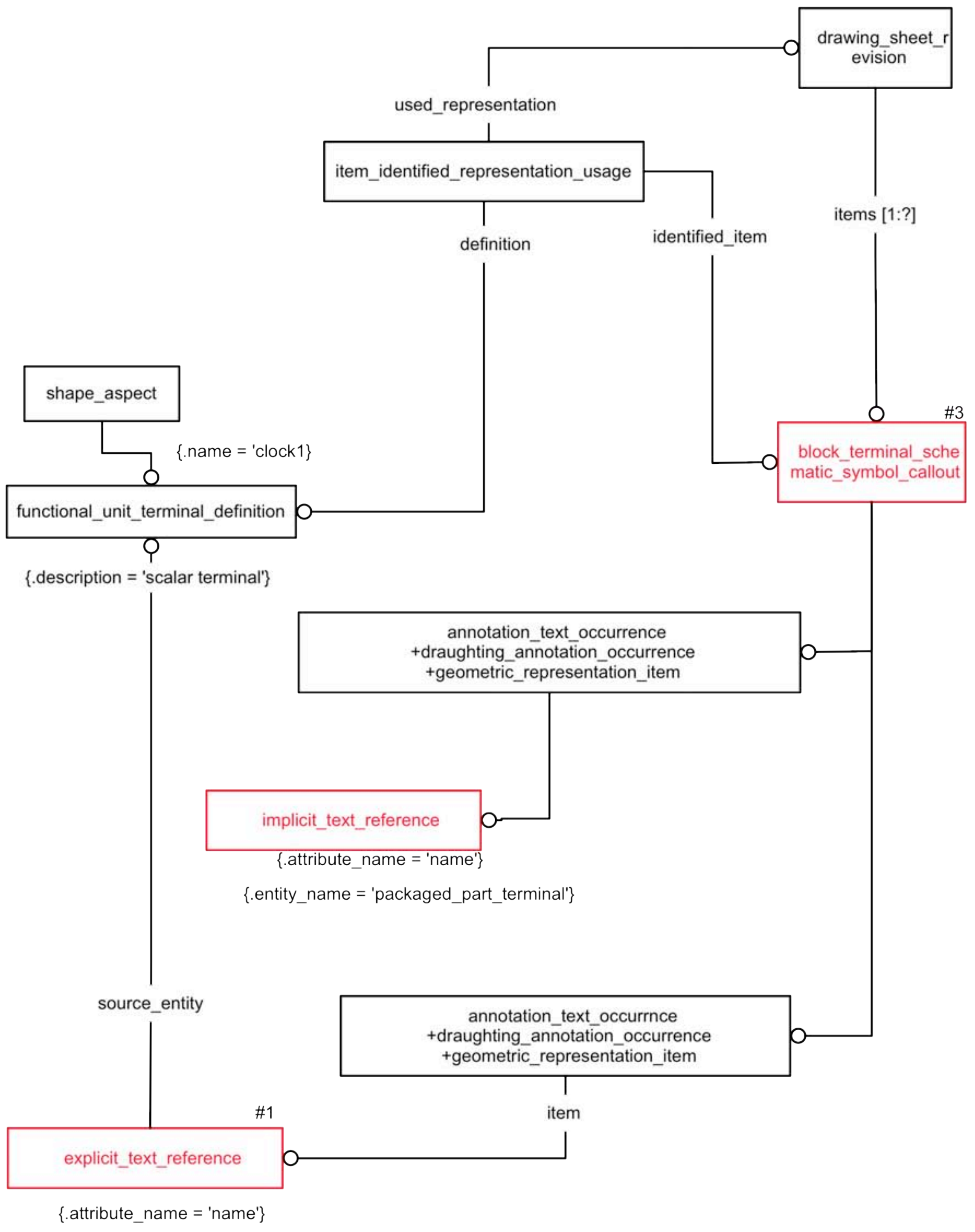
drawing_sheet_revision
used_representation
item_identified_representation_usage
definition
identified_item
items [1:?]
shape_aspect
{.name = 'clock1}
#3
block_terminal_schematic_symbol_callout
functional_unit_terminal_definition
{.description = 'scalar terminal'}
annotation_text_occurrence +draughting_annotation_occurrence +geometric_representation_item
implicit_text_reference
{.attribute_name = 'name'}
{.entity_name = 'packaged_part_terminal'}
source_entity
annotation_text_occurrnce +draughting_annotation_occurrence +geometric_representation_item
#1
item
explicit_text_reference
{.attribute_name = 'name'}

Figure 12 Representation of the schematic symbol elements for a functional interface terminal not mapped to a part terminal

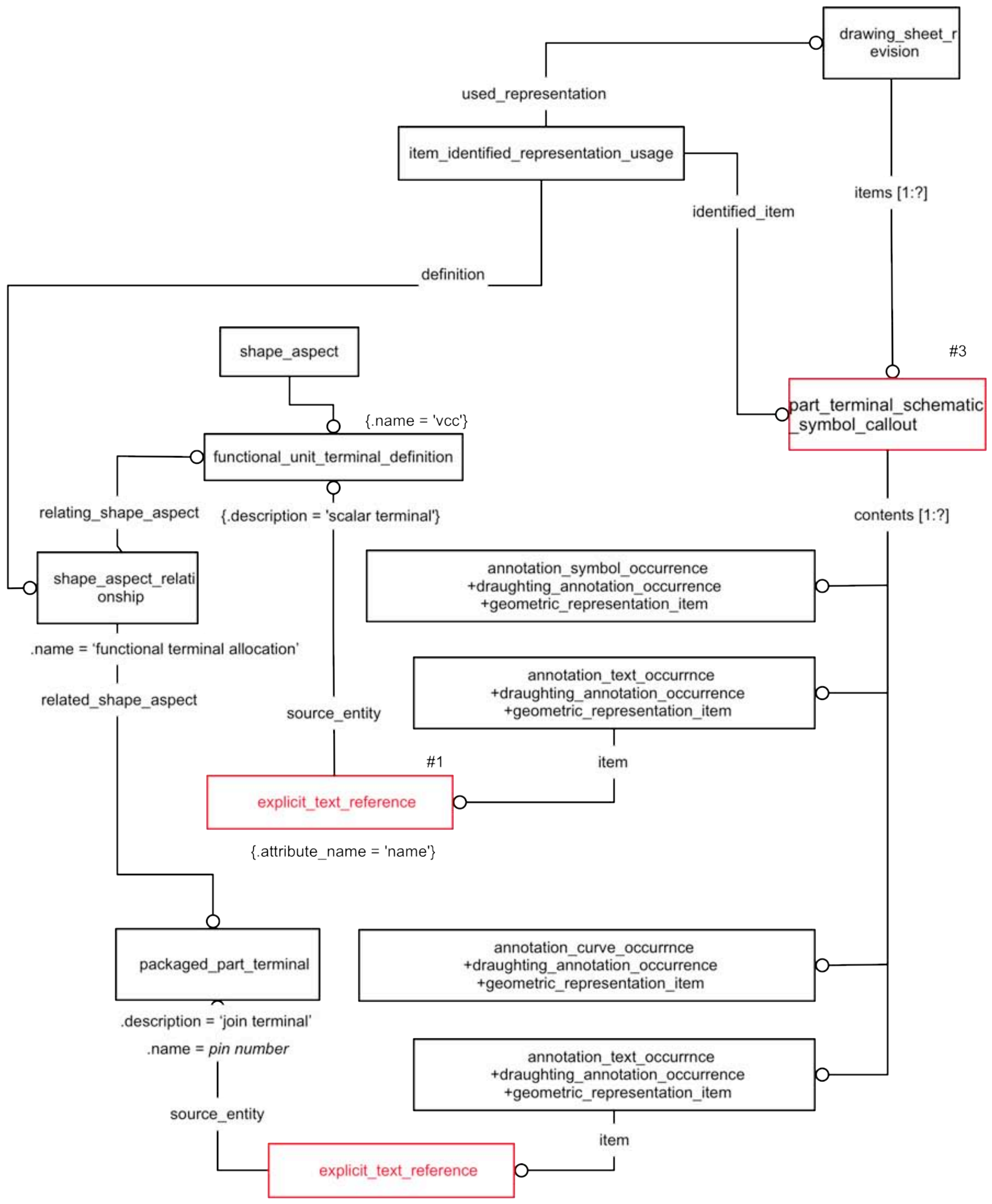
drawing_sheet_revision
used_representation
item_identified_representation_usage
identified_item
items [1:?]
definition
shape_aspect
#3
part_terminal_schematic_symbol_callout
{.name = 'vcc'}
functional_unit_terminal_definition
relating_shape_aspect
{.description = 'scalar terminal'}
contents [1:?]
shape_aspect_relationship
annotation_symbol_occurrence +draughting_annotation_occurrence +geometric_representation_item
.name = 'functional terminal allocation'
annotation_text_occurrnce +draughting_annotation_occurrence +geometric_representation_item
related_shape_aspect
source_entity
#1
item
explicit_text_reference
{.attribute_name = 'name'}
packaged_part_terminal
annotation_curve_occurrnce +draughting_annotation_occurrence +geometric_representation_item
.description = 'join terminal'
.name = pin number
annotation_text_occurrnce +draughting_annotation_occurrence +geometric_representation_item
source_entity
item
explicit_text_reference

Figure 13 Representation of the schematic symbol elements for a functional interface terminal mapped to a part terminal

Proposed recommendations to address scenarios

1. The value of the text string is known and contained in the functional model. The schematic symbol model is explicitly bound to the functional model through the

entity schematic_symbol_representation. Adding an extensible SELECT type annotation_text_occurrence_item and a new ABSTRACT entity type text_reference will extend the domain of annotation_text_occurrence.item. The attribute annotation_text_occurrence.item shall specify the subtype explicit_text_reference. The attribute explicit_text_reference.source_instance shall specify the entity type instance that contains the relevant attribute and the attribute explicit_text_reference.name shall specify the name of the attribute in the entity type instance that is the text string to be displayed. This scenario is illustrated in Figure 11, along with the supporting entities needed to define the font, style, size, and placement of the text annotation element.

2. The schematic symbol is a symbol for a part level functional model.
    a. A part_level_schematic_symbol_representation subtype of schematic_symbol_representation represents the schematic symbol.
    b. The part_level_schematic_symbol_representation will specify an applied_presented_item that will specify the functional_unit and the packaged_part in its items attribute.
    c. The part_level_schematic_symbol_representation will specify a drawing_sheet_revision that contains (using the inverse attribute item_identified_representation_usage.used_representation) an item_identified_representation_usage for each Functional_usage_view_to_part_terminal_assignment related to the packaged_part in its drawing_sheet_revision.items attribute. The item_identified_representation_usage.definition attribute shall specify the Functional_usage_view_to_part_terminal_assignment. There may be other contents of the drawing_sheet_revision (e.g., schematic_property_symbol_callouts).
    d. For terminal and pin numbering, the part_level_schematic_symbol_representation will constrain each item_identified_representation_usage specified in (c) to specify a part_terminal_schematic_symbol_callout. Figure 13 illustrates the schematic symbol data population. The following constraints apply:
        i. First annotation_text_occurrence.item shall specify an instance of explicit_text_reference. The attribute explicit_text_reference.source_instance shall specify an instance of functional_terminal_definition and the attribute explicit_text_reference.attribute_name shall specify 'name'.
        ii. Second annotation_text_occurrence.item shall specify a second instance of explicit_text_reference. The attribute explicit_text_reference.source_instance shall specify an instance of packaged_part_terminal and the attribute explicit_text_reference. attribute_name shall specify 'name'.
    e. For general property presentation see scenario (5) that uses schematic_property_symbol_callouts.

f. Sub element symbols shall only be displayed in the schematic symbol for a part level functional model through the use of geometric transforms using mapped_item and representation_map. This sub-scenario occurs often in the process and industrial plane industry but is seldom seen to date in the electronics industry. The capability may be useful for complex parts like FPGAs.

The result is that the functional terminal name and part pin number assigned in the symbol are explicitly definitional and are displayed as such at the schematic design level independently of the packaged state of the symbol in the schematic.

3. Recommendations for the parametric symbol template are deferred to a future recommended practice.
4. The value of the text string is undefined, but an attribute predefined in the AP 210 schema defines the data type for population in a design schematic.
   a. The subtype schematic_property_symbol_callout shall be instantiated if the callout is not a terminal_schematic_symbol_callout.
   b. The attribute annotation_text_occurrence.item shall specify an instance of the new subtype implicit_text_reference (#4a).
   c. The attribute implicit_text_reference.entity_name shall specify the entity type name that contains the relevant attribute and the attribute implicit_text_reference.attribute_name shall specify the name of the attribute in the entity type that is the text string to be displayed.
   d. For the example of a reference designator in a design, implicit_text_reference.entity_name = 'NEXT_ASSEMBLY_USAGE_OCCURRENCE' and implicit_text_reference.attribute_name = 'REFERENCE_DESIGNATION'.[5] Figure 11 illustrates symbol data populated for this example.

At the time of schematic creation, an instance of explicit_text_reference_occurrence (explicit_text_reference_occurrence is a subtype of explicit_text_reference) will be populated that will specify:

1. The implicit_text_reference (#4a) is specified by explicit_text_reference_occurrence.type_declaration.
2. The next_assembly_usage_occurrence is specified by explicit_text_reference_occurrence\explicit_text_reference.source_in stance. In order to determine the correct instance of next_assembly_usage_occurrence, a processor will need to query the relevant Design_composition_path, Design_functional_unit_allocation_to_assembly_component, Design_functional_unit_allocation_to_reference_functional_unit, Reference_composition_path and Reference_functional_unit_assignment_to_part instances.

[5] String references to entity and attribute names are usually uppercase in EXPRESS schemas.

3. The attribute ‘attribute_name’ inherited from explicit_text_reference is redeclared as DERIVEd and is not provided in the exchange data.

e. For the example of a part pin number, implicit_text_reference.entity_name = ‘PACKAGED_PART_TERMINAL’ and implicit_text_reference.attribute_name = ‘PACKAGED_PART_TERMINAL.NAME’. Note that the name attribute of Packaged_part_terminal is inherited from Shape_element. This example occurs when the schematic symbol is not a part level schematic symbol. Figure 12 illustrates the symbol data population for this example.

After the packaging operation, an instance of explicit_text_reference_occurrence (explicit_text_reference_occurrence is a subtype of explicit_text_reference) will be populated that will specify:

1. The implicit_text_reference (#4a) is specified by explicit_text_reference_occurrence.type_declaration.
2. The packaged_part_terminal is specified by explicit_text_reference_occurrence\explicit_text_reference.source_instance. In order to determine the correct instance of packaged_part_terminal, a processor will need to query the relevant Design_composition_path, Design_functional_unit_allocation_to_assembly_component, Design_functional_unit_allocation_to_reference_functional_unit, Reference_composition_path and Reference_functional_unit_assignment_to_part instances.
3. The attribute ‘attribute_name’ inherited from text_reference is redeclared as DERIVEd and is not provided in the exchange data.

f. For the example of a name denoting a relative location of the functional element in the part, implicit_text_reference.entity_name = ‘REFERENCE_FUNCTIONAL_ASSIGNMENT_TO_PART’ and implicit_text_reference.attribute_name = ‘REFERENCE_FUNCTIONAL_ASSIGNMENT_TO_PART.PATH_ALIAS’. This example occurs when the schematic symbol is for a primitiv element or for an intermediate element defined by the part manufacturer (e.g., xxHC244xx octal buffer has both primitive elements and higher level elements that may be instantiated in a design). When a higher level element is instantiated, that blocks lower level elements that would ordinarily be the decomposition of the higher level element from being instantiated. Note that the EXPRESS string values in this example denote AO names and not MIM entity names.

After the packaging operation, an instance of explicit_text_reference_occurrence will be populated that will specify:

1. The implicit_text_reference (#4a) is specified by explicit_text_reference_occurrence.type_declaration.

        2. The reference_functional_assignment_to_part is specified by explicit_text_reference_occurrence\explicit_text_reference.source_instance. In order to determine the correct instance of reference_functional_assignment_to_part, a processor will need to query the relevant Design_composition_path, Design_functional_unit_allocation_to_assembly_component, and Design_functional_unit_allocation_to_reference_functional_unit instances.
        3. The attribute 'attribute_name' inherited from text_reference is redeclared as DERIVEd and is not provided in the exchange data.
    g. This is the case where it is desired to combine use case (e) and (f) above so as to display the reference designator and relative location of the functional element in the component into what appears to be one string for display at the schematic design application (e.g. "U1-A" is displayed). A delimiter between the two references is required. A new entity composite_sequential_text_reference would be populated that would specify the reference designator related explicit_text_occurrence from (e) as composite_sequential_text_reference.collected_references[1], a text_literal "-" as composite_sequential_text_reference.collected_references[2], and the explicit_text_occurrence from (f) above as composite_sequential_text_reference.collected_references[3]. The specific geometric transforms necessary to support this must be provided by the pre-processor and are not explicitly defined in the schematic extension to ISO 10303-46.
5. The value of the text string is undefined, but a type declaration for the combination of a user defined property name and value is necessary. A schematic may be used to display intended, simulated, or measured power dissipation values for a component. This is an extension of scenario (4).
    a. Model_parameter,
    b. Parameter_assignment,
    c. Independent_property,
    d. Applied_independent_property

   Another new subtype of callout is proposed: schematic_property_symbol_callout. This callout will contain annotation_text_occurrence instances that specify implicit_text_references that are the composition of e.g., Model_parameter or Parameter_assignment or of Independent_property or of Assigned_property. The key extension here is to add these items to item_identified_representation_usage.definition so as to allow an aggregate to be created of the components of the above entity types. In order to specify a particular display sequence the new entity type composite_sequential_text_reference is provided.

6. The value of the text string is undefined, but a type declaration for terms defined in IEC 81714-2:2006 is necessary.

   A preliminary review of the requirements results in the conclusion that the current recommendations for extensions can support the requirements of IEC 81714 but it is future activity to do an implementation to validate that conclusion.

Terminal schematic symbol callout

It is mandatory to establish an explicit relationship between the collection of annotation on the schematic symbol that is presenting an instance of a terminal of the interface and the instance of the terminal. This applies whether the symbol is a part level symbol or a block symbol. A draughting_callout is a well-established concept in STEP draughting implementations that is widely employed for dimensions, notes, and GD&T annotations. A draughting_callout is typically associated with predefined draughting concepts such as leaders, and serves as a grouping mechanism for a collection of related annotation elements. To enable grouping and association of the annotation elements associated with a terminal in a schematic symbol, a clone of draughting_callout is proposed, titled terminal_schematic_symbol_callout. A clone is required because the contents of draughting_callout are not appropriate for a terminal. The proposed terminal_schematic_symbol_callout would be employed to group annotation elements related to a terminal in the interface (Functional_unit_usage_view).

For the case of a block symbol that is not a part level symbol, it is proposed to use an item_identified_representation_usage to associate a functional_unit_terminal_definition with a block_terminal_schematic_symbol_callout subtype of terminal_schematic_symbol_callout for this purpose. For the case of a block symbol that is not a part level symbol, block_terminal_schematic_symbol_callout will have a rule to enforce a one to one relationship to a terminal through an item_identified_representation_usage. Figure 12 illustrates that case.

Figure 13 illustrates a potential schematic symbol representation of a terminal for a part. In this example, two text annotations are present, one for the signal name, and one for the associated pin number. This is possible because of the availability of the part_level_schematic_symbol_representation and of the availability of the part_terminal_schematic_symbol_callout subtype of terminal_schematic_symbol_callout.

The proposal for the case of a part level symbol is described in detail in the previous section in scenario two (2).

In addition to the text annotations, there is an annotation_symbol_occurrence and an annotation_curve_occurrence. The recommended practice is for the placement of the annotation_symbol_occurrence to represent the terminal connection point in the schematic symbol.

To meet the requirements of IEC 81714:2006 for approach angle allowed ranges, an additional property with range is required but the details will be covered in a forthcoming recommendation.

Summary of proposed changes to ISO 10303

The added entity types in visual presentation (ISO 10303-46) include:

composite_sequential_text_reference,
schematic_symbol_representation,
schematic_symbol_callout,
explicit_text_reference,
explicit_text_reference_occurrence,
implicit_text_reference,
text_reference.

The SELECT types text_reference_source and text_literal_or_text_reference are added to visual presentation (ISO 10303-46).
The SELECT type annotation_text_occurrence_item in visual presentation (ISO 10303-46) is changed to an EXTENSIBLE GENERIC_ENTITY SELECT.
The entity type draughting_annotation_occurrence in aic_draughting_annotation (ISO 10303-504) is modified.
A schematic_symbol module is created with the following EXPRESS declarations:

scms_presented_item_select,
schematic_symbol_representation,
part_level_schematic_symbol_representation,
terminal_schematic_symbol_callout,
block_terminal_schematic_symbol_callout,
part_terminal_schematic_symbol_callout,
scms_text_reference_source,
scms_get_presented_function,
scms_get_presented_functional_terminal.

Case Study – Unified representation of a quad nand packaged component

In order to exercise and validate the proposed concepts, a detailed example was populated and analyzed for the representation of a quad nand packaged component. The part-level interface includes 14 terminals. A network definition is populated containing 4 instances of a single nand functional element, and a single instance of a power block. A simple schematic symbol is populated for each of the three interfaces that include annotation symbols, text, and curves. The part level symbol is a part_level_schematic_symbol_representation. The gate and power block symbols are schematic_symbol_representation. The text_reference is populated in both scenarios – a direct reference (explicit_text_reference) to an existing entity (i.e. pin number) in the top-level block; and implicit_text_reference for placeholder values (i.e. reference designator). The functional and schematic model was integrated with a previously existing fully populated package model. The fully populated model successfully passed validation rule checking based on the modified long form schema based on the LKSoft Express validation implementation.

Appendix: Details of proposed extensions to support schematic symbol representation

In order to exercise the proposed concepts for schematic symbol representation, a number of minor changes were made to stepmod modules and resources. These changes represent a full implementation, unless otherwise noted, of the proposed concepts, but need further evaluation and documentation as part of the formal STEP modular development process.

A longform file is located at http://stepmod.cvs.sourceforge.net/viewvc/stepmod/stepmod/data/modules/schematic_symbol/dvlp/express/schematic_symbol_mim_lf.exp.

Further development may be monitored at:

http://locke.dcnicn.com/bugzilla/iso10303/show_bug.cgi?id=3889.

The initial changes are as follows:

```
(* in module schematic_symbol *)
SCHEMA Schematic_symbol_mim;
USE FROM Drawing_definition_mim;  -- ISO/TS 10303-1309
USE FROM Mechanical_design_presentation_representation_with_draughting_mim; -- ISO/TS
10303-1315
--USE FROM Picture_representation_mim;  -- ISO/TS 10303-1308
--USE FROM Shape_feature_mim;    -- ISO/TS 10303-1764
USE FROM Text_representation_mim;     -- ISO/TS 10303-1750
--USE FROM Wireframe_2d_mim; -- ISO/TS 10303-1347
USE FROM Packaged_part_black_box_model_mim;
USE FROM Associative_text_mim;
USE FROM draughting_element_schema;
USE FROM presentation_definition_schema;
USE FROM Functional_decomposition_to_design_mim;
USE FROM Functional_assignment_to_part_mim;
USE FROM aic_draughting_annotation;

--schema collection for schematic symbol project.
--file mim.exp
--creation date: 2/9/2011
--modified date: 3/6/2011

TYPE scms_presented_item_select = EXTENSIBLE GENERIC_ENTITY SELECT BASED_ON
presented_item_select WITH (
    functional_unit,
    packaged_part);
 END_TYPE;

 (*
 modification to ISO 10303-46:
  TYPE annotation_text_occurrence_item = EXTENSIBLE GENERIC_ENTITY SELECT(
   text_literal,
   text_reference,
   annotation_text,
   annotation_text_character,
   defined_character_glyph,
   composite_text);
  END_TYPE;
 *)
(* in module schematic_symbol *)

ENTITY schematic_symbol_representation
  SUBTYPE OF (presented_item_representation);
  SELF\presented_item_representation.presentation : drawing_sheet_revision;
  SELF\presented_item_representation.item         : applied_presented_item;
 DERIVE
  presented_function            : functional_unit  := scms_get_presented_function(item);
  presented_functional_terminal : SET OF functional_unit_terminal_definition  :=
scms_get_presented_functional_terminal(SELF, presentation);
  terminal_symbol_map : SET OF item_identified_representation_usage :=
bag_to_set(QUERY(iiru <*  (USEDIN(presentation,
'PRODUCT_PROPERTY_REPRESENTATION_SCHEMA.ITEM_IDENTIFIED_REPRESENTATI
ON_USAGE.USED_REPRESENTATION'))|
'SCHEMATIC_SYMBOL_MIM.SCHEMATIC_TERMINAL_CALLOUT' IN
TYPEOF(iiru\item_identified_representation_usage.identified_item)));
  schematic_terminal_callout    : SET OF terminal_schematic_symbol_callout    := QUERY( it <*
presentation.items |
'SCHEMATIC_SYMBOL_MIM.TERMINAL_SCHEMATIC_SYMBOL_CALLOUT' IN TYPEOF(it));
 WHERE
   WR1 : SIZEOF(QUERY(pft <* presented_functional_terminal | NOT
(pft\shape_aspect.of_shape = presented_function))) = 0;
  --each presented_functional_terminal shall specify the presented_function as its of_shape;
   WR2 : SIZEOF(terminal_symbol_map) = SIZEOF(schematic_terminal_callout);
--- each schematic_terminal_symbol shall be mapped to a functional terminal.
---note that the size of the functional terminals need not be the same as the number of symbol
callouts to handle the case of a part level symbol.
   WR3 :
('SCHEMATIC_SYMBOL_MIM.PART_LEVEL_SCHEMATIC_SYMBOL_REPRESENTATION' IN
TYPEOF(SELF)) OR (SIZEOF(presented_functional_terminal) =
SIZEOF(terminal_symbol_map));
---Either the symbol shall be a part level symbol or only functional terminals may map to
schematic_terminal_symbols;
END_ENTITY;


(* in module schematic_symbol *)
FUNCTION scms_get_presented_function(input : applied_presented_item) : functional_unit;
LOCAL
   fun : SET OF functional_unit := (QUERY(it <* (input\applied_presented_item.items) |
'FUNCTIONAL_USAGE_VIEW_MIM.FUNCTIONAL_UNIT' IN TYPEOF(it)));
END_LOCAL;
RETURN(fun[LOINDEX(fun)]);
END_FUNCTION;

(* in module schematic_symbol *)
FUNCTION scms_get_presented_functional_terminal(input1 : schematic_symbol_representation;
                                   input2 : drawing_sheet_revision) : SET OF
functional_unit_terminal_definition;
LOCAL
 function_only : BOOLEAN :=
NOT('SCHEMATIC_SYMBOL_MIM.PART_LEVEL_SCHEMATIC_SYMBOL_REPRESENTATIO
N' IN TYPEOF(input1));
 iiru : SET OF item_identified_representation_usage := bag_to_set(USEDIN(input2,
'PRODUCT_PROPERTY_REPRESENTATION_SCHEMA.ITEM_IDENTIFIED_REPRESENTATI
ON_USAGE.USED_REPRESENTATION'));
 futd : SET OF functional_unit_terminal_definition := [];
 sar  : SET OF shape_aspect_relationship := [];
END_LOCAL;
IF function_only
THEN
 REPEAT i := LOINDEX(iiru) TO HIINDEX(iiru);
   IF (('FUNCTIONAL_USAGE_VIEW_MIM.FUNCTIONAL_UNIT_TERMINAL_DEFINITION') IN
TYPEOF( iiru[i]\item_identified_representation_usage.definition)) THEN
   futd := futd + iiru[i]\item_identified_representation_usage.definition;
   END_IF;
END_REPEAT;
RETURN(futd);
ELSE
 REPEAT i := LOINDEX(iiru) TO HIINDEX(iiru);
   IF ((('PRODUCT_PROPERTY_DEFINITION_SCHEMA.SHAPE_ASPECT_RELATIONSHIP')
IN TYPEOF( iiru[i]\item_identified_representation_usage.definition)) AND
   ((iiru[i]\item_identified_representation_usage.definition\shape_aspect_relationship.name =
'functional terminal allocation')))
      THEN
   futd := futd +
iiru[i]\item_identified_representation_usage.definition\shape_aspect_relationship.relating_shape_
aspect;
   END_IF;
END_REPEAT;
RETURN(futd);
END_IF;

(*
this function has a switch to look for a functional terminal as either the direct
item_identified_representation_usage.definition or
as the sar.relating_shape_aspect of the item_identified_representation_usage.definition.
The first case is when the symbol is NOT  part level symbol.
The second case is when the symbol IS a part level symbol.
The path is:
drawing_sheet_revision <-
item_identified_representation_usage.used_representation
item_identified_representation_usage.definition ->
functional_unit_terminal_definition;
or
drawing_sheet_revision <-
item_identified_representation_usage.used_representation
item_identified_representation_usage.definition ->
sar
{sar.name = 'functional terminal allocation'}
sar.relating_shape_aspect;
*)
END_FUNCTION;

(* in module schematic_symbol *)
ENTITY part_level_schematic_symbol_representation
  SUBTYPE OF (schematic_symbol_representation);
 DERIVE
  presented_part  : packaged_part  := scms_get_presented_part(item);
  presented_part_terminal : SET OF packaged_part_terminal :=
scms_get_presented_part_terminal(presentation);
WHERE
   WR1 : SIZEOF(QUERY(pft <* presented_part_terminal | NOT (pft\shape_aspect.of_shape =
presented_part))) = 0;
  --each presented_part_terminal shall specify the presented_part as its of_shape;
   WR2 : (SIZEOF(presented_part_terminal) = SIZEOF(terminal_symbol_map));
  --each presented_part_terminal shall have a symbol;
END_ENTITY;

(* in module schematic_symbol *)
FUNCTION scms_get_presented_part_terminal(input : drawing_sheet_revision) : SET OF
functional_unit_terminal_definition;
LOCAL
 iiru : SET OF item_identified_representation_usage := bag_to_set(USEDIN(input,
'PRODUCT_PROPERTY_REPRESENTATION_SCHEMA.ITEM_IDENTIFIED_REPRESENTATI
ON_USAGE.USED_REPRESENTATION'));
 ppt : SET OF packaged_part_terminal     := [];
 sar  : SET OF shape_aspect_relationship := [];
END_LOCAL;

 REPEAT i := LOINDEX(iiru) TO HIINDEX(iiru);
    IF (('PRODUCT_PROPERTY_DEFINITION_SCHEMA.SHAPE_ASPECT_RELATIONSHIP' IN
TYPEOF( iiru[i]\item_identified_representation_usage.definition)) AND
    ((iiru[i]\item_identified_representation_usage.definition\shape_aspect_relationship.name =
'functional terminal allocation')))
       THEN
    ppt := ppt +
iiru[i]\item_identified_representation_usage.definition\shape_aspect_relationship.related_shape_
aspect;
    END_IF;
END_REPEAT;
RETURN(ppt);
END_FUNCTION;

(* in module schematic_symbol *)
FUNCTION scms_get_presented_part(input : applied_presented_item) : packaged_part;
LOCAL
   pp : SET OF packaged_part := (QUERY(it <* (input\applied_presented_item.items) |
'PACKAGED_PART_BLACK_BOX_MODEL_MIM.PACKAGED_PART' IN TYPEOF(it)));
END_LOCAL;
RETURN(pp[LOINDEX(pp)]);
END_FUNCTION;

(*
addition to draughting_elements_schema: ISO 10303-101
ENTITY schematic_symbol_callout--  include in p101
       SUBTYPE OF (geometric_representation_item);
       contents : SET [1:?] OF draughting_callout_element;
INVERSE
   presented_concept : item_identified_representation_usage FOR identified_item;
END_ENTITY;
*)

(* in module schematic_symbol *)
ENTITY terminal_schematic_symbol_callout
```

```
        ABSTRACT SUPERTYPE OF (ONEOF(block_terminal_schematic_symbol_callout,
part_terminal_schematic_symbol_callout))
        SUBTYPE OF (schematic_symbol_callout);
END_ENTITY;

(* in module schematic_symbol *)
ENTITY block_terminal_schematic_symbol_callout
        SUBTYPE OF (terminal_schematic_symbol_callout);
DERIVE
 functional_terminal : functional_unit_terminal_definition    :=
presented_concept\item_identified_representation_usage.definition;
 signal_name_data : SET [1:?] OF explicit_text_reference   :=
bag_to_set(USEDIN(functional_terminal,
'PRESENTATION_DEFINITION_SCHEMA.EXPLICIT_TEXT_REFERENCE.SOURCE'));
 annotation_text_occurrence : SET OF annotation_text_occurrence := QUERY(c <*
SELF\schematic_symbol_callout.contents |
('PRESENTATION_DEFINITION_SCHEMA.ANNOTATION_TEXT_OCCURRENCE' IN
TYPEOF(c)) AND
('PRESENTATION_DEFINITION_SCHEMA.EXPLICIT_TEXT_REFERENCE' IN
TYPEOF(c\annotation_text_occurrence.item)));
WHERE
 WR1 : SIZEOF(QUERY(ato <* annotation_text_occurrence | (signal_name_data[1] =
ato\annotation_text_occurrence.item\explicit_text_reference.source))) = 1;
END_ENTITY;

(* in module schematic_symbol *)
ENTITY part_terminal_schematic_symbol_callout
        SUBTYPE OF (terminal_schematic_symbol_callout);
DERIVE
 functional_allocation_to_part : shape_aspect_relationship              :=
SELF\schematic_symbol_callout.presented_concept\item_identified_representation_usage.definit
ion;
 functional_terminal : functional_unit_terminal_definition    :=
functional_allocation_to_part\shape_aspect_relationship.relating_shape_aspect;
 part_terminal: packaged_part_terminal                 :=
functional_allocation_to_part\shape_aspect_relationship.related_shape_aspect;
 signal_name_data: SET [1:?] OF explicit_text_reference :=
bag_to_set(USEDIN(functional_terminal,
'PRESENTATION_DEFINITION_SCHEMA.EXPLICIT_TEXT_REFERENCE.SOURCE'));
 pin_number_data: SET [1:?] OF explicit_text_reference   := bag_to_set(USEDIN(part_terminal,
'PRESENTATION_DEFINITION_SCHEMA.EXPLICIT_TEXT_REFERENCE.SOURCE'));
 annotation_text_occurrence : SET OF annotation_text_occurrence :=
       QUERY(c <* SELF\schematic_symbol_callout.contents |
('PRESENTATION_DEFINITION_SCHEMA.ANNOTATION_TEXT_OCCURRENCE' IN
TYPEOF(c)) AND
('PRESENTATION_DEFINITION_SCHEMA.EXPLICIT_TEXT_REFERENCE' IN
TYPEOF(c\annotation_text_occurrence.item)));

WHERE
 WR1 : SIZEOF(QUERY(ato <* annotation_text_occurrence | (signal_name_data[1] =
ato\annotation_text_occurrence.item\explicit_text_reference.source))) = 1;
 WR2 : SIZEOF(QUERY(ato <* annotation_text_occurrence | (pin_number_data[1] =
ato\annotation_text_occurrence.item\explicit_text_reference.source))) = 1;
 WR3 : 'PRODUCT_PROPERTY_DEFINITION_SCHEMA.SHAPE_ASPECT_RELATIONSHIP' IN
TYPEOF(SELF\schematic_symbol_callout.presented_concept\item_identified_representation_us
age.definition);
```

```
END_ENTITY;

--These will need to be split apart and placed in the appropriate module MIMs.
TYPE scms_text_reference_source = EXTENSIBLE GENERIC_ENTITY SELECT BASED_ON
text_reference_source WITH (
        product,
        model_parameter,
        parameter_assignment,
        functional_unit_terminal_definition,
        packaged_part_terminal);
END_TYPE;

--following are in presentation_definition_schema;
--ISO 10303-46
(*
TYPE text_reference_source = EXTENSIBLE GENERIC_ENTITY SELECT();--will include
SELECT type in p46.
END_TYPE;

TYPE text_literal_or_text_reference = SELECT(
    text_literal,
    text_reference);
END_TYPE;

ENTITY text_reference -- include in p46
  ABSTRACT SUPERTYPE OF (ONEOF(composite_sequential_text_reference,
explicit_text_reference, implicit_text_reference))
  SUBTYPE OF(representation_item);
END_ENTITY;

ENTITY composite_sequential_text_reference
  SUBTYPE OF (text_reference);
    collected_references : LIST[2:?] of text_literal_or_text_reference;
WHERE
  WR1 : SIZEOF(QUERY( cr <* collected_references | '
PRESENTATION_DEFINITION_SCHEMA.COMPOSITE_TEXT_REFERENCE' IN TYPEOF(cr)))
= 0;
  --no member of collected_references shall be a composite_text_reference.
  --the list is one level only, not an array.
 END_ENTITY;

ENTITY explicit_text_reference-- include in p46
SUBTYPE OF(text_reference);
  attribute_name     : STRING;
  geometric_property : text_literal;
  source             : text_reference_source;
WHERE
  WR1 : geometric_property\text_literal.literal = '';
END_ENTITY;

ENTITY explicit_text_reference_occurrence-- include in p46
SUBTYPE OF(explicit_text_reference);
 type_declaration   : implicit_text_reference;
DERIVE
  SELF\explicit_text_reference.attribute_name : STRING :=
type_declaration\implicit_text_reference.attribute_name;
END_ENTITY;

ENTITY implicit_text_reference -- include in p46
SUBTYPE OF(text_reference);
  attribute_name     : STRING;
  entity_name        : STRING;
  geometric_property : text_literal;
WHERE
  WR1 : geometric_property\text_literal.literal = '';
END_ENTITY;
*)

(*
modified entity in aic_draughting_annotation ISO 10303-504:
ENTITY draughting_annotation_occurrence
          SUBTYPE OF (annotation_occurrence);
WHERE
---...
          WR7 : (NOT('AP
210_ELECTRONIC_ASSEMBLY_INTERCONNECT_AND_PACKAGING_DESIGN_MIM_LF.AN
NOTATION_TEXT_OCCURRENCE' IN TYPEOF (SELF))) OR
               (SIZEOF (TYPEOF(SELF.item) *
                    ['AP
210_ELECTRONIC_ASSEMBLY_INTERCONNECT_AND_PACKAGING_DESIGN_MIM_LF.CO
MPOSITE_TEXT', 'AP
210_ELECTRONIC_ASSEMBLY_INTERCONNECT_AND_PACKAGING_DESIGN_MIM_LF.TEX
T_LITERAL', 'AP
210_ELECTRONIC_ASSEMBLY_INTERCONNECT_AND_PACKAGING_DESIGN_MIM_LF.TEX
T_REFERENCE']) = 1);
END_ENTITY;
*)


END_SCHEMA;
```