

IBIS-X

Extended I/O Buffer Information Specification (IBIS-X) Version 0.6

IBIS-X is a standard for both describing and modeling the characteristics of an integrated circuit's input/output analog characteristics.

=====

TABLE OF CONTENTS

=====

REVISION HISTORY 5

GENERAL INTRODUCTION..... 6

STATEMENT OF INTENT..... 8

GENERAL SYNTAX RULES AND GUIDELINES..... 10

KEYWORD TREE DIAGRAM..... 12

IBIS FILE HEADER 14

Keyword: [Begin Header] 14

Keyword: [IBIS-X Ver] 14

Keyword: [Include Library] 14

Keyword: [File Name] 15

Keyword: [File Rev] 15

Keyword: [Date] 15

Keyword: [Source] 15

Keyword: [Notes] 16

Keyword: [Disclaimer] 16

Keyword: [Copyright] 16

Keyword: [Support] 16

Keyword: [Redistribution] 17

Keyword: [Redistribution Text] 17

Keyword: [End Header] 17

GLOBAL KEYWORDS..... 18

Keyword: [Comment Char] 18

Keyword: [Include] 18

COMPONENT INFORMATION 19

Keyword: [Begin Component] 19

Keyword: [Manufacturer] 19

Keyword: [Package] 20

Keyword: [Pin] 20

Keyword: [Package Model] 21

Keyword: [Pin Mapping] 21

Keyword: [Diff Pin] 23

Keyword: [Series Pin Mapping] 24

Keyword: [Series Switch Groups] 25

Keyword: [Model Selector] 25

Keyword: [End Component] 27

DEFINING NEW MODELS..... 28

Keyword: [Define "class name"] 28

Keyword: [Begin "class name"] 29

Keyword: ["class name" Selector] 31

PACKAGE MODELING..... 32

Keyword: [Define Package Model] 32

Keyword: [End Package Model] 33

ELECTRICAL BOARD DESCRIPTION 34

Keyword: [Begin Board Description] 35

Keyword: [Manufacturer] 35

Keyword: [Number Of Pins] 35

Keyword: [Pin List] 35

Keyword: [Path Description] 36

Keyword: [Reference Designator Map] 40

Keyword: [End Board Description] 40

END KEYWORD..... 42

Keyword: [End] 42

=====
=====

REVISION HISTORY

=====
=====

06/01/-1 - Stephen Peters

- File Rev. 0.6
- Added description of "class name" selector ([Model Selector] keyword) and updated keyword tree
- Moved revision history to just after the TOC and renumbered sections
- Corrected argument and description of [End] keyword
- Editorial/typo corrections, including replacing occurrences of "CAE" with "EDA"

05/15/01 - Stephen Peters

- File Rev. 0.5
- No change, bumped rev to 0.5 for public release

=====
=====

Section 1

GENERAL INTRODUCTION

=====
=====

This section gives a general overview of the remainder of this document.

Section 2 contains general information about IBIS-X and its relationship to IBIS. Section 3 details the general syntax rules and guidelines for creating an IBIS-X file while section 4 presents a tree diagram of the keywords.

A description of the IBIS-X data transfer template follows in sections 5 through 12. Section 5 details the IBIS file header. Section 6 describes global keywords while section 7 details the information contained in the Component section. Section 8 describes how the [Define] keyword is used to create new model types as well as describes a generalized method for selecting among models or other simulation objects. Sections 9 and 10 detail the package and electrical board description extensions to IBIS-X while section 11 describes the file termination keyword [End].

Section 2

STATEMENT OF INTENT

This document describes a new and expanded template for electronically transporting I/O buffer modeling data between semiconductor vendors, simulation vendors, and end customers. This new template is referred to as IBIS-X.

Background:

Current EDA tools developed specifically for signal integrity simulation and analysis apply a set of well defined and understood data to standard simulation algorithms. These algorithms, in turn, are based on a standard I/O buffer model topology. In other words, it is assumed that the I/O buffers used for digital logic can be abstracted to a set of controlled voltage and current sources connected in a more-or-less fixed manner. Given this fixed model, the user needed only to supply specific data for each I/O buffer. This data consists primarily of the DC I/V (output voltage vs. current) characteristics, plus V/T (output voltage vs. time) waveforms that describe an outputs transition from one state to another. In addition, other "data book" type information is included so as to allow for system level timing analysis and error checking.

The original IBIS specification was the first industry standard method for electronically transporting I/O buffer modeling data. However, while the original IBIS specification has obtained wide industry acceptance, it has become increasingly clear that the original paradigm of data applied to a *fixed* model is no longer able to fully meet the needs of the industry. This is due to a number of issues, some of which are:

- 1) The increasing complexity of I/O buffers
- 2) The advent of new signaling technologies
- 3) Simulation methodologies that require describing the input-to-output relationships of a digital receiver, and finally
- 4) The need to accurately model the effects of on-die power distribution, return paths, and pin to pin coupling on an IO pin's behavior.

Therefore, the extended I/O Buffer Information Specification (IBIS-X) is proposed.

Overview of IBIS-X:

IBIS-X provides two fundamental enhancements over the current IBIS 3.2 specification. The first enhancement is the inclusion of a completely new package model description. This description, based on nodal interconnects, allows the user to model such package related effects as on-die power distribution, pin-to-pin coupling, return path discontinuities, etc. This nodal package description also supports connecting two or more I/O pins to an I/O buffer model, thus directly supporting differential drivers/receivers and series elements.

The second enhancement is the definition of a spice-like modeling language referred to as the IBIS Macro Language (IBIS-ML). This macro language, along with a general 'Define'

mechanism, allows the user to create new types of I/O buffer models - models with different functions, abilities, etc. It is this mechanism that allows IBIS-X to support the needs of the industry into the future.

Commitment to Backwards Compatibility:

The IBIS-X specification will be backwards compatible with ANSI/EIA-656-A (IBIS Ver 3.2). This means that all keywords and sub-parameters defined for IBIS 3.2 are legal in an IBIS-X file. This includes the keywords and sub-parameters used by the package and electrical board descriptions extensions. In addition, the ANSI/EIA IBIS Open Forum will supply to the industry a file containing an IBIS-ML description of the model types, submodels and driver schedule functions defined for IBIS 3.2. This IBIS-ML library file will allow IBIS-X type simulators and other EDA tools to read and process IBIS 3.2 models directly.

Section 3

GENERAL SYNTAX RULES AND GUIDELINES

This section contains general syntax rules and guidelines for IBIS-X files:

- 1) Only ASCII characters, as defined in ANSI Standard X3.4-1986, may be used in an IBIS-X file. The use of characters with codes greater than hexadecimal 07E is not allowed. Also, ASCII control characters (those numerically less than hexadecimal 20) are not allowed, except for TAB characters and line termination sequences.
NOTE: the use of TAB characters is strongly discouraged.
- 2) Except for keywords (words enclosed by a square bracket []), the content of an IBIS-X file is case sensitive.
- 3) Keywords must be enclosed in square brackets, [], and must start in column 1 of the line. No space is allowed immediately after the opening bracket '[' or immediately before the closing bracket ']'. If used, only one pace (' ') or underscore ('_') character separates the parts of a multi-word keyword. Spaces and underscores are equivalent within square bracket. Some keywords may be followed by an argument. Keyword arguments are of three types: either a text string, a file name, or a text block. Text string and file name arguments start on the same line as the keyword and are terminated with a line termination sequence. A text block starts on the same line as the keyword, may extend over multiple lines, and is terminated by the occurrence of the next keyword.
- 4) Keywords and sub-parameters must begin with a letter, and shall consist only of alphanumeric characters and the underscore (_). Spaces are not allowed in sub-parameter names.
- 5) Each IBIS-X data file is divided into sections. Each section is delimited by keywords of the form [Begin "section name"] and [End "section name"]. Within each section only specific keywords are legal (note, however, a keyword may be legal in more than one section). Unless otherwise noted, a section's keywords can appear in any order.
- 6) To facilitate portability between operating systems, file names used in the IBIS-X file must only have lower case characters. File names should have a basename followed by a period (.), followed by a file name extension of no more than three characters. There is no length restriction on the basename. The basename and extension must use characters from the following set (space, '\', 0x20 is not included):

```
a b c d e f g h i j k l m n o p q r s t u v w x y z
0 1 2 3 4 5 6 7 8 9 _ -
```

- 7) A line of the file may have at most 120 characters, followed by a line termination sequence. The line termination sequence must be one of the following two sequences: a linefeed character, or a carriage return followed by linefeed character.
- 8) Anything following the comment character is ignored and considered a comment on that line. The default "|" (pipe) character can be changed by the keyword [Comment Char] to any other character. The [Comment Char] keyword can be used throughout the file as desired.
- 9) Valid scaling factors are:

T = tera	k = kilo	n = nano
G = giga	m = milli	p = pico
M = mega	u = micro	f = femto

When no scaling factors are specified, the appropriate base units are assumed. (These are volts, amperes, ohms, farads, henries, and seconds.) The parser looks at only one alphabetic character after a numerical entry; therefore it is enough to use only the prefixes to scale the parameters. However, for clarity, it is allowed to use full abbreviations for the units, (e.g., pF, nH, mA, mOhm). In addition, scientific notation IS allowed (e.g., 1.2345e-12).
- 10) When creating tables describing a buffer's I-V characteristics, all currents are considered positive if they are into a component.
- 11) All temperatures are represented in degrees Celsius.
- 12) Multiple definitions of a specific symbol are illegal. Specifically, multiple definitions (either redundant or alternate) of an object type of the same object class are illegal. Refer to section 9 for more information on object classes and object types.
- 13) All lines that occur before the [Begin Header] keyword or after the [End] keyword shall be treated as comments and ignored by the parser. The intent of this rule is to facilitate the inclusion of HTML tags, revision control headers, etc. at the beginning of the file.

 Section 4

 KEYWORD TREE DIAGRAM

Following is a tree diagram that outlines the scope and hierarchy of the IBIS-X keywords. Note that unless otherwise stated in the keyword description itself, keywords within a section are position independent. Both required and optional keywords are shown. The global keywords [Comment Char] and [Include] are not shown.

```

/-- Start of File
/-- [Begin Header]
|
|  -- [IBIS-X Ver]
|  -- [Include Library]
|  -- [File Name]
|  -- [File Revision]
|  -- [Date]
|  -- [Source]
|  -- [Notes]
|  -- [Disclaimer]
|  -- [Copyright]
|  -- [Support]
|  -- [Redistribution]
|  -- [Redistribution Text]
\-- [End Header]

/-- [Define "class name"]
|-- IBIS-ML statements
\-- [End "class name"]

/-- [Begin Component] (1)
|
|  -- [Manufacturer]
|  -- [Package]
|  -- [Pin]
|  -- [Package Model]
|  -- [Pin Mapping]
|  -- [Diff Pin]
|  -- [Series Pin Mapping]
|  -- [Series Switch Group]
|  -- [Model Selector] (2)
\-- [End Component]

/-- [Begin "class name"]
|-- Model data sets
\-- [End "class name"]

/-- [Define Package Model]
|-- keywords specific to package (.pkg) models
\-- [End Package Model]

/-- [Begin Board Description]
|-- keywords specific to electrical board description (.ebd) models
\-- [End Board Description]

\-- [End]

```

Notes:

1. Multiple component sections ([Begin Component]/[End Component] keyword pairs) are allowed

2. While [Model Selector] is in the component section, its scope is global.

Section 5

IBIS FILE HEADER

The IBIS file header provides revision information as well as general documentation about the IBIS file. The header section begins with the keyword [Begin Header] and ends with the keyword [End Header]. A legal header section includes the required keywords [IBIS-X Version], [File Rev], [File Name] and [Redistribution]. In addition, the optional keywords [Comment Char], [Include Library], [Date], [Source], [Notes], [Disclaimer], [Copyright] [Support], [Redistribution Text] and [Include] are allowed.

```

Keyword:  [Begin Header]
Required: Yes
Argument: None
Description: This keyword denotes the beginning of the file header section.
Usage Rules: [Begin Header] must be the first keyword in any IBIS-X file.
              This keyword may be preceded by lines of general text and/or
              comments, which should be treated as comments by the parser.
    
```

[Begin Header]

```

Keyword:  [IBIS-X Ver]
Required: Yes
Argument: Text String
Description: Specifies the IBIS-X template version
Usage Rules: [IBIS-X Ver] specifies the version of the IBIS-X specification
              that was used to create the data file. This keyword must
              appear directly after the [Begin Header] keyword.
    
```

[IBIS-X Ver] 0.21 | Used for template variations

```

Keyword:  [Include Library]
Required: No
Argument: File Name
Description: Specifies the file name of a file containing definitions for
              symbols (object classes and/or object types) not found in the
              current file.
Usage Rules: If an IBIS-X file references object types that are not
              defined in the current file, then use this keyword to inform
              the simulation tool where to look for these additional symbols.
              Only one file name argument is allowed per [Include Library]
              keyword. However, the [Include Library] keyword can be used
              any number of times within the header section.
    
```

Note: When including an IBIS-X library file be sure that the included file does not redefine an already existing symbol. For example, if the current file defines a model type 'my_buffer' (e.g. contains the statement "[Define Model] my_buffer"), including a file that also defines (either alternately or redundantly) the model type 'my_buffer' is illegal.

Because the use of [Include Library] is restricted to the header section any symbols included are valid throughout the rest of the file. It is legal to treat the [Include Library] keyword as a textual include that is delayed until after the header section has been parsed.

 [Include Library] ibis3_2.ibs
 [Include Library] MySpecialBuffer.ibs

=====
 Keyword: [File Name]
 Required: Yes
 Argument: Text String (file name)
 Description: Specifies the name of the IBIS-X file.
 Usage Rules: The purpose of this keyword is to document a single filename the file uses even if the file is transferred from one computer to another. The file name must conform to the rules given in section 3, "General Syntax Rules and Guidelines".

 [File Name] IBIS_X.ibs

=====
 Keyword: [File Rev]
 Required: Yes
 Argument: Text String
 Description: Tracks the revision level of a particular .ibs file.
 Usage Rules: Revision level is set at the discretion of the engineer defining the file. The following guidelines are recommended:
 0.x silicon and file in development
 1.x pre-silicon file data from silicon model only
 2.x file correlated to actual silicon measurements
 3.x mature product, no more changes likely

 [File Rev] 0.9 | Used for .ibs file variations

=====
 Keyword: [Date]
 Required: No
 Argument: Text String (Date)
 Description: Date this file was last modified.
 Usage Rules: This keyword is provided to insure the last changed date for this file is not lost if the file is transmitted between computer systems.

 [Date] February 25, 2001

=====
 Keyword: [Source]
 Required: No
 Argument: Text Block
 Description: Records the originating source of model data.
 Usage Rules: Use this keyword to record how the model information was obtained (physical measurement of device, transistor level simulations, data book, etc.).

NOTE: It is recommended that the argument to the [Source] keyword be limited to a maximum of 24 lines of text.

 [Source] From silicon level SPICE model
 From lab measurement

Compiled from manufacturer's data book.

```

=====
Keyword:  [Notes]
Required: No
Argument: Text Block
Description: Optional notes regarding the file.
Usage Rules: The keyword provides a place for the model maker to record
              important notes about the file or model data that are not
              included elsewhere.  Such information may include notes on
              validation level, model limits, usage assumptions, etc.

```

NOTE: It is recommended that the argument to the [Notes] keyword be limited to a maximum of 24 lines of text.

The [Notes] keyword can only be used once.

```

-----
[Notes]   This file is an example IBIS-X file.

```

```

=====
Keyword:  [Disclaimer]
Required: No
Argument: Text Block
Description: Legal disclaimer information
Usage Rules: This keyword provides a place for the user to add a legal
              disclaimer.

```

NOTE: It is recommended that the argument to the [Disclaimer] keyword be limited to a maximum of 96 lines of text.

```

-----
[Disclaimer] This information is for modeling purposes only, and is not
              guaranteed.

```

```

=====
Keyword:  [Copyright]
Required: No
Argument: Text Block
Description: Copyright information
Usage Rules: Because IBIS-X model writers may consider the information in
              these keywords essential to users, and sometimes legally
              required, design automation tools should make this information
              available. Any text following the [Copyright] keyword must be
              included in any derivative models verbatim.

```

NOTE: It is recommended that the argument to the [Copyright] keyword be limited to a maximum of 96 lines of text.

```

-----
[Copyright] Copyright 2000, XYZ Corp., All Rights Reserved

```

```

=====
Keyword:  [Support]
Required: No
Argument: Text Block (URL path name)
Description: Specifies a web site that can be visited to get the latest
              version of the file.
Usage Rules: Following the [Support] keyword is the URL of a web site the
              user may visit for more information on the model or model(s).
              The entire link, including any file extension, is required. The
              [Support] keyword may appear only once between the [Begin
              Header]/[End Header] keyword pair.

```

NOTE: It is recommended that a separate IBIS model web site (not page) be maintained to prevent accidental changes of page name from breaking this link.

 [Support] <http://www.VendorName.com/IbisModels> |example ibis web site root

=====
 Keyword: [Redistribution]
 Required: Yes
 Argument: Text String (Yes, No, Specific)
 Description: Indicates to EDA tool companies and model users who may use and redistributed this file.
 Usage Rules: Following the [Redistribution] keyword is one of three arguments: "Yes", "No" and "Specific".

An argument value of "Yes" means that a EDA tool vendor or end user may freely distribute the model as long as no fee is charged. A fee may be charged if authorized by the model creator.

An argument value of "No" means that the model may not be redistributed or retransmitted in any form.

An argument value of "Specific" means that the [Redistribution Text] keyword contains specific information on licensing details or where to find them. Use of the "Specific" argument prevents automated redistribution.

 [Redistribution] Yes

=====
 Keyword: [Redistribution Text]
 Required: No, unless the argument to the [Redistribution] keyword is 'Specific'.
 Argument: Text block
 Description: Provides a place for the user to document license details.
 Usage Rules: If the argument to the [Redistribution] keyword is 'Specific', then the user must include additional information on licensing details, or where to find them. This keyword provides a place for this information.

Note: it is recommended that the argument to the [Redistribution Text] keyword be limited to a maximum of 24 lines of text.

 [Redistribution Text] Contact company for latest license agreement

=====
 Keyword: [End Header]
 Required: Yes
 Argument: None
 Description: Marks the end of an IBIS-X header section.
 Usage Rules: [End Header] must be the last keyword in an IBIS-X header section.

 [End Header]

=====
 =====
 Section 6

GLOBAL KEYWORDS

=====
 =====
 The keywords [Comment Char] and [Include] are global keywords. A global keyword is a keyword that can, subject to any restrictions outlined in the keyword descriptions themselves, be used as required throughout an IBIS-X file.

=====
 =====
 Keyword: [Comment Char]
 Required: No
 Argument: Text String
 Description: Defines a new comment character to replace the default
 "|" (pipe) character.
 Usage Rules: The new comment character to be defined must be followed by
 the underscore character and the letters "char". For example:
 "|_char" redundantly redefines the comment character to be
 the pipe character. The new comment character is in effect
 only following the [Comment Char] keyword. The following
 characters MAY be used:

! " # \$ % & ' () * , : ; < > ? @ \ ^ ` { | } ~

If used, the [Comment Char] keyword can only appear after the
 [Begin Header] and [IBIS-X Ver] keywords. A file may contain
 multiple [Comment Char] keywords.

 [Comment Char] |_char

=====
 =====
 Keyword: [Include]
 Required: No
 Argument: File Name
 Description: Include the contents of the named file.
 Usage Rules: This keyword forces the contents an external file to be
 inserted directly at the point the [Include] keyword is
 used. This keyword is a global keyword (i.e. is not restricted
 to any one section of an IBIS-X file) and may be used throughout
 the file as desired. Note however that the first [Include] must
 occur after the [Begin Header] and [IBIS-X Ver] keywords.
 Only one file name argument is allowed per [Include] keyword.

Rules for included files:
 If the included file contains a [Begin "section"] keyword it
 must also contain the corresponding [End "section"] keyword.
 Obviously, any keywords used in the included file must be
 appropriate for the section in which it is included.

Nested includes (i.e. an included file containing an [Include]
 keyword) is legal, subject to the scoping rule above.

 [Include] Release_notes.txt

Section 7

COMPONENT INFORMATION

The purpose of the component section is two fold. First, this section provides some general information (component name, manufacturer, and so on) about the component(s) the IBIS-X file is describing. Secondly, this section contains a listing of the component pins plus a default package description that specifies which I/O buffer models are connected to which package pins.

A component description begins with the keyword [Begin Component] and ends with the keyword [End Component]. More than one component description may be included in an IBIS-X file, but each individual component description must be contained between a [Begin Component]/[End Component] keyword pair. Positioned between these two keywords are the required keywords [Manufacturer], [Package] and [Pin]. Optional keywords (retained to support backwards compatibility to earlier IBIS versions) include [Package Model], [Pin Mapping], [Diff Pin], [Series Pin Mapping], [Series Switch Group] and [Model Selector].

```

Keyword: [Begin Component]
Required: Yes
Argument: Text String (Component Name)
Description: Marks the beginning of the description of the component named
             by this keyword.
Sub-Params: Si_location, Timing_location
Usage Rules: Each component description contained in an IBIS-X file must
             begin with the [Begin Component] keyword, followed by the
             name of the component. The component name may include blank
             characters, however, they are not recommended due to usability
             issues.
    
```

Si_location and Timing_location are optional sub-parameters that specify where the signal integrity (SI) related and/or timing related measurements are made for the component. Allowed values for either sub-parameter are 'Die' or 'Pin'. The default value of both sub-parameters is 'Pin'.

```

[Begin Component]      MC74ALS257
Si_location      Pin   | Optional sub parameters to give measurement
Timing_location  Die   | location positions
    
```

```

Keyword: [Manufacturer]
Required: Yes
Argument: Text String (Manufactures Name)
Description: Specifies the manufacturer's name of the component.
Usage Rules: Blank characters are allowed.
    
```

NOTE: it is highly recommended that each manufacturer use a consistent name in all .ibs files.

```

[Manufacturer] Intel Corp.
    
```

```

=====
Keyword:  [Package]
Required: Yes
Argument: None
Description: Defines a default range of values for the component package pin
            inductance, capacitance and resistance.
Sub-Params: R_pkg, L_pkg, C_pkg
Usage Rules: Following the [Package] keyword the three sub-parameters are
            listed, one per line. Each sub-parameter is followed by the
            typical, minimum and maximum value for that sub-parameter,
            listed in that order. A typical (typ) value must be specified.
            If data for the other columns are not available then "NA" must
            be entered into that column. The numerically smallest value of
            L_pkg, R_pkg or C_pkg must be placed in the minimum (min) column
            while the numerically largest value must be placed in the
            maximum (max) column.

Other Notes: R/L/C values for individual pins can be specified by the [Pin]
            keyword and those values will override the values listed by the
            R_pkg, L_pkg and C_pkg sub-parameters.

```

NOTE: Even if the data given in this keyword is superceded by detailed pin parasitic data given elsewhere, it is highly recommended that the [Default Package] keyword still contain valid data.

```

-----
[Package]
| Variable      typ          min          max
R_pkg          250.0m      225.0m      275.0m
L_pkg          15.0nH      12.0nH      18.0nH
C_pkg          18.0pF      15.0pF      20.0pF

```

```

=====
Keyword:  [Pin]
Required: Yes
Argument: None
Description: Lists the component's pins and maps them to a specific I/O
            buffer model.
Sub-Params: signal_name, model_name, R_pin, L_pin, C_pin
Usage Rules: All pins on a component must be specified. The first column
            under the [Pin] keyword lists the pin name. The second column,
            signal_name, lists the name of the signal on that pin as given
            in the component's data sheet. The third column, model_name,
            contains either the name of a specific I/O buffer
            model, the name of a model selector as listed in a [Model
            Selector] keyword or the designation "POWER", "GND" or "NC".
            The purpose of the model_name column is to document which I/O
            buffer model is attached to which component pins or,
            alternately, which component pins connect to external power
            (POWER), ground (GND) or are internally unconnected (NC). Note
            that a specific I/O model may be mapped to more than one pin.
            The pin, signal_name and model_name columns are required.

```

Following the required three columns are three optional columns. The values listed in these columns override, on a pin by pin bases, the R_pkg, L_pkg and C_pkg values given in the [Package] keyword. When these columns are included the column heading R_pin, L_pin, and C_pin must be present, although the columns can be put in any order. While R_pin, L_pin and C_pin are optional data, if any one column is present then all columns must be present.

Any one of the entries under the R_pin, L_pin and C_pin columns can use the value of "NA". If used, that pin uses the default L, R or C value listed under the [Package] keyword.

Other Notes: The [Pin] keyword is directly analogous to the [Pin] keyword in earlier IBIS versions, and is included to maintain backwards compatibility.

[Pin]	signal_name	model_name	R_pin	L_pin	C_pin
1	RAS0#	Buffer1	200.0m	5.0nH	2.0pF
2	RAS1#	Buffer2	209.0m	NA	2.5pF
3	EN1#	Input1	NA	6.3nH	NA
4	A0	3-state			
5	D0	I/O1			
6	RD#	Input2	310.0m	3.0nH	2.0pF
7	WR#	Input2			
8	A1	I/O2			
9	D1	I/O2			
10	GND	GND	297.0m	6.7nH	3.4pF
11	RDY#	Input2			
12	GND	GND	270.0m	5.3nH	4.0pF
.					
.					
18	Vcc3	POWER			
19	NC	NC			
20	Vcc5	POWER	226.0m	NA	1.0pF

```

=====
Keyword: [Package Model]
Required: No
Argument: Text String (name of package model)
Description: Indicates the name of the package model to be used for the
component.
Usage Rules: Spaces are allowed in a package model name. A simulator will
search for a matching package model name as an argument to a
[Define Package Model] keyword in the current IBIS-X file
first. If a match is not found, the simulator will next look
for a match in an external .pkg file. If the matching package
model is in an external .pkg file, it must be located in the
same directory as the .ibs file. The file names of .pkg files
must follow the rules for file names given in section 3,
General Syntax Rules and Guidelines.

```

NOTE: to insure uniqueness, it is highly recommended that the company name or initials be included in the package model name.

Other Notes: Use the [Package Model] keyword within a [Component] to indicate which package model should be used for that component. The specification permits .ibs files to contain a [Define Package Model] keywords as well. When package model definitions occur within an .ibs file, their scope is "local"; they are known only within that .ibs file and no other. In addition, within that .ibs file they override any globally defined package models that have the same name.

```

-----
[Package Model]      QS-SMT-cer-8-pin-pkgs

```

```

=====
Keyword: [Pin Mapping]
Required: No
Argument: None

```

Description: Used to define a components internal power and ground busses as well as document which I/O buffer models share common power and ground busses.

Sub-Params: pulldown_ref, pullup_ref, gnd_clamp_ref, power_clamp_ref
 Usage Rules: Following the [Pin Mapping] keyword are either three or five columns. The first column contains a pin name. Each pin name must match one of the pin names declared previously in the [Pin] keyword. The second column, pulldown_ref, contains the names of the component's internal ground buss(es). The third column, pullup_ref, contains the names of the component's power buss(es). Unless specified otherwise in a package description, ground busses are assumed to connect to a model's pulldown structure while power busses are assumed to connect to a model's pullup structure. The fourth and fifth columns, gnd_clamp_ref and power_clamp_ref are optional. They contain entries that specify the busses available for connection to a model's gnd_clamp and power_clamps.

If the [Pin Mapping] keyword is present, then the bus connections for EVERY pin listed in the [Pin] section must be given.

Each line must contain either three or five columns. Use the designation "NC" for entries that are not needed.

All entries with identical labels are assumed to be connected. Each unique entry label must connect to at least one pin whose model_name is POWER or GND.

If a pin has no connection, then both the pulldown_ref and pullup_ref sub-parameters for it will be "NC".

GND and POWER pin entries and busses are designated by entries in either the pulldown_ref or pullup_ref columns. There is no implied association to any column other than through explicit designations in other pins.

For any other type of pin, the pulldown_ref column contains the power connection for the [Pulldown] table for non-ECL type I/O buffer models. This is also the power connection for the [GND Clamp] table and the [Rgnd] model unless overridden by a specification in the gnd_clamp_ref column.

Also, the pullup_ref column contains the power connection for the [Pullup] table and, for ECL type models, the [Pulldown] table. This is also the power connection for the [POWER Clamp] table and the [Rpower] model unless overridden by a specification in the power_clamp_ref column.

When 5 columns are specified, the headings gnd_clamp_ref and power_clamp_ref must be used. Otherwise, these headings can be omitted.

[Pin Mapping]	pulldown_ref	pullup_ref	gnd_clamp_ref	power_clamp_ref
1	GNDBUS1	PWRBUS1	Signal pins and their associated	
2	GNDBUS2	PWRBUS2	ground and power connections	
3	GNDBUS1	PWRBUS1	GNDCLMP	PWRCLAMP
4	GNDBUS2	PWRBUS2	GNDCLMP	PWRCLAMP
5	GNDBUS2	PWRBUS2	NC	PWRCLAMP
6	GNDBUS2	PWRBUS2	GNDCLMP	NC
			Some possible clamping connections	
:			are shown above for illustration	
:			purposes	

11	GNDBUS1	NC	One set of ground connections.
12	GNDBUS1	NC	NC indicates no connection to
13	GNDBUS1	NC	power bus.
21	GNDBUS2	NC	Second set of ground connections
22	GNDBUS2	NC	
23	GNDBUS2	NC	
31	NC	PWRBUS1	One set of power connections.
32	NC	PWRBUS1	NC indicates no connection to
33	NC	PWRBUS1	ground bus.
41	NC	PWRBUS2	Second set of power connections
42	NC	PWRBUS2	
43	NC	PWRBUS2	
51	GNDCCLMP	NC	Additional power connections
52	NC	PWRCLMP	for clamps

```

=====
Keyword: [Diff Pin]
Required: No
Argument: None
Description: Associates differential pins, their differential threshold
             voltages, and differential timing delays.
Sub-Params: inv_pin, vdiff, tdelay_typ, tdelay_min, tdelay_max
Usage Rules: Enter only differential pin pairs. The first column, [Diff
             Pin], contains a non-inverting pin name. The second column,
             inv_pin, contains the corresponding inverting pin name for
             I/O output. Each pin name must match the pin names declared
             previously in the [Pin] section of the IBIS file. The third
             column, vdiff, contains the specified differential
             threshold voltage between pins if the pins are Input or I/O
             model types. For output only differential pins, the vdiff
             entry is 0 V. The fourth, fifth, and sixth columns,
             tdelay_typ, tdelay_min, and tdelay_max, contain launch delays
             of the non-inverting pins relative to the inverting pins. The
             values can be of either polarity.

             If a pin is a differential input pin, the differential input
             threshold (vdiff) overrides and supersedes the need for Vinh
             and Vinl.

             If vdiff is not defined for a pin that is defined as requiring
             a Vinh by its [Model] type, vdiff is set to the default value
             of 200 mV.

Other Notes: The output pin polarity specification in the table overrides
             A models Polarity specification such that the pin in the
             [Diff Pin] column is Non-Inverting and the pin in the inv_pin
             column is Inverting. This convention enables one model to
             be used for both pins.

             Each line must contain either four or six columns. If "NA" is
             entered in the vdiff, tdelay_typ, or tdelay_min columns, its
             entry is interpreted as 0 V or 0 ns. If "NA" appears in the
             tdelay_max column, its value is interpreted as the tdelay_typ
             value. When using six columns, the headers tdelay_min and
             tdelay_max must be listed. Entries for the tdelay_min column
             are based on minimum magnitudes; and tdelay_max column,
             maximum magnitudes. One entry of vdiff, regardless of its
             polarity, is used for difference magnitudes.
=====
    
```

[Diff Pin]	inv_pin	vdiff	tdelay_typ	tdelay_min	tdelay_max	
3	4	150mV	-1ns	0ns	-2ns	Input or I/O pair
7	8	0V	1ns	NA	NA	Output* pin pair
9	10	NA	NA	NA	NA	Output* pin pair
16	15	200mV	1ns			Input or I/O pin pair
20	19	0V	NA			Output* pin pair, tdelay = 0
22	21	NA	NA			Output*, tdelay = 0

* Could be Input or I/O with vdiff = 0

=====
 Keyword: [Series Pin Mapping]
 Required: No
 Argument: None
 Description: Used to associate two pins joined by a series model.
 Sub-Params: pin_2, model_name, function_table_group
 Usage Rules: Enter only series pin pairs. The first column, [Series Pin Mapping], contains the series pin for which input impedances are measured. The second column, pin_2, contains the other connection of the series model. Each pin must match the pin names declared previously in the [Pin] section of the IBIS file. The third column, model_name, associates the Series or Series_switch model for the pair of pins in the first two columns. The fourth column, function_table_group, contains an alphanumeric designator string to associate those sets of Series_switch pins that are switched together.

Each line must contain either three or four columns. When using four columns, the header function_table_group must be listed.

One possible application is to model crossbar switches where the straight through On paths are indicated by one designator and the cross over On paths are indicated by another designator. If the model referenced is a Series model, then the function_table_group entry is omitted.

Other Notes: If the model_name is for a non-symmetrical series model, then the order of the pins is important. The [Series Pin Mapping] and pin_2 entries must be in the columns that correspond with Pin 1 and Pin 2 of the referenced model.

This mapping covers only the series paths between pins. The package parasitics and any other elements such as additional capacitance or clamping circuitry are defined by the model_name that is referenced in the [Pin] keyword. The model_names under the [Pin] keyword that are also referenced by the [Series Pin Mapping] keyword may include any legal model or reserved model except for Series and Series_switch models. Normally the pins will reference a model whose type is 'Terminator'. For example, a Series_switch model may contain Terminator models on EACH of the pins to describe both the capacitance on each pin and some clamping circuitry that may exist on each pin. In a similar manner, Input, I/O or Output models may exist on each pin of a Series model that is serving as a differential termination.

[Series Pin Mapping]	pin_2	model_name	function_table_group	
2	3	CBTSeries	1	Four independent groups
5	6	CBTSeries	2	
9	8	CBTSeries	3	
12	11	CBTSeries	4	

22	23	CBTSeries	5	Straight through path
25	26	CBTSeries	5	
22	26	CBTSeries	6	Cross over path
25	23	CBTSeries	6	
32	33	Fixed_series		No group needed

=====

Keyword: [Series Switch Groups]
 Required: No
 Argument: None
 Description: Used to define allowable switching combinations of series switches described using the names of the groups in the [Series Pin Mapping] keyword function_table_group column
 Sub-Params: On, Off
 Usage Rules: Each state line contains an allowable configuration. A typical state line will start with 'On' followed by all of the on-state group names or an 'Off' followed by all of the off-state group names. Only one of 'On' or 'Off' is required since the undefined states are presumed to be opposite of the explicitly defined states. The state line is terminated with the slash '/' character.

The group names in the function_table_group are used to associate switches whose switching action is synchronized by a common control function. The first line defines the assumed (default) state of the set of series switches. Other sets of states are listed and can be selected through a user interface or through automatic control.

```
[Series Switch Groups]
| Function Group States
On 1 2 3 4 / | Default setting is all switched On.
|
Off 1 2 3 4 / | All Off setting.
On 1 / | Other possible combinations below.
On 2 /
On 3 /
On 4 /
On 1 2 /
On 1 3 /
On 1 4 /
On 2 3 /
On 2 4 /
On 3 4 /
On 1 2 3 /
On 1 2 4 /
On 1 3 4 /
On 2 3 4 /
| Off 4 / | The last four lines above could have been replaced
| Off 3 / | with these four lines with the same meaning.
| Off 2 /
| Off 1 /
On 5 / | Crossbar switch straight through connection
On 6 / | Crossbar cross over connection
Off 5 6 / | Crossbar open switches
|
```

=====

Keyword: [Model Selector]
 Required: No
 Argument: Text String (Model Selector group name)
 Description: Used to pick a model from a list of model(s) for a pin which uses a programmable buffer.

Usage Rules: A programmable buffer must have a model defined for each one of its models used in the .ibs file. The names of these models must be unique and are listed under the [Model Selector] keyword and/or pin list. The argument of the [Model Selector] keyword must match a corresponding model name listed under the [Pin] or [Series Pin Mapping] keyword. An .ibs file must contain enough [Model Selector] keywords to cover all of the model selector names specified under the [Pin] and [Series Pin Mapping] keywords.

The section under the [Model Selector] keyword must have two fields. The two fields must be separated by at least one white space. The first field lists the [Model] name. The second field contains a short description of the model shown in the first field. The contents of the description is not specified. The purpose of the descriptions is to aid the user of the simulator tool in making intelligent buffer mode selections and it can be used by the simulator tool in a user interface dialog box as the basis of an interactive buffer selection mechanism.

The first entry under the [Model Selector] keyword shall be considered the default by the simulator tool for all those pins which call this [Model Selector].

The operation of this selection mechanism implies that a group of pins which use the same programmable buffer (i.e. model selector name) will be switched together from one model to another. Therefore, if two groups of pins, for example an address bus and a data bus, use the same programmable buffer, and the user must have the capability to configure them independently, one can use two [Model Selector] keywords with unique names and the same list of models; however, the usage of the [Model Selector] is not limited to these examples. Many other combinations are possible.

Note: While the [Model Selector] keyword appears in a component section, it references [Models] that are in the top level scope of an .ibs file. Therefore, this keyword is considered to have a global scope. As a result, if the .ibs file has multiple component sections that include [Model Selector] keywords then the argument for each [Model Selector] must be unique.

```

-----
|
|[Pin]   signal_name   model_name   R_pin   L_pin   C_pin
|
| 1     RAS0#         Progbuffer1  200.0m  5.0nH   2.0pF
| 2     EN1#         Input1      NA       6.3nH   NA
| 3     A0           3-state
| 4     D0           Progbuffer2
| 5     D1           Progbuffer2  320.0m  3.1nH   2.2pF
| 6     D2           Progbuffer2
| 7     RD#         Input2      310.0m  3.0nH   2.0pF
|
| .
| .
| .
| 18    Vcc3         POWER
|
|[Model Selector]   Progbuffer1
|
|OUT_2   2 mA buffer without slew rate control
|OUT_4   4 mA buffer without slew rate control
|OUT_6   6 mA buffer without slew rate control

```

```
OUT_4S      4 mA buffer with slew rate control
OUT_6S      6 mA buffer with slew rate control
|
[Model Selector]      Progbuffer2
|
OUT_2       2 mA buffer without slew rate control
OUT_6       6 mA buffer without slew rate control
OUT_6S      6 mA buffer with slew rate control
OUT_8S      8 mA buffer with slew rate control
OUT_10S     10 mA buffer with slew rate control
```

```
=====
Keyword:    [End Component]
Required:   Yes
Argument:   Text String (component name)
Description: Marks the end of the description of the component named by
             the corresponding [Begin Component] keyword.
Usage Rules: [End Component] must be the last keyword in any IBIS-X
             component section.
=====
```

```
[End Component] MC74ALS257
```

Section 8

DEFINING NEW MODELS

This section of the specification describes how to create new model types using a general purpose define mechanism. More formally, the specification allows the user to define new 'classes' of objects, as well as new 'types' of objects within a class. In this specification an 'object' most commonly represents an I/O buffer model. However, new objects need not be limited to new types of models. The define mechanism can be used to define a circuit used to test an I/O buffer, a subcircuit used in a package description, or even create a brand new keyword.

Defining a model (or, more formally, creating an object) is a two step process. In the first step, the [Define] keyword is used to create a general purpose structural and/or behavioral description of a model. Note that this description is parameterized (i.e., contains placeholders for specific data values). In essence, the description is a template that describes the structure and behavior of any model (object) of that particular class and type. Second, once the general purpose template has been created, the user supplies a set of data value(s) for each parameter in the template. It is only after data has been supplied to the model template can a specific model be considered fully defined and ready to be used by a simulation tool.

Creating an Object Template

The [Define] keyword is used to create an object template. This process is described below:

```

Keyword:  [Define "class name"]
Required: No
Argument: Text String (class type)
Description: This keyword is used to define a new class of object or a new
type within a class.
Usage Rules: A type or class definition begins with an opening square
bracket ([), followed by the literal 'Define', followed by the
name of the class, followed by a closing square bracket (]).
This is followed by the name of the type of object within the
class and, optionally, a list of comma separated external
connection ports enclosed in parentheses. All types within a
class must use the same port list. The actual structure and
function of the object type is then defined using IBIS-X macro
language (IBIS-ML) constructs. A type or class name
definition is terminated by the [End "class name"] keyword. The
"type name" following the [End "class name"] keyword is
required.

Example:
[Define "class name"] "type name" (optional port list)
|
| IBIS-ML constructs as defined by the IBIS-X Macro Language
| Reference Manual.
|
[End "class name"] "type name"

```

For the purposes of this specification, objects that share

common characteristics or purposes are grouped together in a 'class', while each object within a class is referred to as a 'type'.

As stated in the general syntax rules, multiple definitions of symbols are prohibited. In particular, once an object type and class have been defined, they may not be redefined anywhere in the current .ibs file or any included files.

The [Define]/[End] keywords are global in scope, and can be used anytime after the [End Header] keyword has been encountered. A class and object type must be defined before an object of that class and type is customized (see the following paragraphs regarding "Customizing an Object"). The object/type definition can be contained in an IBIS library referenced by the [Include Library] keyword or in the .ibs file itself.

```
-----
[Define Model] my_type (in, out, pwr, gnd)
|
| IBIS-ML constructs as defined by the IBIS-X Macro Language Reference
| Manual
|
[End Model] my_type
=====
```

Supplying Data To An Object Template:

As mentioned above, when creating a new object template IBIS-ML constructs are used to describe the object's structure and behavior. However, the IBIS-ML description is a general description with contains references to symbolic names. In effect, each object description is parameterized, thus allowing the user to supply a unique data set to a specific copy of an object template. Supplying a data set to a specific copy of a template is referred to as *customizing an object*. Customizing an object is the second step in creating a specific object. The syntax for customizing an object is shown below:

```
-----
Keyword: [Begin "class name"]
Required: No
Argument: Text String (object name)
Description: This keyword is used to customize (supply data to) an object
              created by a previous [Define "class name"] keyword.
Usage Rules: An object customization begins with an opening square bracket
              followed by the literal 'Begin', followed by the class name of
              the object, followed by a closing square bracket. This is
              followed by an object name which identifies the specific object
              being created. On the next line is the object type declaration,
              which consists of the "class name" appended with the literal
              '_type', followed by the type name. Together, these two lines
              identify a unique object of a particular class and type.
              This is followed by the data specific to that object. An object
              customization is terminated with the [End "class name"]
              construct.
```

Example:

```
[Begin "class name"] "object name"
"class name"_type "type name"
|
| optional data specific to this object
```

[End "class name"]

Note that it is legal to create an object template that is not parameterized and thus does not need to be customized. However, to create a specific useable object an object customization (sans data) is still required to be performed on the template.

An object's class and type template must be created (i.e. the [Define "class name"] keyword must be encountered) before the template is customized and an object created.

```
-----
[Begin Model] abc
Model_type my_type
|
| Optional data specific to this object
|
[End Model]
```

Example:

Following is an example showing the creation of an object (model) called 'data_receiver'. This model is a member of the class of objects called 'Receiver', and is of the type called 'differential'.

First, the [Define] keyword is used to create a template that describes the structure and operation of the user's differential receivers.

```
[Define Receiver] differential (in_pos, in_neg, output, pwr, gnd)
|
| IBIS-ML constructs
|
[End Receiver] differential
```

Secondly, supplying a specific set of data to the general template for differential receivers creates the specific model 'data_receiver':

```
[Begin Receiver] data_receiver
Receiver_type differential
|
| data set for this model
|
[End Receiver]
```

Now suppose the user wants to model another differential receiver that has slightly different characteristics than the first receiver. Since the general purpose template already exists, all the user has to do to create a new model is supply a second set of data to the template and give it a new object name.

```
[Begin Receiver] data_receiver_alt
Receiver_type differential
|
| data set for this model
|
[End Receiver]
```

The user now has two different models - `data_receiver` and `data_receiver_alt` - and both are based on the same template. The user can create an unlimited number of models from the same template, subject only to the physical limits of the EDA tools themselves.

Pre-Defined Objects:

In order to support the backwards compatibility requirements of IBIS-X, an IBIS-X capable simulator will supply an include file that defines the IBIS 3.2 classes [Model] and [Submodel], as well as the various types associated with these classes. Refer to the *IBIS-X Library Guide* for specific details on how to use these pre-defined [Model] and [Submodel] classes and types.

Class Name Selector:

The [Model Selector] keyword documented in the component section is a specific example of a *class name selector*. A class name selector construct allows the file creator to collect a set of objects of the same class into a group with a unique name. Then, wherever an object of that class is used or needed, the file creator can instead substitute the group name, thereby giving the user (thru an EDA tools user interface) the ability to select one out of a set of objects. The formal definition of a class name selector is given below.

```

=====
Keyword: ["class name" Selector]
Required: No
Argument: Text String (group name)
Description: This keyword is used to organize a collection of objects into
a group which can then be referenced wherever objects of that
class are required.
Usage Rules: A class name selector begins with an opening square bracket,
followed by the class name of the objects being grouped,
followed by the literal 'Selector' and a closing square bracket.
Following the ["class name" Selector] line are two columns of
data. The first column contains the object's name. The second
column contains a short description of that object. The intent
of the description is to aid the user in understanding which
object to select. This description must fit on the same line as
the objects name.

All objects listed under the ["class name" Selector] must be of
the same class, and must exist in the .ibs file. In the event
that an EDA tool does not support this construct or the user
does not make a choice, the first entry listed under ["class
name" Selector] is considered the default object, and will be
used wherever the group name appears.

For a complete example refer to the [Model Selector] keyword
description in the Component Information section of this
specification.
=====

```

Section 9

PACKAGE MODELING

As mentioned in section 8 of this specification, a user may, thru the use of the [Package Model] keyword, reference an alternate package model (one other than the default RLC pin parameters). This section of the specification defines the keywords associated with the IBIS-X nodal-description based package model.

Note: As of Rev 0.5, the definition of the IBIS-X nodal-description based package model is a work in progress; only the introductory and termination keywords of the IBIS-X package model have been documented in this release. A complete description of this package model format will be included in later revisions of this specification.

A package models can be in a separate file or can exist in the IBIS-X .ibs file between the [Define Package Model]... [End Package Model] keyword(s) (more than one instance of a [Define Package Model]/[End Package Model] keyword pair is allowed). Simulators that do not support these keywords will ignore all entries between the [Define Package Model] and [End Package Model] keywords.

When package model definitions occurs within an .ibs file, their scope is "local" -- they are known only within that .ibs file and no other. In addition, within that .ibs file, they override any globally defined package models that have the same name.

In order to maintain backwards compatibility, the IBIS-X specification shall also include, by reference, the package model keywords and constructs as defined by the IBIS 3.2 specification.

USAGE RULES FOR THE .PKG FILE:

A package model may be stored in a separate file with a '.pkg' extension as shown below:

<basename>.pkg

If the package model is stored in a separate .pkg file, the file <basename> must adhere to the General Syntax Rules. The file itself must adhere to the General Syntax rules, begin with an IBIS-X header section as defined in this specification and must be terminated with the [End] keyword. Note that any IBIS keywords not explicitly listed or referenced by this section of the specification are prohibited in a .pkg file. The .pkg file is for package models only.

Keyword:	[Define Package Model]
Required:	Yes
Argument:	Text String (package model name)
Description:	Marks the beginning of a package model description.
Usage Rules:	If a .pkg or .ibs file contains data for more than one package, each description must begin with a new [Define Package Model] keyword. For every package model name defined under the [Package Model] keyword, there must be a matching [Define

Package Model] keyword.

Note: While the package model name may include blank characters, they are not recommended due to usability issues.

[Define Package Model] QS-SMT-cer-8-pin-pkgs

=====
Keyword: [End Package Model]
Required: Yes
Argument: None
Description: Marks the end of a package model description.
Usage Rules: This keyword must come at the end of each complete package model description.

 Optionally, a comment may be added after the [End Package Model] keyword to clarify which board model has ended.

[End Package Model] | QS-SMT-cer-8-pin-pkgs

=====
=====
Section 10ELECTRICAL BOARD DESCRIPTION
=====

A "board level component" is the generic term used to describe a printed circuit board (PCB) or substrate which contains components or even other boards, and which can connect to another board through a set of user visible pins. The electrical connectivity of such a board level component is referred to as an "Electrical Board Description". For example, a SIMM module is a board level component that is used to attach several DRAM components on the PCB to another board through edge connector pins. An electrical board description file (a .ebd file) is defined to describe the connections of a board level component between the board pins and its components on the board.

A fundamental assumption regarding the electrical board description is that the inductance and capacitance parameters listed in the file are derived with respect to well-defined reference plane(s) within the board. Also, this current description does not allow one to describe electrical (inductive or capacitive) coupling between paths. It is recommended that if coupling is an issue, then an electrical description be extracted from the physical parameters of the board.

What is, and is not, included in an Electrical Board Description is defined by its boundaries. For the definition of the boundaries, see the Description section under the [Path Description] Keyword.

USAGE RULES:

A .ebd file is intended to be a stand-alone file, not associated with any .ibs file. Electrical Board Descriptions are stored in a file whose name looks like <filename>.ebd, where <filename> must conform to the naming rules given in the General Syntax Section of this specification. The .ebd extension is mandatory.

CONTENTS:

A .ebd file is structured similar to a standard IBIS file. It must begin with a valid header section as defined by the IBIS-X specification, and the file is terminated by the [End] keyword. The actual board description is contained between the keywords [Begin Board Description] and [End Board Description], and includes the keywords listed below:

[Begin Board Description]
[Manufacturer]
[Number Of Pins]
[Pin List]
[Path Description]
[Reference Designator Map]
[End Board Description]

More than one [Begin Board Description]/[End Board Description] keyword pair is allowed in a .ebd file.

```

=====
Keyword:  [Begin Board Description]
Required:  Yes
Argument:  Text String (board level component name)
Description:  Marks the beginning of an Electrical Board Description.
Usage Rules:  The keyword is followed by the name of the board level
              component.  If the .ebd file contains more than one [Begin
              Board Description] keyword, then each name must be unique.
              The length of the component name must not exceed 40 characters
              in length, and blank characters are allowed.  For every
              [Begin Board Description] keyword there must be a matching
              [End Board Description] keyword.
=====

```

```

-----
[Begin Board Description]  16Meg X 8 SIMM Module
=====

```

```

=====
Keyword:  [Manufacturer]
Required:  Yes
Argument:  Text String (manufactures name)
Description:  Declares the manufacturer of the components(s) that use this
              .ebd file.
Usage Rules:  Following the keyword is the manufacturer's name.  It must not
              exceed 40 characters, and can include blank characters.

              Note: It is highly recommended that each manufacturer use a
              consistent name in all .ebd files.
=====

```

```

-----
[Manufacturer]  Quality SIMM Corp.
=====

```

```

=====
Keyword:  [Number Of Pins]
Required:  Yes
Argument:  None
Description:  Tells the parser the number of pins to expect.  Pins are any
              externally accessible electrical connection to the component.
Usage Rules:  The field must be a positive decimal integer.  Note: The
              simulator must not limit the Number Of Pins to any value less
              than 1,000.  The [Number Of Pins] keyword must be positioned
              before the [Pin List] keyword.
=====

```

```

-----
[Number Of Pins]  128
=====

```

```

=====
Keyword:  [Pin List]
Required:  Yes
Argument:  None
Description:  Tells the parser the pin names of the user accessible pins.
              It also informs the parser which pins are connected to power
              and ground.
Sub-Params:  signal_name
Usage Rules:  Following the [Pin List] keyword are two columns.  The first
              column lists the pin name while the second lists the data book
              name of the signal connected to that pin.  There must be as
              many pin_name/signal_name rows as there are pins given by the
              preceding [Number Of Pins] keyword.  Pin names must be the
              alphanumeric external pin names of the part.  The pin names
              cannot exceed eight characters in length.  Any pin associated
              with a signal name that begins with "GND" or "POWER" will be
              interpreted as connecting to the boards ground or power plane.
              In addition, "NC" is a legal signal name and indicates that the
              Pin is a 'no connect'.
=====

```

A SIMM Board Example:

```
[Pin List]  signal_name
A1          GND
A2          data1
A3          data2
A4          POWER5   | this pin connects to 5v
A5          NC       | a no connect pin
.
.
A22         POWER3.3 | this pin connects to 3.3v
B1          casa
.
.
etc.
```

```
=====  

Keyword:   [Path Description]  

Required:  Yes  

Argument:  None  

Description: This keyword allows the user to describe the connection  

             between the user accessible pins of a board level component  

             and other pins or pins of the ICs mounted on that board. Each  

             pin to node connection is divided into one or more cascaded  

             "sections", where each section is described in terms of its  

             L/R/C per unit length. The Fork and Endfork sub-parameters  

             allow the path to branch to multiple nodes, or another pin. A  

             path description is required for each pin whose signal name is  

             not "GND", "POWER" or "NC".
```

Board Description and IC Boundaries:

In any system, each board level component interfaces with another board level component at some boundary. Every electrical board description must contain the components necessary to represent the behavior of the board level component being described within its boundaries. The boundary definition depends upon the board level component being described.

For CARD EDGE CONNECTIONS such as a SIMM or a PC Daughter Card plugged into a SIMM Socket or Edge Connector, the boundary should be at the end of the board card edge pads as they emerge from the connector.

For any THROUGH-HOLE MOUNTED COMPONENT, the boundary will be at the surface of the board on which the component is mounted.

SURFACE MOUNTED COMPONENT models end at the outboard end of their recommended surface mount pads.

If the board level component contains an UNMATED CONNECTOR, the unmated connector will be described in a separate file, with its boundaries being as described above for the through-hole or surface mounted component.

```
Sub-Params: Len, L, R, C, Fork, Endfork, Pin, Node
Usage Rules: Each individual connection path (user pin to node(s))
              description begins with the [Path Description] keyword and a
              path name, followed by the sub-parameters used to describe the
              path topology and the electrical characteristics of each
              section of the path. The path name must not exceed 40
              characters, blanks are not allowed, and each occurrence of the
```

[Path Description] keyword must be followed by a unique path name. Every signal pin (pins other than POWER, GND or NC) must appear in one and only one path description per [Begin Board Description]/[End Board Description] pair. Pin names do not have to appear in the same order as listed in the [Pin List] table. The individual sub-parameters are broken up into those that describe the electrical properties of a section, and those that describe the topology of a path.

Section Description Sub-parameters:

The Len, L, R, and C sub-parameters specify the length, the series inductance, resistance, and the capacitance to ground of each section in a path description.

Len	The physical length of a section. Lengths are given in terms of arbitrary 'units'. Any non-zero length requires that the parameters that follow must be interpreted as distributed elements by the simulator.
L	The series inductance of a section, in terms of 'inductance/unit length'. For example, if the total inductance of a section is 3.0nH and the length of the section is 2 'units', the inductance would be listed as L = 1.5nH (i.e. 3.0 / 2).
C	The capacitance to ground of a section, in terms of capacitance per unit length.
R	The series DC (ohmic) resistance of a section, in terms of ohms per unit length.

Topology Description Sub-parameters:

The Fork and Endfork sub-parameters denote branches from the main pin-to-node or pin-to-pin connection path. The Node sub-parameter is used to reference the pin of a component or board as defined in a .ibs or .ebd file. The Pin sub-parameter is used to indicate the point at which a path connects to a user visible pin.

Fork	This sub-parameter indicates that the sections following (up to the Endfork sub-parameter) are part of a branch off of the main connection path. This sub-parameter has no arguments.
Endfork	This sub-parameter indicates the end point of a branch. For every Fork sub-parameter there must be a corresponding Endfork sub-parameter. As with the Fork sub-parameter, the Endfork sub-parameter has no arguments. The Fork and Endfork parameters must appear on separate lines.
Node	reference_designator.pin This sub-parameter is used when the connection path connects to a pin of another, externally defined component. The arguments of the Node sub-parameter indicate the pin and reference designator of the external component. The pin and reference designator portions of the argument are separated by a period ("."). The reference designator is mapped to an external component description (another .ebd file or .ibs file) by the [Reference Designator Map] Keyword. Note that a Node MUST reference a model of a passive or active component. A Node is not an arbitrary connection point between two elements or paths.
Pin	This sub-parameter is used to mark the point at which a path description connects to a user accessible pin. Every path description must contain at least one

occurrence of the Pin sub-parameter. It may also contain the reserved word NC. The value of the Pin sub-parameter must be one of the pin names listed in the [Pin List] section.

Note: The reserved word NC can also be used in path descriptions in a similar manner as the sub-parameters in order to terminate paths. This usage is optional.

Using The Sub-parameters to Describe Paths:

A section description begins with the Len sub-parameter and ends with the slash (/) character. The value of the Len, L, R, and C sub-parameters and the sub-parameter itself are separated by an equals sign (=); white space around the equals sign is optional. The Fork, Endfork, Node and Pin sub-parameters are placed between section descriptions (i.e., between the concluding slash of one section and the 'Len' parameters that starts another). The arguments of the Pin and Node sub-parameter are separated by white space.

Specifying a Len or L/R/C value of zero is allowed. If Len = 0 is specified, then the L/R/C values are the total for that section. If a non-zero length is specified, then the total L/R/C for a section is calculated by multiplying the value of the Len sub-parameter by the value of the L, R, or C sub-parameter. However, as noted below, if a non-zero length is specified, that section MUST be interpreted as distributed elements.

Legal Sub-parameter Combinations for Section Descriptions:

A) Len, and one or more of the L, R and C sub-parameters. If the Len sub-parameter is given as zero, then the L/R/C sub-parameters represent lumped elements. If the Len sub-parameter is non-zero, then the L/R/C sub-parameters represent distributed elements and both L and C must be specified, R is optional. The segment Len .../ must not be split; the whole segment must be on one line.

B) The first sub-parameter following the [Path Description] keyword must be 'Pin', followed by one or more section descriptions. The path description can terminate in a Node, another pin or the reserved word, NC. However, NC may be optionally omitted.

Dealing With Series Elements:

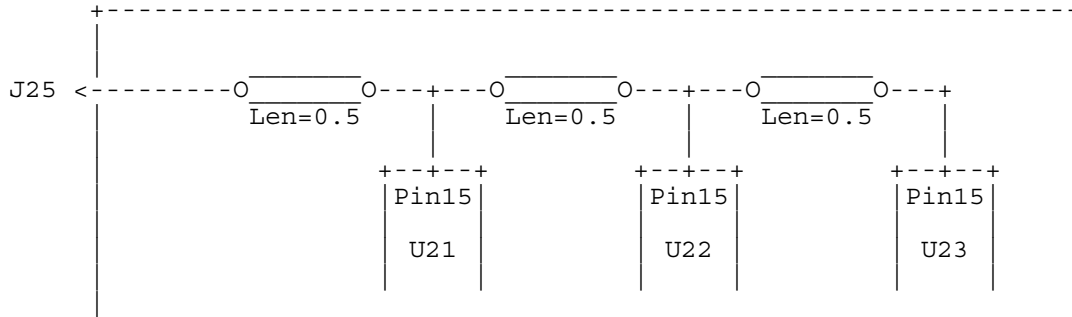
A discrete series R or L component can be included in a path description by defining a section with Len=0 and the proper R or L value. A discrete series component can also be included in a path description by writing two back to back node statements that reference the same component (see the example below). Note that both ends of a discrete, two terminal component MUST be contained in a single [Path Description]. Connecting two separate [Path Description]s with a series component is not allowed.

An Example Path For a SIMM Module:

```
[Path Description] CAS_2
Pin J25
Len = 0.5 L=8.35n C=3.34p R=0.01 /
Node u21.15
```

```

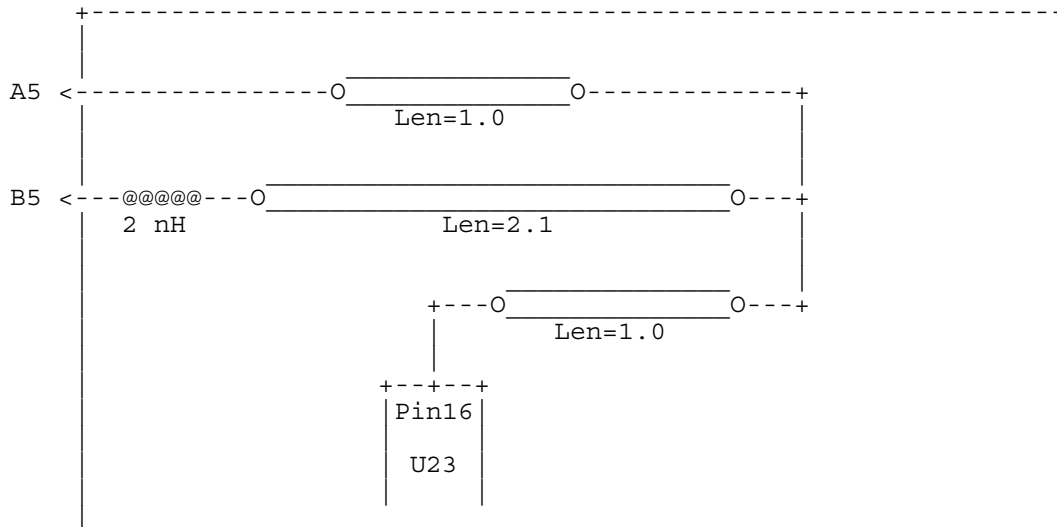
Len = 0.5 L=8.35n C=3.34p R=0.01 /
Node u22.15
Len = 0.5 L=8.35n C=3.34p R=0.01 /
Node u23.15
    
```



A Description Using The Fork and Endfork Sub-parameters:

```

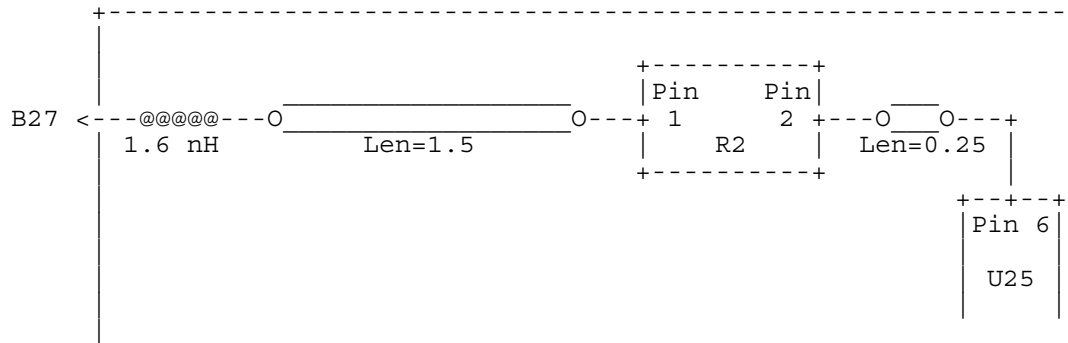
[Path Description] PassThru1
Pin B5
Len = 0 L=2.0n /
Len = 2.1 L=6.0n C=2.0p /
Fork
Len = 1.0 L = 1.0n C= 2.0p /
Node u23.16
Endfork
Len = 1.0 L = 6.0n C=2.0p /
Pin A5
    
```



A Description Including a Discrete Series Element:

```

[Path Description] sig1
Pin B27
Len = 0 L=1.6n /
Len = 1.5 L=6.0n C=2.0p /
Node R2.1
Node R2.2
Len = 0.25 L=6.0n C=2.0p /
Node U25.6
    
```



Keyword: [Reference Designator Map]
 Required: Yes, if any of the path descriptions use the Node sub-parameter
 Argument: None
 Description: Maps a reference designator to a component or electrical board description contained in an .ibs or .ebd file.
 Usage Rules: The [Reference Designator Map] keyword must be followed by a list of all of the reference designators called out by the Node sub-parameters used in the various path descriptions. Each reference designator is followed by the name of the .ibs or .ebd file containing the electrical description of the component or board, then the name of the component itself as given by the .ibs or .ebd file's [Component] or [Begin Board Description] keyword respectively. The reference designator, file name and component name terms are separated by white space. By default the .ibs or .ebd files are assumed to exist in the same directory as the calling .ebd file. It is legal for a reference designator to point to a component that is contained in the calling .ebd file.

Note: It is recommended that the reference designator be limited to ten characters.

[Reference Designator Map]

External Part References:

Ref Des	File name	Component name
u23	pp100.ibs	Pentium(R)_Pro_Processor
u24	simm.ebd	16Meg X 36 SIMM Module
u25	ls244.ibs	National 74LS244a
u26	r10K.ibs	My_10K_Pullup

Keyword: [End Board Description]
 Required: Yes
 Argument: None
 Description: Marks the end of an Electrical Interconnect Description.
 Usage Rules: This keyword must come at the end of each complete electrical interconnect model description.

Optionally, a comment may be added after the [End Electrical Description] keyword to clarify which board model has ended.

[End Board Description] | End: 16Meg X 8 SIMM Module

=====
=====
Section 11

END KEYWORD

=====
=====
An IBIS-X file is terminated with the [End] keyword as shown below. The [End] keyword is also used to terminate a stand-alone .pkg or .ebd file.

=====
Keyword: [End]
Required: Yes
Argument: None
Description: Marks the end of the .ibs, .pkg, or .ebd file. Any text following this keyword should be ignored.

[End]