IBIS Interconnect SPICE Subcircuits Specification
(IBIS-ISS)

Draft 1.2
May 11, 2011

Contents

1 Overview

The IBIS Open Forum, in order to enable easier data exchange between users of signal/power integrity simulation and physical layout/routing software tools, is issuing a generic circuit description format, to be called "IBIS Interconnect SPICE Subcircuits" (IBIS-ISS).

This format is similar in structure and major functions to the SPICE (Simulation Program with Integrated Circuit Emphasis) nodal syntax developed at the University of California at Berkeley and since implemented in various forms by individual software tool vendors. IBIS-ISS is the first industry-wide attempt to standardize SPICE subcircuit representation.

This version of IBIS-ISS is based on a subset of HSPICE®, used with permission from Synopsys, Inc. HSPICE is a registered trademark of Synopsys, Inc.

2 Goals and Scope

The syntax of IBIS-ISS is intended to:

- describe interconnect structures (such as PCB traces, connectors, cables, etc.) electrically, for analysis in a signal integrity and/or power integrity context
- describe the arrangement or topology of interconnect structures, as they relate to each other and to active devices in a system

To these ends, IBIS-ISS includes support for:

- elementary circuit elements (resistors, capacitors, inductors)
- transmission line elements (lossless and lossy)
- frequency-domain network parameters (e.g., S-parameters)
- parameter/variable passing to elements and subcircuits
- dependent and limited independent sources
- string-based node naming
- user-defined comments
- abstraction through modular, user-defined subcircuit definitions

IBIS-ISS does NOT include or cover:

- descriptions of complete netlists intended for input "as-is" to simulation tools
- model formats or "process cards" for active devices (e.g., diodes, transistors)
- controls or options for any simulation engine (e.g., precision, algorithm selection)
- simulation or analysis types (e.g., DC, transient)
- sweep or run control (e.g., Monte Carlo)
- geometrical descriptions for field solvers
- support for other kinds of data extraction/export (e.g., S-parameter generation)
- measurement, printing or probing
- encryption support

3 Conventions

The following typographical conventions are used in IBIS-ISS. Note that these may be combined (e.g., Courier font in bold type).

Table 1: Document Conventions

Convention	Description
`Courier`	Indicates statement syntax
Italic	Indicates a user-defined value, where a specific text string will replace the italic text shown (e.g., `Rxxxx` is a generic representation of a resistor element name, such as `Rname`)
Bold	Indicates verbatim text in syntax descriptions
[ ]	Denotes optional arguments
...	Indicates that arguments of the same type may be added as appropriate to the element structure: pin1 pin2 ... pinn
|	Indicates a choice among defined alternatives, such as low | medium | high
+	Indicates a continuation of a statement across lines. Note that continuation may only be used between arguments and shall not split any single non-quoted-string argument.

4 Input Structure and Data Entry

This section describes the input file and structures for representing input data.

4.1   Input File Guidelines

An input file consists of a collection of statements describing a portion of a complete circuit. This input file is intended for inclusion in a larger netlist or circuit description, to be used by a simulation tool.

An input filename may be up to 1024 characters long. The input file shall be in ASCII format (as defined in ISO/IEC 8859-1:1998, with each character encoded in 8 bits). The input file shall not be in a binary, packed or compressed format.

A line in IBIS-ISS is a text string consisting of any legal combination of characters ending in a line-termination sequence as defined by the operating system (e.g, line feed or LF in UNIX systems). Any individual input line may be up to 1024 characters long.

4.2   Statements and Arguments

A statement in IBIS-ISS is a text string consisting of character sequence arguments and delimiters (see Table 3: IBIS-ISS Special Characters for a list of delimiters). An IBIS-ISS file may contain multiple statements (the number of statements is not limited by the IBIS-ISS definition, but may be limited by the computer architecture and/ or operating system used to process the file).

Statements may be split across lines, provided a line continuation character or sequence (defined later) is used. No more than one statement may appear on any line. Statements in an input file may appear in any order.

Any valid string of characters between two delimiters is an argument.

For the purpose of this specification, statements are grouped into the following types:

- Element instances
- Parameter definitions
- File includes
- Subcircuit definitions
- Model definitions
- Comments
- Subcircuit ending statements

Subcircuit ending statements, subcircuit definitions, model definitions, parameter definitions and file includes all begin with the period (.) character.

The specific syntax of the above statement types are described in the sections below.

- IBIS-ISS ignores differences between upper and lower case in input statements, except in quoted filenames.

- To continue a statement across multiple lines, the plus (+) sign shall be used as the first non-blank character in the continuation line. The + sign shall be used in addition to a delimiter and never to split up the character sequences between delimiters.

- Quoted strings with extended length (such as file names including paths and expressions) may span multiple lines using a double backslash (\\ ) sequence , where the double backslash precedes the line-termination sequence. Note that the double backslash (\\ ) sequence may be preceded by whitespace when used within a quoted string. Individual arguments may be split across lines using a double backslash, but any leading whitespace will be concatenated. See 4.11Line Continuations for details and an example.

- The following characters are reserved for special use and shall not be used as part of any parameter or node name:

    ( ) = " '

- "Directional" or matched quotation marks, as represented by ISO 8859-1 (extended ASCII) characters 0x145d, 0x146d, 0x147d and 0x148d, are prohibited in IBIS-ISS files.

- To prevent conflicts with proprietary SPICE variants, the following strings shall not be used as part of any parameter or node name: time, temper, hertz. Further, the strings listed in Table 2 shall not be used as part of any parameter or node name in the associated element.

Element	Prohibited String Sequence
Capacitor	POLY, TC, SENS
E-element G-element	AND, DELAY, FOSTER, LAPLACE, NAND, NPWL, NOR, VCCS, OPAMP, OR, POLE, POLY, PWL, SPUR, TRANSFORMER, VCR, VCCAP, VCVS, FREQ, ZTRANS, VMRF, NOISE, NOISEFILE, MNAME, PHASE, SCALE, MAX, PAR
F-element H-element	POLY, PWL, AND, NAND, OR, NOR, VMRF, CCCS, CCVS, DELAY
Inductor	POLY, TC, SENS, RELUCTANCE, TRANSFORMER_NT, FILE
Resistor	POLY,TC,SENS
S-element	ZO, Z0, MNAME
T-element	IC
W-element	RLGCFILE, PRINTZO, RLGCMODEL,TABLEMODEL, FSMODEL, UMODEL, SMODEL

4.3   **Special Characters**

The following table lists the special characters that may be used as part of node names, element parameter names, and element instance names. For detailed discussion, see the appropriate sections in this chapter.

Note:

To avoid unexpected results or error messages, do not use the following mathematical characters in a parameter name in IBIS-ISS: * - + ^ and /.

Special Character*		Usage in Node Names	Usage in Element and Subcircuit Instance Names**	Usage in Parameter Names**	Remarks
~	Tilde	Legal anywhere	Included only	Included only	n/a
!	Exclamation point	Legal anywhere	Included only	Included only	n/a
@	At sign	Legal anywhere	Included only	Included only	n/a
#	Pound sign	Legal anywhere	Included only	Included only	n/a
$	Dollar sign	Included only (avoid if after a number in node name)	Included only	Included only	In-line comment character
%	Percent	Legal anywhere	Included only	Included only	n/a
^	Caret	Legal anywhere	Included only	Included only (avoid usage)	"To the power of", i.e., 2^5, two raised to the fifth power
&	Ampersand	Legal anywhere	Included only	Included only	n/a
*	Asterisk	Included only (avoid using in node names)	Included only	Included only (avoid using in parameter names)	Comment and wildcard character. Double asterisk (**) is "to the power of".

( )	Parentheses	Illegal	Illegal	Illegal	Delimiter
-	Minus	Included only	Included only	Illegal	n/a
_	Underscore	Legal anywhere	Included only	Included only	n/a
+	Plus sign	Included only	Included only	Included only (avoid usage)	Continues previous line, except for quoted strings
=	Equals	Illegal	Illegal	Optional in .PARAM statements	Delimiter
< >	Less/more than	Legal anywhere	Included only	Included only	n/a
?	Question mark	Legal anywhere	Included only	Included only	Wildcard character
/	Forward slash	Legal anywhere	Included only	Illegal	n/a
{ }	Curly braces	included only, converts to [ ]	Included only	Included only	Engine shall auto-convert to square brackets ([ ])
[ ]	Square brackets	Included only	Included only	Included only	n/a
\\	Double backslash	Included only	Illegal	Illegal	Continuation character sequence

|	Pipe	Legal anywhere	Included only	Included only	n/a
,	Comma	Illegal	Illegal	Illegal	Delimiter
.	Period	Illegal	Illegal	Illegal	Used only as a special statement identifier (i.e., .PARAM)
:	Colon	Included only	Included only	Included only	Delimiter for element attributes
;	Semi-colon	Included only	Included only	Included only	n/a
" "	Double-quotes	Illegal	Illegal	Illegal	Expression and filename delimiter
' '	Single quotes	Illegal	Illegal	Illegal	Expression and filename delimiter
	Blank (whitespace)	May be used before \\ sequence for continuations	Illegal	Illegal	Delimiter

* any position in string, including the first character

** cannot be the first character; element key letter only

"Legal anywhere" = any position in string, including the first character
"Included only"=any position except first character

4.4   **First Character**

The first non-blank character in every line specifies how IBIS-ISS interprets the remainder of the line.

13

If the First Character is...	Indicates
. (period)	Statement identifier (e.g., .PARAM)
c, C, e, E, f, F, g, G, h, H, k, K, l, L, r, R, s, S, t, T, v, V, w, W, x, X	Element instantiation
* (asterisk)	Comment line
+ (plus)	Continues previous line

4.5   Delimiters

Delimiters separate arguments in the input file. Input delimiters are: tab, blank, comma (,), equal sign (=), and parentheses ( ).

In addition, single (') or double quotes (") delimit and group expressions and filenames.

4.6   Instance Names

The names of element instances begin with the element key letter or identifier as listed below. Instance names may be up to 1024 characters long.

Table 5: Element Identifiers

Key Letter (First Char)	Element	Example Statement
C	Capacitor	`Cbypass 1 0 10pf`
E	Voltage-controlled voltage source	`Ea 1 2 3 4 1K`

F	Current-controlled current source	`Fsub n1 n2 vin 2.0`
G	Voltage-controlled current source	`G12 4 0 3 0 10`
H	Current-controlled voltage source	`H3 4 5 Vout 2.0`
K	Linear mutual inductor (general form)	`K1 L1 L2 1`
L	Linear inductor	`LX a b 1e-9`
R	Resistor	`R10 21 10 1000`
S	S-parameter element	`S1 nd1 nd2 s_model2`
T	Transmission line	`Txxx in 0 out 0 Zo=50` `TD=30n`
V	DC voltage source	`V1 8 0 DC=0`
W	Transmission line	`W1 in1 0 out1 0 N=1 L=1`
X	Subcircuit instance	`X1 2 4 17 31 MULTI WN=100` `LN=5`

4.7   **Numbers**

Numbers may be entered as integer, floating point, floating point with an integer exponent, or integer or floating point with one of the scale factors listed below.

Table 6: Scale Factors

Scale Factor	IEEE Standard Prefix	IEEE Standard Symbol	Multiplying Factor

T	tera	T	1e+12
G	giga	G	1e+9
MEG	mega	M	1e+6
K	kilo	k	1e+3
MIL	n/a	mil	25.4e-6
M	milli	m	1e-3
U	micro	μ	1e-6
N	nano	n	1e-9
P	pico	p	1e-12
F	femto	f	1e-15
A	atto	a	1e-18

Note:

Scale factor A is not a scale factor in a character string that contains amps. For example, IBIS-ISS-compliant tools shall interpret the string "20amps" as 20 amperes of current, not as 20e-18mps.

- Numbers may use exponential format or engineering key letter format, but not both (1e-12 or 1p, but not 1e-6u).

- To designate exponents, use D or E.

- Trailing alphabetic characters are interpreted as units comments (for example, 1kV is interpreted as 1k, while 1w is interpreted as 1).

- Units comments are not checked.

4.8   **Parameters and Expressions**

- Parameter names shall begin with an alphabetic character. Subsequent characters in the parameter name shall be either alphanumeric characters or special characters as defined in Table 3: IBIS-ISS Special Characters

- If multiple definitions are given for the same parameter, IBIS-ISS uses the last parameter definition even if that definition occurs later in the input than a reference to the parameter.

- A parameter shall be defined before that parameter is used in a definition for another parameter.

- To delimit expressions, single quotes shall be used.

- Expressions shall not exceed 1024 characters.

Parameters are evaluated only once, at the time of parsing. Dynamic, recursive or iterative definitions of parameters are prohibited (e.g., parameters defined in terms of the voltage at a node, where that voltage is evaluated at every time step in a transient analysis).

- Parameters are used in two contexts.

    o Parameters in parameter definition statements are strings, defining names that are assigned specific values by the statement. These values may themselves be interpreted as strings (using the "str()" construction noted elsewhere), numeric values, an expression or equation, or strings matching parameters defined elsewhere.

    o Parameters may also appear in element instances, model definitions and subcircuit definitions. These parameters may be user-defined or may use names defined by the syntax of the element. Parameter names are input arguments. Delimiters shall precede and follow names.

- Parameter names may be up to 1024 characters long and are not case-sensitive.

4.9   **Node Name (or Node Identifier) Conventions**

Nodes are the points of connection between elements in the input circuit description. If entirely numeric, node numbers shall be between 1 and 999999999999999 (1 to 1e16-1). A node number of 0 is permitted but is interpreted as ground. Letters that follow a leading number in a node name are ignored; this means that node strings such as '3n5' and '3' shall be interpreted as referring to the same node.

When the node name begins with a letter or a valid special character, the node name may contain a maximum of 1024 characters. See Table 3: IBIS-ISS Special Characters for a list of valid special characters.

17

To indicate the ground node, use either the number `0` or the names `GND`, `!GND`, `GROUND`, or `GND!`. Every node shall have at least two connections, except for transmission line nodes (unterminated transmission lines are permitted).

4.10   Element, Instance, and Subcircuit Naming Conventions

Instances and subcircuits are elements and as such, follow the naming conventions for elements.

Element names begin with an identifying letter designating the element type, followed by up to 1023 characters (see Table 3: IBIS-ISS Special Characters for a list of valid special characters). Element identifiers are R for resistor, C for capacitor and so on.

4.11   Line Continuations

Statements may be continued across lines in one of three ways:

- Statements are continued across lines using the + character as the first non-blank character in the continued line

- Quoted strings are continued across lines using the \\ sequence as the last two characters at the end of the line to be continued

- Individual arguments are continued using the \\ sequence, without leading whitespace

The + sign shall be used only between arguments and delimiters and never to split up the character sequence used for a single argument.

Here is an example of comments and line continuation in a statement:

```
* This shows continuation of a statement describing a
* resistor
Rexample
+ n3 n4 R=30
```

To continue a statement, including a quoted string, with extended length across two lines, an argument may be split using a double backslash (\\ ) sequence. Note that any whitespace preceding the double backslash will be concatenated with the line immediately following it. Here are examples of string and argument continuation.

```
*** string continuation ***
R6 4 0 R='res1-\\
```

```
res2'
R5 4 0 R='res1- \\
res2'
*** argument continuation ***
R4 node1 no\\
de2 R= 'res1-res2'
```

Note that whitespace leading the continued input will be interpreted as a delimiter, if the input is not a quoted string. In the example above, whitespace before the string `de2` will cause an error, as `R4` will appear to have three nodes.

4.12   IBIS-ISS Structure

IBIS-ISS files shall include at least one subcircuit definition, aside from any included files. An IBIS-ISS file may contain multiple subcircuit definitions.

5 Parameters

Parameters are similar to the variables used in most programming languages. Parameters hold values assigned when the circuit design is created. Parameters may store static values for a variety of quantities (resistance, source voltage, rise time, and so on). Parameters may also be alphabetic strings used with elements where string input is expected (for example, filenames or model names).

5.1   Using Parameters in Simulation (.PARAM)

Defining Parameters

Parameters may be defined using the methods shown below. Note that a .param statement without an assignment is not permitted.

Table 7: .PARAM Statement Syntax and Examples

Usage	Description/Example
Simple Assignment	**.PARAM** *SimpleParam***=**1e-12
Algebraic Definition	**.PARAM** *AlgebraicParam***=**'SimpleParam*8.2'
User-defined Function	**.PARAM** *MyFunc(x,y)***=**'SQRT((x*x)+(y*y))'
String Assignment	**.PARAM** *StringParam***=str(**'mystring'**)**
Subcircuit Definition	**.SUBCKT** *SubName ParamDefName***=***Value*
Subcircuit Instance	**X***xxx nodename1 ... nodenamen* **+** *SubName* **+** *ParamDefName* **=** *Value* | **str(**'*string*'**)**

A parameter definition in IBIS-ISS always uses the last value found in the input statements. The definitions below assign a value of 3 to the DupParam parameter.

```
.PARAM DupParam=1
...
```

20

```
.PARAM DupParam=3
```

IBIS-ISS assigns `3` as the value for all instances of `DupParam`, including instances that are earlier in the input than the `.PARAM DupParam=3` statement.

Note that any tail-truncated substring of ".parameters" containing at least the characters ".para" is also acceptable as an alternative to ".param".

The parameter resolution order is:

1. Resolve all literal assignments.
2. Resolve all expressions.
3. Resolve all function calls.

Assigning Parameters

The following types of values may be assigned to parameters:

- Constant real number
- Algebraic expression of real values
- Strings not for algebraic evaluation

Any complex expression to be evaluated shall be enclosed in single quotes.

A simple expression consists of one parameter name. Simple expressions shall not be enclosed in single or double quotes.

The parameter keeps the assigned value, unless a later definition changes its value.

Assignments of different values to the same parameter name in different areas of the IBIS-ISS file are resolved in the following order (in decreasing order of priority):

1. .SUBCKT call (where a parameter is assigned as part of a subcircuit instance)
2. .SUBCKT definition (where a parameter symbol is defined for passing in a value, and a default value is assigned)
3. .PARAM statement

See also Section 5.4 for additional rules and an example.

5.2   Using Algebraic Expressions

In IBIS-ISS, an algebraic expression, with quoted strings, may replace any parameter. Some uses of algebraic expressions are:

- Parameters:

```
.PARAM x='y+3'
```

- Inline expressions in elements:

```
R1 1 0 r='27*3.14'
```

Built-In Functions and Variables

In addition to simple arithmetic operations (+, -, *, /), the built-in functions and variables listed below may be used in IBIS-ISS expressions.

Table 8: IBIS-ISS Built-in Functions

IBIS-ISS Form	Function		Description
sin(x)	sine		Returns the sine of x (radians)
cos(x)	cosine		Returns the cosine of x (radians)
tan(x)	tangent		Returns the tangent of x (radians)
asin(x)	arc sine		Returns the inverse sine of x (radians)
acos(x)	arc cosine		Returns the inverse cosine of x (radians)
atan(x)	arc tangent		Returns the inverse tangent of x (radians)
sinh(x)	hyperbolic sine		Returns the hyperbolic sine of x
cosh(x)	hyperbolic cosine		Returns the hyperbolic cosine of x
tanh(x)	hyperbolic tangent		Returns the hyperbolic tangent of x
abs(x)	absolute value		Returns the absolute value of x: |x|

sqrt(x)	square root		Returns the square root of the absolute value of x: sqrt(-x)=-sqrt(|x|)
pow(x,y)	absolute power		Returns the value of x raised to the integer part of y: $x^{(integer\ part\ of\ y)}$
pwr(x,y)	signed power		Returns the absolute value of x, raised to the y power, with the sign of x: (sign of x)$\|x\|^{y}$
x**y	power		If x<0, returns the value of x raised to the integer part of y. If x=0, returns 0. If x>0, returns the value of x raised to the y power.
log(x)	natural logarithm		Returns the natural logarithm of the absolute value of x, with the sign of x: (sign of x)log(|x|)
log10(x)	base 10 logarithm		Returns the base 10 logarithm of the absolute value of x, with the sign of x: (sign of x)$\log_{10}(\|x\|)$
exp(x)	exponential		Returns e, raised to the power x: e^{x}
db(x)	decibels		Returns the base 10 logarithm of the absolute value of x, multiplied by 20, with the sign of x: (sign of x)20$\log_{10}(\|x\|)$
int(x)	integer		Returns the integer portion of x. The fractional portion of the number is lost.
nint(x)	integer		Rounds x up or down, to the nearest integer.

sgn(x)	return sign		Returns -1 if x is less than 0. Returns 0 if x is equal to 0. Returns 1 if x is greater than 0
sign(x,y)	transfer sign		Returns the absolute value of x, with the sign of y: (sign of y)|x|
def(x)	parameter defined		Returns 1 if parameter x is defined. Returns 0 if parameter x is not defined.
min(x,y)	smaller of two args		Returns the numeric minimum of x and y
max(x,y)	larger of two args		Returns the numeric maximum of x and y
[cond] ?x : y	ternary operator		Returns *x* if *cond* is not zero. Otherwise, returns *y*. .param z='condition ? x:y'
<	relational operator (less than)		Returns 1 if the left operand is less than the right operand. Otherwise, returns 0. .param x=y<z (y less than z)
<=	relational operator (less than or equal)		Returns 1 if the left operand is less than or equal to the right operand. Otherwise, returns 0. .param x=y<=z (y less than or equal to z)
>	relational operator (greater than)		Returns 1 if the left operand is greater than the right operand. Otherwise, returns 0. .param x=y>z (y greater than z)

>=	relational operator (greater than or equal)		Returns 1 if the left operand is greater than or equal to the right operand. Otherwise, returns 0. .param x=y>=z (y greater than or equal to z)
==	equality		Returns 1 if the operands are equal. Otherwise, returns 0. .param x=y==z (y equal to z)
!=	inequality		Returns 1 if the operands are not equal. Otherwise, returns 0. .param x=y!=z (y not equal to z)
&&	Logical AND		Returns 1 if neither operand is zero. Otherwise, returns 0. .param x=y&&z (y AND z)
||	Logical OR		Returns 1 if either or both operands are not zero. Returns 0 only if both operands are zero. .param x=y||z (y OR z)

5.3   String Parameters

Parameters may be defined and instantiated using strings. String parameters use special syntax; characters such as single quotes ('), double quotes ("), or curly brackets ({}) alone are not sufficient for string parameter definition or instantiation.

When defining a parameter that is a character string, the combination `str('string')` may be used to define the parameter, where `string` is the string to be used as the parameter value. When the parameter is used, the parameter name is called as `str(parameter_name)`. Note that quotation marks are not used in the call `str(parameter_name)`.

IBIS-ISS supports string parameter definition and instantiation for the following:

* `.PARAM` statements
* `.SUBCKT` statements
* `TSTONEFILE` keyword in the S-element
* W-element arguments `RLGCMODEL` and `TABLEMODEL`

Note that the `str(parameter_name)` construction is not required for arguments that take string inputs (for example, RLGCMODEL and TABLEMODEL assignments).

5.4   Parameter Scoping and Passing

A parameter is defined either by a `.param` statement (local to the subcircuit in which it is defined), or may be passed into a subcircuit, or may be assigned as part of a .subckt definition.

All parameters defined within a subcircuit are local to that subcircuit under the defined name. Parameter values may be passed between subcircuits, but the values shall be passed explicitly, with the parameter present both in the subcircuit definition and in the instance where it is used. IBIS-ISS does not support global parameters.

```
.subckt def 1 2
.param x=1
x1 1 2 abc x=2
.subckt abc 1 2 x=3
.param x=4
r1 1 2 R=x
.ends abc
.ends def
```

In the example above, the value of r1 is 2 ohms. The .SUBCKT definition x=3 and the local .PARAM assignments x=1 and x=4 are overridden by the .SUBCKT instance assignment x=2.

6 File Includes

The include statement inserts another file's contents in the current file at evaluation. Note that any tail-truncated substring of ".include" containing at least the characters ".inc" is also acceptable as an alternative to ".include".

An include file may contain nested .INCLUDE calls to another include file.

Syntax

.INCLUDE *'file_path_and_name'*

.inc *'file_path_and_name'*

Table 9: File Include Arguments

Argument	Description
file_path_and_name	File name, optionally preceded by a file path, of a file for computer operating systems that support tree-structured directories. Any strings, including directory separator characters, valid under the computer's operating system may be used.

Use this statement to include the contents of another file in the current circuit description. The file path and file name shall be enclosed in single or double quotation marks.   Relative paths are strongly suggested.

```
.INCLUDE '/myhome/subcircuits/circuit'
```

7 Comments

Comments require an asterisk (*) as the first non-blank character in the line (to make the entire line a comment) or a dollar sign ($) directly in front of the comment text.

Syntax

```
* <comment_on_a_line_by_itself>
```

or

```
<IBIS-ISS statement> $ <comment following input>
```

Comment statements may appear anywhere in the circuit description. The dollar sign ($) shall be used for comments that do *not* begin in the first character position on a line (for example, for comments that follow a statement on the same line). If it is not the first nonblank character, then the dollar sign shall be preceded by either:

- Whitespace
- Comma (,)
- Valid numeric expression

The dollar sign may also be used within node or element names. For example:

```
* RF=1K GAIN SHOULD BE 100
$ CIRCUIT EXAMPLE
VIN 1 0 PL 0 0 5V 5NS $ 10v 50ns
R12 1 0 1MEG $ FEED BACK
.PARAM a=1w$comment a=1, w treated as a space and ignored
.PARAM a=1k$comment a=1e3, k is a scale factor
```

In the example above, the first parameter statement definition assigns a value of 1 to the parameter a, as training alphabetic characters are interpreted as units comments and w is not a valid scale factor. The second parameter statement definition uses k, a valid scale factor, which changes the assigned value of a to 1000 (see Section 4.7).

A dollar sign is the preferred way to indicate comments, because of the flexibility of its placement within the code.

Comments shall not appear after the double-slash (\\) sequence before a line-termination sequence.

8 Model Definitions (.MODEL Statements)

Model definitions are used to specify the electrical parameters for W-element and S-element instances.

The specific syntax for W-element and S-element .MODEL definitions is detailed below, as part of the W-element and S-element portions of the IBIS-ISS specification.

9 Subcircuit Definitions

A subcircuit definition groups a set of statements into a module that may be reused an unlimited number of times. The definition is instantiated using the X-element. At least one node is required. Parameters may be optionally assigned to a value as part of the .subckt definition. The parameter value is a default and may be overridden in instantiations of the subcircuit by assignments using the same parameter name.

Syntax

```
.subckt name n1 [n2 n3 …] [parnam=val]
statement
statement
statement
…
.ends
```

9.1   Subcircuit Scoping Rules

A .subckt or .model definition shall occur in the subcircuit in which the subcircuit or model is referenced, or in a calling subcircuit at any level above.

10 Subcircuit Definition Ending Statements

Subcircuit definitions shall end with the .ends statement. See Subcircuit Definitions above for syntax and examples.

11 Elements

The sections below describe the individual circuit elements that may appear in an IBIS-ISS file.

11.1   Subcircuits

Reusable cells are the key to saving labor in any CAD system. To create and simulate a reusable circuit, construct it as a subcircuit. Parameters are used to expand the utility of a subcircuit.

X*xxx* creates an instance of a subcircuit. The subcircuit shall have already been defined elsewhere in the IBIS-ISS file using a .SUBCKT statement.

Syntax
```
Xxxx n1 [n2 n3 …] subnam [parnam = val] [M = val]
```

The name of an X-element instance shall begin with the character "X", followed by up to 1023 characters (see Table 3: IBIS-ISS Special Characters for a list of valid special characters).

Table 10: Subcircuit Arguments

Argument	Definition
n1 …	Node names for external reference.
subnam	Subcircuit model reference name.
parnam=val	A parameter name (*parnam*) set to a value (*val*) for use only in the subcircuit. It overrides a parameter value in the subcircuit definition.
M=*val*	Multiplier value *val* instantiates *val* number of subcircuits connected in parallel to the named nodes. The multiplier *val* shall be a positive, non-zero integer.

11.2   Linear Resistor

A linear resistor is a basic electrical circuit element for impeding current flow.

Syntax

```
Rxxx n1 n2 [R =] value
```

The value of a linear resistor may be a constant, or an expression of parameters.

The name of an R-element instance shall begin with the character "R", followed by up to 1023 characters (see Table 3: IBIS-ISS Special Characters for a list of valid special characters).

Table 11: Linear Resistor Arguments

Argument	Description
n1, *n2*	Names of connecting nodes.
R=*value*	Resistance value, in ohms. The text "**R=**" is optional.

11.3   Linear Capacitor

A linear capacitor is a basic electrical circuit element for charge storage.

Syntax

```
Cxxx n1 n2 [C =] value
```

The value of a linear capacitor may be a constant, or an expression of parameters.

The name of a C-element instance shall begin with the character "C", followed by up to 1023 characters (see Table 3: IBIS-ISS Special Characters for a list of valid special characters).

Table 12: Linear Capacitor Arguments

Argument	Description
n1, n2	Names of connecting nodes.
C=*value*	Capacitance value, in farads. The text "**C=**" is optional.

11.4   Voltage Source

A voltage source establishes a DC potential difference between two nodes. A zero-volt source may be used to create a short between two nodes.

Syntax

`Vxxx n1 n2 [DC =] val`

The name of a V-element instance shall begin with the character "V", followed by up to 1023 characters (see Table 3: IBIS-ISS Special Characters for a list of valid special characters).

Table 13: Voltage Source Arguments

Argument	Description
n1, n2	Names of connecting nodes.
DC=val	Sets val as the DC voltage between the source nodes. The text "DC=" is optional.

11.5   Mutual Inductor

A mutual inductor describes inductive coupling between two defined inductors.

Syntax

`Kxxx Lyyy Lzzz [K =] coupling`

The name of a K-element instance shall begin with the character "K", followed by up to 1023 characters (see Table 3: IBIS-ISS Special Characters for a list of valid special characters).

Table 14: Mutual Inductor Arguments

Argument	Description
Lyyy	Name of the first of two coupled inductors. This inductor shall be defined elsewhere in the file.

L*zzz*	Name of the second of two coupled inductors. This inductor shall be defined elsewhere in the file.
K=*coupling*	Sets *coupling* as the coefficient of mutual coupling. This is a non-zero unitless real number. If the coupling coefficient is negative, the direction of coupling reverses. This is equivalent to reversing the polarity of either of the coupled inductors. Use the **K=***xxx* syntax when defining the coupling coefficient using a parameter name or an equation. The text "**K=**" is optional.

11.6   Linear Inductor

Syntax

`L`*xxx n1 n2* [`L =`] *inductance*

The name of an L-element instance shall begin with the character "L", followed by up to 1023 characters (see Table 3: IBIS-ISS Special Characters for a list of valid special characters).

Table 15: Linear Inductor Arguments

Argument	Description
n1, n2	Names of connecting nodes.
L=*inductance*	Sets *inductance* as the inductance value, in henries. The text "**L=**" is optional.

11.7   T-element (Ideal Transmission Line)

Syntax

`T`*xxx in refin out refout* `Zo=`*val* `TD=`*val* [`L=`*val*]

The name of a T-element instance shall begin with the character "T", followed by up to 1023 characters (see Table 3: IBIS-ISS Special Characters for a list of valid special characters).

Argument	Description
in	Signal input node.
refin	Ground reference for the input signal.
out	Signal output node.
refout	Ground reference for the output signal.
Zo=*val*	Sets *val* as the characteristic impedance of the transmission line in ohms. Note that the beginning character sequence may be either zo or z0.
TD=*val*	Sets *val* as the propagation time delay of the transmission line, in seconds. If physical length (L) is specified, then units for TD are considered in seconds per meter.
L=*val*	Sets *val* as the physical length of the transmission line, in meters. Default=1.

11.8   **W-element (Coupled Transmission Line)**

The W-element is a versatile transmission line model that may be used to describe a variety of transmission line structures, from a simple lossless line to complex frequency-dependent lossy-coupled lines.

> **Syntax**
> **W***xxx i1 i2 ... iN iR o1 o2 ... oN oR* **N=***val* **L=***val*
> **+ RLGCMODEL=***name* | **TABLEMODEL=***name* [**FGD=***val*]

The name of a W-element instance shall begin with the character "W", followed by up to 1023 characters (see Table 3: IBIS-ISS Special Characters for a list of valid special characters).

The W-element is organized in terms of coupled conductors, each of which has two terminals (one at each end of the conductor) which connect to other circuit elements. The nodes of the W-element correspond to its conductor terminals as shown in Table 17: W-element (Coupled Transmission Line) Arguments.   A reference conductor is always assumed. The number of conductors is therefore related to the number of W-element nodes as $n=2*(c+1)$, where c is the number of conductors (including the reference) and n is the number of nodes.

Table 17: W-element (Coupled Transmission Line) Arguments

Argument	Description
N=*val*	Sets positive non-zero integer *val* as the number of signal conductors (excluding the reference conductor).
i1...iN	Node names for the near-end signal conductor terminals
iR	Node name for the near-end reference conductor terminal
o1... oN	Node names for the far-end signal conductor terminals
oR	Node name for the far-end reference conductor terminal
L=*val*	Defines *val* as the length of the transmission line(s), in meters

RLGCMODEL=_name_	Defines string _name_ as the name of the RLGC model
TABLEMODEL=_name_	Defines string _name_ as the name of the frequency-dependent tabular model
FGD=_val_	Specifies the cut-off frequency of dielectric loss, in hertz, as positive real number _val_. Zero is permitted.

Parameters in the W-element element declaration may be declared in any order. Specify the number of signal conductors, N, after the list of nodes. The W-element does not limit the number of coupled conductors.

The W-element supports two formats to specify transmission line properties:

- Format 1: RLGC specification, specified in a .MODEL statement

- Format 2: Frequency-dependent tabular specification, specified in a .MODEL statement

The **RLGCMODEL** and **TABLEMODEL** arguments associate a W-element definition with a .MODEL using either Format 1 or Format 2, respectively. A W-element shall use either **RLGCMODEL** or **TABLEMODEL**, but not both.

In both formats, the characteristics of the W-element are expressed in per unit length matrices: R_o (DC resistance), L, G, C, R_s (skin effect), and G_d (dielectric loss).

Format 1: RLGC Model

The RLGC .MODEL format supports frequency-independent RLGC matrices per unit length. The format also supports frequency-dependent R and G data, and lossless (LC) lines. It does not support RC lines.

Because RLGC matrices are symmetrical, the RLGC model specifies only the lower triangular parts of the matrices. The syntax of the RLGC model for the W-element is:

```
.MODEL name W MODELTYPE=RLGC N=val
+ Lo=matrix_entries
+ Co=matrix_entries [Ro=matrix_entries] [Go=matrix_entries]
+ [Rs=matrix_entries] [Gd=matrix_entries] [Rognd=val]
+ [Rsgnd=val] [Lgnd=val]
```

Argument	Description	Units
N=_val_	Sets positive non-zero integer _val_ as the number of signal conductors (excluding the reference conductor).	
Lo=_matrix_entries_	DC inductance matrix, per unit length.	H/m
Co=_matrix_entries_	DC capacitance matrix, per unit length.	F/m
Ro=_matrix_entries_	DC resistance matrix, per unit length.	Ω/m
Go=_matrix_entries_	DC shunt conductance matrix, per unit length.	S/m
Rs=_matrix_entries_	Skin effect resistance matrix, per unit length.	Ω/(m•√Hz)
Gd=_matrix_entries_	Dielectric loss conductance matrix, per unit length.	S/m•Hz
Lgnd=_val_	Defines _val_ as the DC inductance value, per unit length for ground (reference line).	H/m
Rognd=_val_	Defines _val_ as the DC resistance value, per unit length for ground (reference line).	Ω/m
Rsgnd=_val_	Defines _val_ as the skin effect resistance value, per unit length for ground (reference line).	Ω/(m•√Hz)

The following example shows RLGC input for the W-element:

```
* W-Element example, four conductors
W1 N=3 1 3 5 0 2 4 6 0 RLGCMODEL=example_rlc l=0.97
```

```
* RLGC matrices for a four-conductor lossy
.MODEL example_rlc W MODELTYPE=RLGC N=3
+ Lo=
+ 2.311e-6
+ 4.14e-7 2.988e-6
+ 8.42e-8 5.27e-7 2.813e-6
+ Co=
+ 2.392e-11
+ -5.41e-12 2.123e-11
+ -1.08e-12 -5.72e-12 2.447e-11
+ Ro=
+ 42.5
+ 0 41.0 + 0 0 33.5
+ Go=
+ 0.000609
+ -0.0001419 0.000599
+ -0.00002323 -0.00009 0.000502
+ Rs=
+ 0.00135
+ 0 0.001303
+ 0 0 0.001064
+ Gd=
+ 5.242e-13
+ -1.221e-13 5.164e-13
+ -1.999e-14 -7.747e-14 4.321e-13
```

RLGC matrices in the RLGC model of the W-element are in the Maxwellian format.

Format 2: Frequency-Dependent Tabular Specification

The tabular RLGC model may be used as an extension of the analytical RLGC model to model any arbitrary frequency-dependent behavior of transmission lines (this model does not support RC lines).

The W-element syntax supports tables of data (use a `.MODEL` statement of type w). To accomplish this, the `.MODEL` statement refers to .MODEL statements where the "type" is SP (described in Small-Signal Parameter Data Frequency Table Model (SP Model)), which contain the actual table data for the RLGC matrices.

The W-element tabular model requires the following:

- R and G tables shall include zero-frequency data points.
- L and C tables shall include infinite-frequency data points as well as zero-frequency data points.

To specify a zero-frequency point, either the DC argument shall be used or, alternatively, the f parameter in the DATA field of the SP model may be set to a value of 0. To specify an infinity frequency point, use the INFINITY argument.

See also, Small-Signal Parameter Data Frequency Table Model (SP Model).

11.9   Frequency-Dependent Matrices

The static (constant) L and C matrices are accurate for a wide range of frequencies. In contrast, the static (DC) R matrix applies to only a limited frequency range, mainly due to the skin effect. A good approximate expression of the R resistance matrix with the skin effect, is:

Equation 1
$$R(f) \cong R_o + \sqrt{f}(1+j)R_s$$

where:
- R_o is the DC resistance matrix.
- R_s is the skin effect matrix.

The imaginary term depicts the correct frequency response at high frequency; however, it might cause significant errors for low-frequency applications.

In contrast, the G (loss) conductance matrix is often approximated as:

Equation 2
$$G(f) \cong G_o + \frac{f}{\sqrt{1+\left(\frac{f}{f_{gd}}\right)^2}} G_d$$

Where,
- G_o models the shunt current due to free electrons in imperfect dielectrics.
- G_d models the power loss due to the rotation of dipoles under the alternating field.
- f_{gd} is a cut-off frequency.

If f_{gd} is not set, or if f_{gd} is set to 0, then G(f) keeps a linear dependency on the frequency. In the W-element, the default f_{gd} is zero (that is, G(f) does not use the f_{gd} value).

Both of these are ways to fit the RLGC model fit with actual measurements.
If measured or computationally extracted data is used for a tabular RLGC model, it should be more accurate if parameter extraction is accurately done.

Small-Signal Parameter Data Frequency Table Model (SP Model)

The small-signal parameter data frequency table model (SP model) is a generic model that describes frequency-varying behavior.

Syntax

```
.MODEL name sp N=val [FSTART=val] [FSTOP=val] [NI=val]
+ [SPACING=spacing] VALTYPE=REAL [INFINITY=matrixval]
+ [INTERPOLATION=val] [EXTRAPOLATION=val]
+ DATA=(npts ...) | DATAFILE=filename
```

Table 19: W-element SP Model Arguments

Argument	Description
N=val	Sets positive, non-zero integer val as the matrix dimension (number of transmission lines).
FSTART=val	Sets val as the starting frequency point for data. Default=0.
FSTOP=val	Sets val as the final frequency point for data. Use this parameter only for the **LINEAR** and **LOG** spacing formats.
NI=val	Number of frequency points per interval. Use this parameter only for the **DEC** and **OCT** spacing formats. Default=10.
SPACING=spacing	Sets spacing as one of the following data sample spacing formats: • **LIN**: uniform spacing with frequency step of (**FSTOP**-**FSTART**)/(npts-1). Default. The entry **LINEAR** is also a legal alternative. • **OCT**: octave variation with **FSTART** as the starting frequency and NI points per octave. npts sets the final frequency. • **DEC**: decade variation with **FSTART** as the starting frequency and NI points per decade. npts sets the final frequency. • **LOG**: logarithmic spacing. **FSTART** and **FSTOP**

	are the starting and final frequencies. • **POI**: non-uniform spacing. Pairs data points with frequency points. **NONUNIFORM** is an acceptable alternative entry.
VALTYPE=REAL	Sets data formatting to real values only (required for compatibility with existing SPICE implementations).
INFINITY=*val*	Sets *val* as the data point at infinity. Typically real-valued. This data format shall be consistent with the **MATRIX** and **VALTYPE** specifications. *npts* does not count this point.
INTERPOLATION=*val*	Sets *val* as the interpolation scheme: • **STEP**: piecewise step (default). • **LINEAR**: piecewise linear. • **SPLINE**: b-spline curve fit.
EXTRAPOLATION=*val*	Sets *val* as one of the following the extrapolation schemes to be used for the model data during simulation: • **NONE**: no extrapolation is allowed. Simulation terminates if a required data point is outside of the specified range. • **STEP**: uses the last boundary point. The default. • **LINEAR**: linear extrapolation by using the last two boundary points. If the data point at infinity is specified, then extrapolation is not used. Note that, while STEP is the default setting for INTERPOLATION, due to differences between implementations, the LINEAR setting is strongly recommended.

DATA= *npts …*	Data, where *npts* is the number of data points defined. The syntax to be used depends on spacing:
	• Syntax for **LIN** spacing:
	`.MODEL name sp SPACING=LIN [N=dim]`
	`+ FSTART=f0 DATA=npts d1 d2 ...`
	• Syntax for **OCT** or **DEC** spacing:
	`.MODEL name sp SPACING=DEC [N=dim]`
	`+ FSTART=f0 NI=n_per_intval`
	`+ DATA=npts d1 d2 ...`
	`.MODEL name sp SPACING=OCT [N=dim]`

	``
+ FSTART=f0 NI=n_per_intval
+ DATA=npts d1 d2 ...
``` • Syntax for **POI/NONUNIFORM** spacing: ```
.MODEL name sp SPACING=NONUNIFORM
+ [N=dim]DATA=npts f1 d1 f2 d2 ...
``
DATAFILE=*filename*	Sets filename as the name of an external file containing the data points.

An SP model definition shall use either **DATA** or **DATAFILE**, but not both.

If an external file is used for the SP model data points, this file shall contain only raw numbers without any suffixes, comments or continuation characters. The first number in the file shall be an integer value to indicate the number of sampling points in the file (*npts*). Then, sampling data shall follow. The order of sampling data shall be the same as in the DATA statement. This data file has no limitation on line length.

Note that, while STEP is the default setting for INTERPOLATION, due to differences between implementations, the LINEAR setting is strongly recommended.

W-element Model Definition Syntax

```
.MODEL name W MODELTYPE=TABLE N=val
+ LMODEL=l_freq_model CMODEL=c_freq_model
+ [RMODEL=r_freq_model] [GMODEL=g_freq_model]
```

Table 20: W-element Model Definition Arguments

Argument	Description
N=*val*	Sets *val* as the positive, non-zero integer number of signal conductors (excluding the reference conductor).
LMODEL=*l_freq_model*	Sets *l_freq_model* as the SP model name for the inductance matrix array.
CMODEL= *c_freq_model*	Sets *c_freq_model* as the SP model name for the capacitance matrix array.
RMODEL=*r_freq_model*	Sets *r_freq_model* as the SP model name for the resistance matrix array. By default, it is zero.

GMODEL=_g_freq_model_	Sets _g_freq_model_ as the SP model name for the conductance matrix array. By default, it is zero.

11.10 S-element

An S-element is a frequency-domain set of network data, described using scattering parameters.

Syntax

```
Sxxx n1 n2 ... nN [nRef] MNAME=Smodel_name
```

The name of an S-element instance shall begin with the character "S", followed by up to 1023 characters (see Table 3: IBIS-ISS Special Characters for a list of valid special characters).

Table 21: S-element Arguments

Argument	Description
n1 n2...nN	Nodes of an S-element. Three kinds of definitions are permitted: ■ With no reference node nRef, the default reference node is GND. Each node ni (i=1~N) and GND construct one of the N ports of the S-element. ■ With one reference node, nRef is defined. Each node ni (i=1~N) and the nRef construct one of the N ports of the S-element. ■ With N reference nodes, each port has its own reference node. The node definition may be written more clearly: n1+ n1- n2+ n2- ... nN+ nN- Each pair of the nodes (ni+ and ni-, i=1~N) constructs one of the N ports of the S-element.
nRef	Reference node
MNAME=Smodel_name	Sets Smodel_name as the name of the S model. Note that string parameters are supported in calling an **MNAME**.

The nodes of the S-element shall be placed immediately after the identifier string.

The following example illustrates the *nd1 nd2... ndN* no reference, single reference, and multi-reference parameters.

```
**S-parameter example

* no reference
S_no_ref n1 n2 mname=s_model

* single reference
S_one_ref n1 n3 gnd mname=s_model

*multi-reference
S_multi_ref n1 gnd n4 gnd mname=s_model
```

IBIS-ISS will match the S-element instance node definitions to the number of ports *n* specified in the S-element .MODEL definition, as follows.

- For *n* nodes, the S-element assumes no reference node.

- For *n*+1 nodes, the S-element assumes one reference node.

- For 2*n* nodes, the S-element assumes *n* signal nodes and *n* reference nodes. Each pair of nodes comprises a signal and a reference node.

S-Element Model Syntax

Use the following syntax to describe specific S-element models:

```
.MODEL Smodel_name S N=val TSTONEFILE=filename
```

Argument	Description
Smodel_name	Name of the S model.
S	Specifies that the model type is an S model.
N=*val*	Sets positive, non-zero integer *val* as the number of ports for the S-model. This value must match the number of ports defined in the associated Touchstone file.

TSTONEFILE=*filename*	Sets the string *filename* as the name of a Touchstone file. Note that string parameters are supported for TSTONEFILE Example:
	
	`.subckt sparam n1 n2 tsfile=str('ss_ts.s2p')`
	`S1 n1 n2 0 mname=s_model`
	`.model s_model S TSTONEFILE=str(tsfile)`
	`.ends`
	`x1 A B sparam tsfile=str('ss_ts.s2p')`
	`...`
	
	For details, see *Touchstone® File Format Specification* by the IBIS Open Forum (http://www.eda.org/ibis/).

11.11   E-element (Voltage-Controlled Voltage Source)

This section explains the E-element syntax and parameters.

Syntax (Linear Form)

`Exxx n+ n- [VCVS] in+ in- gain`

Syntax (Laplace Transform)

`Exxx n+ n- LAPLACE in+ in-  k0, k1, ..., kn / d0,`
`+ d1, ..., dm`

Syntax (Pole-Zero Function)

`Exxx n+ n- POLE in+ in- a αz1, fz1, ..., αzn, fzn / b,`
`+ αp1, fp1, ..., αpm, fpm`

Syntax (Foster Pole-Residue Form)

`Exxx n+ n- FOSTER in+ in- k0 k1`
`+ (Re{A1}, Im{A1})/ (Re{p1}, Im{p1})`
`+ (Re{A2}, Im{A2})/ (Re{p2}, Im{p2})`
`+ (Re{A3}, Im{A3})/ (Re{p3}, Im{p3})`
`+ ...`

E-element Arguments

The E-element arguments are described in the following list. Note that the element name shall begin with the character "E" and be followed by up to 1023 characters (see Table 3: IBIS-ISS Special Characters for a list of valid special characters).

Table 23: E-element Arguments

Argument	Description
gain	Voltage gain

in+/-	Positive or negative controlling nodes. Specify one pair for each dimension
n+/-	Positive or negative node of a controlled element
VCVS	Identifier argument for voltage-controlled voltage source. VCVS is a reserved word; do not use it as a node or parameter name

Laplace Transform Details

H(s) is a rational function, with parameters used to define the values of all coefficients $(k_0, k_1, ..., d_0, d_1, ...)$.

Pole-Zero Function Details

The following equation defines H(s) in terms of poles and zeros:

$$H(s) = \frac{a \cdot (s + \alpha_{z1} - j2\pi f_{z1})...(s + \alpha_{zn} - j2\pi f_{zn})(s + \alpha_{zn} + j2\pi f_{zn})}{b \cdot (s + \alpha_{p1} - j2\pi f_{p1})...(s + \alpha_{pm} - j2\pi f_{pm})(s + \alpha_{pm} + j2\pi f_{pm})}$$

Equation 3

The complex poles or zeros are in conjugate pairs. The element description specifies only one of them, and the program includes the conjugate. Parameters may be used to specify the a, b, α, and f values.

```
Elow_pass out 0 POLE in 0 1.0 / 1.0, 1.0,0.0 0.5,0.1379
```

The `Elow_pass` statement describes a low-pass filter, with the transfer function:

$$H(s) = \frac{1.0}{1.0 \cdot (s + 1)(s + 0.5 + j2\pi \cdot 0.1379)(s + 0.5 - (j2\pi \cdot 0.1379))}$$

Foster Pole-Residue Details

The following equation defines H(s) in terms of poles and residues:

$$H(s) = k_0 + k_1 s + \sum_i \left(\frac{Re\{A_i\} + j \cdot Im\{A_i\}}{s - (Re\{p_i\} + j \cdot Im\{p_i\})} + \frac{Re\{A_i\} - j \cdot Im\{A_i\}}{s - (Re\{p_i\} - j \cdot Im\{p_i\})} \right)$$

In the Foster pole-residue syntax, parentheses, commas, and slashes are separators—they have the same meaning as whitespace. A pole-residue pair is represented by four numbers (real and imaginary part of the residue, then real and imaginary part of the pole).

For convergence, the $Re\{p_i\}$ shall be less than zero.

For example, to represent a E(s) equation in the form

$$E(s) = 0.001 + 1 \times 10^{-12}s + \frac{0.0008}{s + 1 \times 10^{10}} + \frac{(0.001 - j0.006)}{s - (-1 \times 10^8 + j1.8 \times 10^{10})}$$

$$+ \frac{(0.001 + j0.006)}{s - (-1 \times 10^8 - j1.8 \times 10^{10})}$$

The IBIS-ISS syntax would be:

```
E1 1 0 FOSTER 2 0 0.001 1e-12
+ (0.0004, 0)/(-1e10, 0) (0.001, -0.006)/(-1e8, 1.8e10)
```

Note:

For real poles, half the residue value is entered because it is applied twice. In the above example, the first pole-residue pair is real, but is written as "A1/(s-p1)+A1/(s-p1)"; therefore, 0.0004 is entered rather than 0.0008.

11.12   F-element (Current-Controlled Current Source)

This section explains the F-element syntax and parameters.

Syntax

```
Fxxx n+ n- [CCCS] vn1 gain
```

F-element Arguments

The F-element parameters are described in the following list. Note that the F-element name shall begin with the character "F", followed by up to 1023 characters (see Table 3: IBIS-ISS Special Characters for a list of valid special characters).

Table 24: F-element Arguments

Argument	Description
CCCS	Identifier argument for current-controlled current source. CCCS is an IBIS-ISS reserved word; do not use it as a node or parameter name.
gain	Current gain.
n+/-	Connecting nodes for a positive or negative controlled source.
vn1	Name of the voltage source through which the controlling current flows.

Note that the controlling current flows from the positive node of source *vn1* to the negative node of source *vn1*, per the definition of *vn1* elsewhere in the circuit.

11.13   G-element (Voltage-Controlled Current Source)

This section explains G-element syntax statements, and their parameters.

Syntax (Linear Form)

`Gxxx n+ n- [`**`VCCS`**`] in+ in- transconductance`

Syntax (Laplace Transform)

`Gxxx n+ n- `**`LAPLACE`**` in+ in-   k0, k1, ..., kn / d0,`
`+ d1, ..., dm`

Syntax (Pole-Zero Function)

`Gxxx n+ n- `**`POLE`**` in+ in- a αz1, fz1, ..., αzn, fzn / b,`
`+ αp1, fp1, ..., αpm, fpm`

Syntax (Foster Pole-Residue)

`Gxxx n+ n- `**`FOSTER`**` in+ in- k0 k1`
`+ (Re{A1}, Im{A1})/ (Re{p1}, Im{p1})`
`+ (Re{A2}, Im{A2})/ (Re{p2}, Im{p2})`
`+ (Re{A3}, Im{A3})/ (Re{p3}, Im{p3})`
`+ ...`

G-element Arguments

The G-element arguments are described in the following list. Note that the G-element name shall begin with the character "G", followed by up to 1023 characters (see Table 3: IBIS-ISS Special Characters for a list of valid special characters).

Table 25: G-element Arguments

Argument	Description
in+/-	Positive or negative controlling nodes. Specify one pair for each dimension.

n+/-	Positive or negative node of the controlled element. Current flows from the positive to the negative node.
transconductance	Voltage-to-current conversion factor.
VCCS	Identifier argument for the voltage-controlled current source. VCCS is a reserved IBIS-ISS word; do not use it as a node or parameter name.

Laplace Transform Details

For the Laplace transform, the transconductance, H(s), is a rational function, in the following form:

$$H(s) = \frac{k_0 + k_1 s + \ldots + k_n s^n}{d_0 + d_1 s + \ldots + d_m s^m}$$

Equation 4

Parameters may be used to define the values of all coefficients (k_0, k_1, ..., d_0, d_1, ...).

Pole-Zero Function Details

For the pole-zero function, the following equation defines H(s) in terms of poles and zeros:

$$H(s) = \frac{a \cdot (s + \alpha_{z1} - j2\pi f_{z1}) \ldots (s + \alpha_{zn} - j2\pi f_{zn})(s + \alpha_{zn} + j2\pi f_{zn})}{b \cdot (s + \alpha_{p1} - j2\pi f_{p1}) \ldots (s + \alpha_{pm} - j2\pi f_{pm})(s + \alpha_{pm} + j2\pi f_{pm})}$$

Equation 5

The complex poles or zeros are in conjugate pairs. The element description specifies only one of them, and the simulation program will include the conjugate automatically. Parameters may be used to specify the a, b, α, and f values. An example is shown below.

```
Ghigh_pass 0 out POLE in 0 1.0 0.0,0.0 / 1.0 0.001,0.0
```

The `Ghigh_pass` statement describes a high-pass filter, with the transfer function:

$$H(s) = \frac{1.0 \cdot (s + 0.0 + j \cdot 0.0)}{1.0 \cdot (s + 0.001 + j \cdot 0.0)}$$

Foster Pole-Residue Form

The following equation defines H(s) in terms of poles and residues:

$$H(s) = k_0 + k_1 s + \sum_i \left(\frac{Re\{A_i\} + j \cdot Im\{A_i\}}{s - (Re\{p_i\} + j \cdot Im\{p_i\})} + \frac{Re\{A_i\} - j \cdot Im\{A_i\}}{s - (Re\{p_i\} - j \cdot Im\{p_i\})} \right)$$

In the Foster pole-residue syntax, parentheses, commas, and slashes are separators—they have the same meaning as a space. A pole-residue pair is represented by four numbers (real and imaginary part of the residue, then real and imaginary part of the pole).

For convergence, the *Re{p$_i$}* shall be less than zero.

For example, to represent a G(s) in the form,

$$G(s) = 0.001 + 1 \times 10^{-12} s + \frac{0.0008}{s + 1 \times 10^{10}} + \frac{(0.001 - j0.006)}{s - (-1 \times 10^8 + j1.8 \times 10^{10})} +$$

$$\frac{(0.001 + j0.006)}{s - (-1 \times 10^8 - j1.8 \times 10^{10})}$$

The IBIS-ISS syntax would be:

```
G1 1 0 FOSTER 2 0 0.001 1e-12
+(0.0004, 0)/(-1e10, 0) (0.001, -0.006)/(-1e8, 1.8e10)
```

Note:

> For real poles, half the residue value is entered because it is applied twice. In the above example, the first pole-residue pair is real, but is written as "A1/(s-p1)+A1/(s-p1)"; therefore, 0.0004 is entered rather than 0.0008.

11.14   H-element (Current-Controlled Voltage Source)

This section explains H-element syntax statements, and defines their parameters.

Syntax

```
Hxxx n+ n- [CCVS] vn1 transresistance
```

Argument	Description
CCVS	Identifier argument for the current-controlled voltage source. CCVS is an IBIS-ISS reserved word; do not use it as a node or parameter name.
n+/-	Connecting nodes for positive or negative controlled source.
transresistance	Current-to-voltage conversion factor.
vn1	Name of voltage source through which controlling current flows.

Note that the controlling current flows from the positive node of source *vn1* to the negative node of source *vn1*, per the definition of *vn1* elsewhere in the circuit.

12 Best Practices

This section details syntax recommendations for ensuring maximum compatibility with existing proprietary SPICE variants. While not requirements for IBIS-ISS, following these practices will help maintain the portability of IBIS-ISS files.

- Exponent ranges should be limited to between e-60 and e+60.

- The use of a single backslash character (\) as a line continuation character is not supported in IBIS-ISS.

- The use of an "X" character as the "Meg" scale factor is not supported in IBIS-ISS.

- For maximum compatibility, in-line comments may be preceded by the * (asterisk) character instead of the $ (dollar) character. However, while not illegal, this practice should be avoided.

- Instance names shall begin with the appropriate identifying character. The remaining characters of a instance name should be limited to upper- and lower-case alphabetic characters, numeric characters 0-9 and the characters:
    ~!@#%&_<>?[]|:;

- Parameter names should begin with a-z or A-Z, and the remaining characters should be limited to upper- and lower-case alphabetic characters, numeric characters 0-9 and the characters:
    ! # $ % [ ]

- While a parameter may be defined in more than one .param statement within a subcircuit, this practice is best avoided.

- Node names should either be all numeric (0-9), or follow the rules of instance names stated above.

13 References

1. Luca Daniel and Joel Philips, "Model Order Reduction for Strictly Passive and Causal Distributed Systems". Proceeding of DAC, 2002.

2. Colin Gordon, Thomas Blazeck, and Raj Mittra. "Time-Domain Simulation of Multiconductor Transmission Lines with Frequency-Dependent Losses." *IEEE Transactions on Computing-Aided Design*, vol. II, No. 11, pp. 1372-1387, 1992.

3. Omar Wing and Qingjian Yu, "Computational Models of Transmission Lines with Skin Effects and Dielectric Loss," *IEEE Transactions on Circuits and Systems-I: Fundamental Theory and Applications,* Vols. 41, No. 2, pp. 107-119, 1994.

4. Bjorn Gustavsen and Adam Semlyen, *"*Rational Approximation of Frequency Domain Responses by Vector Fitting." *IEEE Transaction on Power Delivery*. Vols. 14, No. 3, pp. 1052-1061, 1999.

5. E. Hammerstad, E. and O. Jensen, "Accurate models for microstrip computer aided design." *IEEE MTT-S Int. Microwave Symp. Dig.* pp. 407-409, 1980.

6. C.A. Balanis, *Advanced Engineering Electromagnetics.* New York : Wiley, 1989.

7. Eric Bogatin. *Signal Integrity Simplified.* Upper Saddle River : Prentice Hall, 2003.

8. Ravi Kollipara, et al. "Practical Design Considerations for 10 to 25 Gbps Copper Backplane Serial Links." DesignCon, 2006.

9. *HSPICE® User Guide: Simulation and Analysis E-2010.12*, Mountain View, CA : Synopsys, Inc., 2010.

10. *HSPICE® User Guide: Signal Integrity E-2010.12*, Mountain View, CA : Synopsys, Inc., 2010.