|==========

List of all current IBIS-EBD keywords:

|==========

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

[Begin Board Description] 16Meg X 8 SIMM Module
[Manufacturer] Quality SIMM Corp.
[Number Of Pins] 128
[Pin List] signal_name

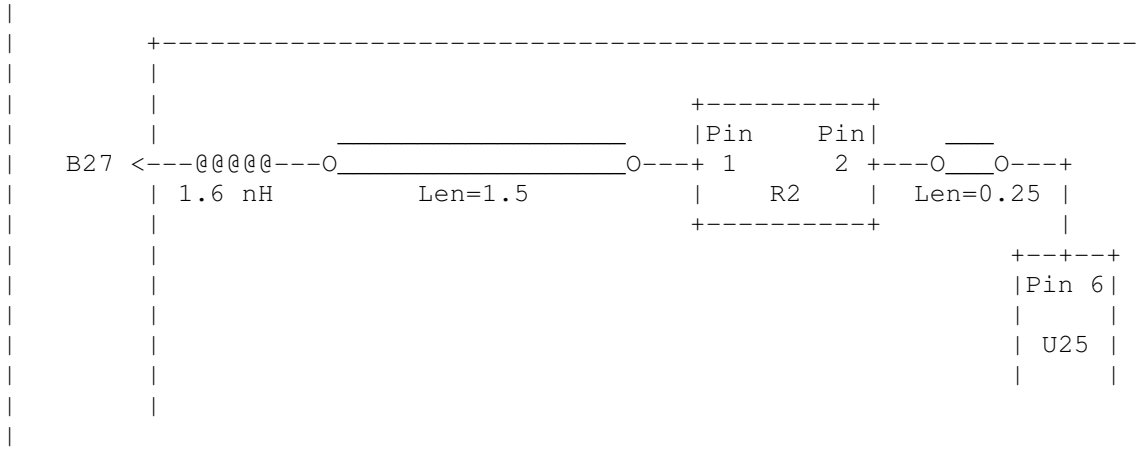
```
 A1          GND
 A2          data1
 A3          data2
 A4          POWER5    | This pin connects to 5 V
 A5          NC        | a no connect pin
| .
| .
 A22         POWER3.3  | This pin connects to 3.3 V
 B1          casa
| .
| .
```

|----------
| 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
|
|        +-----------------------------------------------------------
|        |
|        |          _______          _______          _______
|   J25 <---------O_______O---+---O_______O---+---O_______O---+
|        |         Len=0.5    |    Len=0.5    |    Len=0.5    |
|        |                    |               |               |
|        |                 +--+--+         +--+--+         +--+--+
|        |                 |Pin15|         |Pin15|         |Pin15|
|        |                 |     |         |     |         |     |
|        |                 | U21 |         | U22 |         | U23 |
|        |                 |     |         |     |         |     |
|        |
|
```

```
|-------------------------------------------------------------------------
|  A Description Using The Fork and Endfork Subparameters:
|
[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
|
|        +-----------------------------------------------------------
|        |
|        |                  _________________
|    A5 <-----------------O_________________O-------------+
|        |                     Len=1.0                     |
|        |                                                 |
|        |             ____________________________        |
|    B5 <---@@@@@---O____________________________________O---+
|        |  2 nH                Len=2.1                    |
|        |                                                 |
|        |                                 _____________   |
|        |                      +---O_____________O---+
|        |                      |       Len=1.0
|        |                      |
|        |                   +--+--+
|        |                   |Pin16|
|        |                   |     |
|        |                   | U23 |
|        |                   |     |
|        |
|
```

```
|-------------------------------------------------------------------------------
|  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
|
```




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

From Walter:

A “Free Form” view specifies the name of the file and subckt, and the “Sub-Language” that the subckt is written in. The only requirement is that a Sub-Language must be documented with the “spice” elements that are being used. The entity can document these elements by simply pointing to a proprietary simulator manual. The Sub-Languages and their descriptions will not be approved or controlled by IBIS. It would be nice if IBIS could maintain a WEB site with Sub-Language documentation. Creators of EMD models will be encouraged to use a small set of Sub-Languages.

“Free Form” is the way that it is being done today. EMD now becomes a standard way of wrapping “Free Form” models.

The following is an example EMD file with “Free Form” models for a four-pin connector.

```
[Module] 4_pin_example
[mPin]
J1.1 Tx+
J1.2 Tx-
J1.3 Rx+
J1.4 Rx-
J2.1 Rx+
J2.2 Rx-
J2.3 Tx+
J2.4 Tx-

[Diff mPins]
J1.1 J1.2
J1.3 J1.4
J2.1 J2.2
J2.3 J2.4

[Extended Nets]
write J1.1 J1.2 J2.1 J2.2
read  J1.3 J1.4 J2.3 J2.4

[View] uncoupled_IBIS_Macro 5GHz IBIS_Macro
write J1.1 J1.2 J2.1 J2.2 file=IBIS_Macro_uncoupled.mod model=write
read  J1.3 J1.4 J2.3 J2.4 file=IBIS_Macro_uncoupled.mod model=read
[End View]

[View] coupled_IBIS_Macro 5GHz IBIS_Spice
framis J1.1 J1.2 J2.1 J2.2 J1.3 J1.4 J2.3 J2.4 file=IBIS_Spice.mod model=framis
[End View]

[View] coupled_ICM 5GHz ICM_rlgc
framis J1.1 J1.2 J2.1 J2.2 J1.3 J1.4 J2.3 J2.4 file=rlgc.icm model=framis
[End View]

[View] coupled_ICM 5GHz ICM_sNp
framis J1.1 J1.2 J2.1 J2.2 J1.3 J1.4 J2.3 J2.4 file=sNp.icm model=framis
[End View]

[View] uncoupled_pole-zero 2GHz Verilog_AMS
write J1.1 J1.2 J2.1 J2.2 file=Verilog_AMS_uncoupled.mod model=write
read  J1.3 J1.4 J2.3 J2.4 file=Verilog_AMS_uncoupled.mod model=read
[End View]
```

```
[View] coupled_pole-zero 2GHz VHDL_AMS
framis J1.1 J1.2 J2.1 J2.2 J1.3 J1.4 J2.3 J2.4 file=VHDL_AMS.mod model=framis
[End View]

[View] uncoupled_sNp 10GHz EDA_Vendor_1_Extraction
framis J1.1 J1.2 J2.1 J2.2 J1.3 J1.4 J2.3 J2.4 file=sNp_uncoupled.mod
model=framis
[End View]

[View] coupled_sNp 10GHz EDA_Vendor_2_Extraction
framis J1.1 J1.2 J2.1 J2.2 J1.3 J1.4 J2.3 J2.4 file=sNp_coupled.mod model=framis
[End View]

[End Module]
```

Resemblances between the two:

```
[Module] 4_pin_example


[mPin]
J1.1 Tx+
J1.2 Tx-
J1.3 Rx+
J1.4 Rx-
J2.1 Rx+
J2.2 Rx-
J2.3 Tx+
J2.4 Tx-

[Diff mPins]
J1.1 J1.2
J1.3 J1.4
J2.1 J2.2
J2.3 J2.4

[Extended Nets]
write J1.1 J1.2 J2.1 J2.2
read J1.3 j1.4 J2.3 J2.4
```

```
[View] uncoupled_IBIS_Macro 5GHz IBIS_Macro
write J1.1 J1.2 J2.1 J2.2 file=IBIS_Macro_uncoupled.mod model=write
read  J1.3 J1.4 J2.3 J2.4 file=IBIS_Macro_uncoupled.mod model=read
[End View]
```

```
[Begin Board Description]  16Meg X 8 SIMM Module
[Manufacturer] Quality SIMM Corp.
[Number Of Pins] 128
[Pin List]  signal_name
 1          Tx+
 2          Tx-
 3          Rx+
 4          Rx-
 …
 …
 …
```

```
[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
```

[Reference Designator Map]

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

Proposal:

1) Remove the EBD section from the IBIS specification and make it an independent specification of its own, as ICM.

Reason: This is needed to establish a hierarchy between the various models which are instantiated to describe a design. The EBD file would serve as a (top level) description for a board (netlist), instantiating driver(s), receiver(s), package(s), connector(s), etc… instances and/or other instances of (nested) EBD files. These instantiations may be references to [Model]s in .IBS files, or [Define Package Model]s in .PKG files, or [Begin ICM Model] in .ICM files, or [xxx] in .EBD files, etc… Details and specifics TBD.

2) Implement the following

Multi-line keywords will always start with [Begin Keyword Name] and finish with [End Keyword Name]. This allows for nesting keywords within other keywords and also makes parsing easier. Single line keywords will use the [Keyword Name] format, and must be no more than one line long.

We should start the file with the usual bookkeeping keywords, such as `[File Name]`, `[Version]`, `[Date]`, etc… Multiple `[Begin Module]` and `[End Module]` keywords within a single file should be permitted.

```
[Begin Header]
[EBD Ver]
[File Name]
[File Rev]
[Date]
[Source]
[Begin Notes]  [End Notes]
[Begin Disclaimer] [End Disclaimer]
[Begin Copyright] [End Copyright]
[Support]                    | Do we need this?
[Redistribution]             | Do we need this?
[Redistribution Text]        | Do we need this?
[End Header]

[Comment Char]


[Begin Module] ModuleName

[Begin Terminal List]        | Connection points to the outside world
InstanceName1.NodeName1   SignalName1
InstanceName1.NodeName2   SignalName2
InstanceName2.NodeName1   SignalName3
InstanceName2.NodeName2   SignalName4
etc…
[End Terminal List]

[Begin Parameter List]       | Parameters that can be passed into the module
ParameterName1 = DefaultValue1
ParameterName2 = DefaultValue2
[End Parameter List]
```

```
[Begin Reference Designator List]
InstanceName1  File = filename.ibs   Component = ComponentName
InstanceName2  File = FileName.icm   ICMmodel  = ICMfamilyName(ICMmodelName)
InstanceName3  File = FileName.ebd   Module    = ModuleName(ViewName)
InstanceName4  File = FileName.vhd   Entity    = EntityName(ArchitectureName)
InstanceName5  File = FileName.vams  Module    = ModuleName
InstanceName6  File = FileName.sp    SubCkt    = SubcircuitName
[End Reference Designator List]
```

Multiple `[Begin View]` and `[End View]` keywords within a module section should be permitted.

```
[Begin View]  ViewName1
MaxFreq = Value
Corner = Value
etc…

[Begin Diff Pin List]   | Not sure about this yet (may need to go outside [View]
(InstanceName1.NodeName1, InstanceName1.NodeName2)
(InstanceName1.NodeName3, InstanceName1.NodeName4)
(InstanceName2.NodeName1, InstanceName2.NodeName2)
(InstanceName2.NodeName3, InstanceName2.NodeName4)
[End Diff Pin List]

[Begin Extended Nets List]   | Not sure about this yet (put it outside [View]?
ExtendedNetName1(InstanceName1.NodeName1, InstanceName2.NodeName1, etc…)
ExtendedNetName2(InstanceName1.NodeName1, InstanceName2.NodeName1, etc…)
[End Extended Nets List]
```

The following lines illustrate how the "circuit elements" are instantiated and a netlist is formed. The first example uses the "parameter assignment by order" notation:

```
InstanceName   Terminals(NodeName1, NodeName2, NodeName3, …)
               Parameters(Value1, Value2, Value3, …);
```

The next instantiation example uses the "explicit parameter assignment" notation:

```
InstanceName   Terminals(NodeName1, NodeName2, NodeName3, …)
               Parameters(ParameterName2InsideModule = Value2,
                          ParameterName5InsideModule = Value4);
```

Note that there is only one format for the `Terminals` list. The following rules apply to the node names of the `Terminals` list. These rules make the "black box" approach possible, which means that no inside information is needed about the terminal names from within the model that is being instantiated. (Should we consider eliminating the "explicit parameter assignment" notation for this reason, since it requires the knowledge of the parameter names that are inside a model description)?

1. The node names (`NodeName1`, `NodeName2`, etc…) do not have to match the node names inside the model of the element that is being instantiated. The connections will always be made "by order" to the terminals of the element model.
2. The scope of these node names is local to the Terminals list within an element instantiation statement, i.e. two identical node names in two different instantiation statements do not imply that they are connected with an ideal short (as opposed to the way it is done in SPICE). In order to

make such connections, a separate keyword called `[Connection Map]` is needed which lists groups of `NodeName`-s using the full "dot notation" (`InstanceName.NodeName`) syntax.

If parameter passing is done by order, the parameter values must to be listed in the order that corresponds to the order of the parameters in the module description. If fewer parameters are supplied in the instantiation, the remaining parameters in the module will refer to their default values if defined in the module, or remain unassigned (an error condition). If more parameters are supplied than defined in the module, the extra parameters will be ignored.

If the explicit parameter passing syntax is used, the instantiation statement may supply any number of matching parameters in any order, but parameters which do not have defaults in the module description and do not receive passed down values will remain unassigned (an error condition).

```
[Begin Connection Map]
(InstanceName1.NodeName1, InstanceName2.NodeName1, etc…)
(InstanceName1.NodeName2, InstanceName2.NodeName2, etc…) etc…
[End Connection Map]
```

All node name references must use the full "dot notation" syntax. Each node within a pair of parentheses will be connected with an ideal short.

```
[End View] ViewName

[End Module] ModuleName
```

Example:

```
…
…
[Begin Module] ExamplePluginCard

[Begin Terminal List]        | Connection points to the outside world
J1.A1   Tx+
J1.A2   Tx-
J1.A3   Power1
J1.B1   NC
J1.B2   NC
J1.B3   Power2
J2.1    Rx+
J2.2    Rx-
J2.3    NC
J2.4    NC
[End Terminal List]

[Begin Parameter List]       | Parameters that can be passed into the module
Zo = 50 Ohms
Length_T1 = 200 mil
Length_T2 = 500 mil
Length_T3 = 900 mil
Rterm = 50
[End Parameter List]

[Begin Reference Designator List]
U1  File = filename.ibs   Component = ComponentName
```

```
U2  File = filename.ibs   Component = ComponentName
P1  File = FileName.icm   ICMmodel  = ICMfamilyName(ICMmodelName)
J1  File = FileName.ebd   Module    = ModuleName(ViewName)
U1  File = FileName.vhd   Entity    = EntityName(ArchitectureName)
U2  File = FileName.vhd   Entity    = EntityName(ArchitectureName)
U1  File = FileName.vams  Module    = ModuleName
U2  File = FileName.vams  Module    = ModuleName
P1  File = FileName.sp    SubCkt    = SubcircuitName
[End Reference Designator List]

[Begin View]  ViewName1
MaxFreq = 1 GHz
Corner = Typical
etc…

[Begin Diff Pin List]   | Not sure about this yet (may need to go outside [View]
J1.A1   J1.A2
J2.1    J2.2
[End Diff Pin List]

[Begin Extended Nets List]   | Not sure about this yet (put it outside [View]?
ExtendedNetName1(InstanceName1.NodeName1, InstanceName2.NodeName1, etc…)
ExtendedNetName2(InstanceName1.NodeName1, InstanceName2.NodeName1, etc…)
[End Extended Nets List]

J1   Terminals(A1, A2, A3, B1, B2, B3);

T1   Terminals(n11, n12, n13, n14, n21, n22, n23, n24)
     Parameters(Zo, Length_T1);

R1   Terminals(1, 2)  Parameters(Rterm);
R2   Terminals(1, 2)  Parameters(Rterm);

T2   Terminals(n11, n12, n13, n14, n21, n22, n23, n24)
     Parameters(Zo, Length_T2);

P1   Terminals(NodeMapName1(SigP_pin, SigN_pin, Power1_pin, Power3_pin)
               NodeMapName2(SigP_pad, SigN_pad, Power1_pad, Power3_pad)
               NodeMapName3(SigP_pad, SigN_pad, Power1_pad, Power3_pad));

U1   Terminals(SigP, SigN, Power1, Power3);
U2   Terminals(SigP, SigN, Power1, Power3);

T3   Terminals(n11, n12, n21, n22)
     Parameters(Zo, Length_T3);

J2   Terminals(1, 2, 3, 4);

[Begin Connection Map]
(J1.A1, T1.n11)              | Put T1 between J1 and R1/R2
(J1.A2, T1.n12)
(J1.A3, T1.n13)
(J1.B3, T1.n14)

(T1.n21, R1.1)               | Put series resistors between T1 and T2
(T1.n22, R2.1)
(R1.2, T2.n11)
```

```
(R2.2, T2.n12)
(T1.n23, T2.n13)
(T1.n24, T2.n14)

(T2.n21, P1.NodeMapName1.SigP_pin)     | Put T2 between R1/R2 and P1
(T2.n22, P1.NodeMapName1.SigN_pin)
(T2.n23, P1.NodeMapName1.Power1_pin)
(T2.n24, P1.NodeMapName1.Power2_pin)

(P1.NodeMapName2.SigP_pad, U1.SigP)    | Place U1 into P1
(P1.NodeMapName2.SigN_pad, U1.SigN)
(P1.NodeMapName2.Power1_pad, U1.Power1)
(P1.NodeMapName2.Power2_pad, U1.Power2)

(P1.NodeMapName3.SigP_pad, U2.SigP)    | Place U2 into P1 (a stacked die)
(P1.NodeMapName3.SigN_pad, U2.SigN)
(P1.NodeMapName3.Power1_pad, U2.Power1)
(P1.NodeMapName3.Power2_pad, U2.Power2)

(T2.n21, T3.n11)                       | Put T3 between P1 and J2
(T2.n22, T3.n12)
(T3.n21, J2.1)
(T3.n22, J2.2)

[End Connection Map]

[End View] ViewName
[End Module] ExamplePluginCard
```