****
****

Buffer Issue Resolution Document (BIRD)
BIRD ID#: {TBD}
ISSUE TITLE: Clarification of the Table Format for IBIS_AMI.
REQUESTOR: Bob Ross, Teraspeed Consulting Group
DATE SUBMITTED:
DATE REVISED: May 6, 2011
DATE ACCEPTED BY IBIS OPEN FORUM:

****
****

STATEMENT OF THE ISSUE:

The definition of Format Table is unclear when there is only one row in the Table in the 5.0 version of the IBIS spec, and what the EDA tool needs to do if the parameter is Reserved or Model_Specific, and Usage is Info, In, Out, or InOut.

Also, the exiting Type selection rule technically limits Table columns to a single type selection (without using stated or implied inconsistent interpretations). There is need for having columns with different type selections for current and future generality. Single valued Type is still supported, but a multiple Type for Tables only are also introduced.

****

STATEMENT OF THE RESOLVED SPECIFICATIONS:

On pg. 140 replace the following lines:

```
|     Table     The parameter name "Table" names a branch of the parameter
|               tree rather than a single leaf.  One of the leaves of this
|               branch can be named "Labels" and, if provided, is to be
|               assigned a string value containing a list of column names.
|               For example:
|
|                 (Rx_Clock_PDF
|                   (Usage Info)
|                   (Type Float)
|                   (Format Table
|                     (Labels Row_No Time_UI Density)
|                     (-50 -0.1 1e-35)
|                     (-49 -0.98 2e-35)
|                     ...
|                     (0 0 1e-2)
|                     ...
|                     (49 0.98 2e-35)
|                     (50 0.1 1e-35)
|                   ) | End Table
|                 ) | End Rx_Clock_PDF
|
```

with these lines:

Table:

The Format Table states that this parameter consists of one or more columns of data, with each row delimited by parenthesis '(' and ')'. All rows must contain the same number of entries (columns). At least one row must be included. Default is illegal when Format Table is used.

The column entries must be of Type Float, UI, Integer, String or Boolean. Type Tap is illegal. If only one Type is provided, then all Table entries shall be of the specified type.

(Type <type>)

For Table only, Type can also be used to designate the entries for each column. In this case type entries shall be given for each column in the Table:

(Type <type1> <type2> <type3> ...)

Labels is an optional leaf within Table and it is followed by a String entry for each column in the Table. Quoted null entries are permitted. Labels shall be positioned immediately before the first row in a Table and are of the form:

(Labels **<"label1"> <"label2"> <"label3">** ...)

If Table is used for a reserved parameter, the rules for the number of columns and their meaning are described in the Reserved_Parameters section.

The EDA tool and the algorithmic model must always transmit the entire content of a table through the AMI_parameters_in or AMI_parameters_out string (defined in Section 10 and illustrated in the examples below). Only the parameter_name and values in the table are included in the parameter string. The values in each row of the table are flattened into a single row of values without the parentheses surrounding each row when producing the parameter string.

For Usage Out and InOut, the number of rows returned by the executable model may differ from the number of rows documented in the .ami file, but a minimum of one row must be returned all the time. Multiple GetWave calls are not required to return the same number of rows. **For Usage Out, a one-row Table is required in the .ami file to serve as a template for single and multi-row tables. This can be used by the EDA tool to reconstruct a sequence of data values returned by the executable model into a table with as many rows as needed, and optionally for parameter initialization before being replaced by the actual Table data returned by the executable model.**

Examples:

Single Row Table where all numbers are Float: Note, '1' is a legal float entry.

```
(fwd (Usage In) (Type Float)
      (Table
        (1 -0.169324  1.40308  0.33024)
```

```
      )
      (Description "Application Description")
)
```

The EDA tool sends to the executable model in the parameter string:

```
      (fwd  1 -0.169324  1.40308  0.33024)
```

Single Row, all numbers would be encoded as integers by the EDA tool:

```
(bit_pattern (Usage In) (Type Integer)
      (Table
        (1 1 1 1 0 0 0 1 0 0 1)
      )
      (Description "Bit Pattern Sequence")
)
```

The EDA tool sends to the executable model in the parameter string:

```
      (bit_pattern  1 1 1 1 0 0 0 1 0 0 1)
```

Multiple row Table application with Labels: The optional Labels line is added above the first row. It is not sent or returned to/from the executable model, but is available to the EDA tool for information.

```
(poles (Usage InOut) (Type Float)
      (Table
      (Labels "complex_conj_flag" "real_part" "imag_part")
            (1    -5e8        0)
            (2    -9.4e8      8.3e8)
            (1    -7.3e8      0)
      )
      (Description "Two real and two complex poles")
)
```

The EDA tool sends to the executable model in the parameter string:

```
      (poles  1  -5e8  0  2  -9.4e8  8.3e8  1  -7.3e8  0)
```

An updated set with a different number of pole and row entries can be returned with a similar sequence to be converted back into the same or different number of rows.

Type used to specify the type entry for each column: The example above is modified with Type entries for each column.

```
(poles (Usage InOut) (Type Integer Float Float)
      (Table
      (Labels "complex_conj_flag" "real_part" "imag_part")
            (1    -5e8        0)
            (2    -9.4e8      8.3e8)
            (1    -7.3e8      0)
      )
```

```
      (Description "Two real and two complex poles")
)
```

The encoding in the previous example is sent to the EDA tool and returned to the executable model.

Example of two rows with Type entries for each column: The fourth column numbers are interpreted as UI values.

```
(pdf (Usage In) (Type Integer Integer Float UI Float)
      (Table
      (Labels "Row" "Bin number" "Time" "UI" "Probability")
            (1   -5   -5e-9   -1     1e-5)
            (2   -4   -4e-9   -0.8   1e-4)
      )
      (Description "Probability Distribution Function Table")
)
```

The EDA tool sends to the executable model in the parameter string:

```
      (pdf  1  -5  -5e-9  -1  1e-5  2  -4  -4e-9  -0.8  1e-4  ...)
```

Example above, but with Usage Out: Only one row is necessary in the .ami file.

```
(pdf (Usage Out) (Type Integer Integer Float UI Float)
      (Table
      (Labels "Row" "Bin number" "Time" "UI" "Probability")
            (1   -5   -5e-9   -1     1e-5)
      )
      (Description "Probability Distribution Function Table")
)
```

One row is provided as a template, but, the executable model can return in the parameter string different data and more than one row such as shown.

```
      (pdf  1  -6  -6e-9  -1.2  3e-6  2  -5  -5e-9  -1  9e-6 ...)
```

****

ANALYSIS PATH/DATA THAT LED TO SPECIFICATION:

The existing IBIS specification needs clear definition the Format 'Table' and how it should be treated when used as a format for a Usage In, InOut, and Out parameter for all cases of single-row and multi-row table. Tables have been used in practice for Model_Specific applications to pass single-row information and bit streams, and this support is made legal. The stated examples in Version 5.0 are mostly supported, but the interpretation is clarified.

Several key concepts are introduced.

1. The .ami file rules of for surrounding each row by '(' and ')' describe the number of columns to the EDA tool and ASSUMED beforehand by the

executable model for both Reserved_Parameters and Model_Specific parameters
2. Because of 1) the table with multiple rows can be decoded by the EDA tool from a single row parameter string in the legal format (parameter_name val1 val1 val3 ...) into a multi-row table.
3. Because of 1) an InOut and Out parameter can be returned from the executable model with a different number of rows than entered into the .ami file. In fact, a single row .ami entry can serve as a template for an Out parameter
4. Labels are defined in the .ami file for usage by the EDA tool only. They also may serve as documentation in the .ami file. They are not sent to/from the executable model. The number of labels must equal the number of row columns, and the position of Labels is specified.
5. Type is used to describe all table entries with a single type. The Type rule is extended to support multiple column types. This can be used for current applications and to support and obvious generalization of Table for future use. The column type information in the .ami file is available to the EDA tool and is assumed to be known by the executable model. So a multi-type encoding still follows the BNF syntax (parameter_name val val val ...).

With the new Type extension, a table is sent/received as a (parameter_name val1 val val2 ...) string that can be encoded with val1 and val3 as a number and val2 as a quoted string (or as a Boolean). For multi-row tables, the Type sequence cycle is repeated.

The existing Table example for Rx_Clock_PDF is removed. A reference is made that the rules for entries for reserved parameters is moved to the section where Reserve_Parameters are defined. For example, a reserved parameter might support Table with only three columns and a defined interpretation for each column. The scope of the Table definition is to define the primitive and related syntax and not to get into details about a parameter that has not yet been introduced. The details would have to be introduced later as part of this BIRD or as a separate BIRD.

In the .ami file, a variation of a single-row table without the first row parethesis ‘(‘ and ‘)’ NOT supported. At least one parenthesis pair must surround the first row. (List is a single row set of entries without the parenthesis). There was no need for this exception.

Table entries can be have defined meaning and limitations for Reserved_Parameters. Float can be used as a generic numerical entry along with defined meanings for the column entries. However, just using Type as a flag for just one column is not supported, even if this was implied in Version 5.0. Such usage made Type a flag with its primary definition as a type specification self-contradictory, and this proposal removes that legacy interpretation and provides a better-defined extension.

****

ANY OTHER BACKGROUND INFORMATION:

This proposal is adapted and expanded from earlier proposals and discussions with Ambrish Varma, Walter Katz, and Arpad Muranyi.

Earlier proposals to deal with multi-row tables encoding the first column in a table into a sub-parameter is unnecessary with this BIRD.

Boolean is currently not defined in the IBIS 5.0 BNF, but this BIRD assumes that this data type will be clarified.

The Type definition extension in this BIRD could also be moved to where Type is first introduced.

Some reserved parameter description changes are need. However due to other discussions related to jitter parameters, this action is deferred and may be handled in another BIRD.

****