****
****

BIRD ID#: 137.1
ISSUE TITLE: AMI_parameters_in, AMI_parameters_out, msg Clarifications
REQUESTOR: Arpad Muranyi, Mentor Graphics; Curtis Clark, Ansys
DATE SUBMITTED: August 2, 2011
DATE REVISED:
DATE ACCEPTED BY IBIS OPEN FORUM:

****
****

STATEMENT OF THE ISSUE:

The IBIS 5.0 specification has several problems with the definition of AMI_parameters_out.

On pg. 186, section 3.1.2.6 doesn't mention whether AMI_parameters_in and AMI_parameters_out are required for the AMI_Init function, and what the pointer values should be in the absence of any parameters.

On pg. 189, section 3.2.2.4 says that AMI_parameters_out is optional for AMI_GetWave, but doesn't spell out what the pointer's value should be in case the parameter is not used.

The rule for AMI_parameters_in/out for AMI_Init needs to be defined, and the value of the pointer should also be defined in case the parameters are not used to eliminate any ambiguity in the specification.

****

STATEMENT OF THE RESOLVED SPECIFICATIONS:

On pg. 186, replace these lines:

```
| 3.1.2.6 AMI_parameters (_in and _out)
| ====================================
|
| Memory for AMI_parameters_in is allocated and de-allocated by the EDA
| platform.  The memory pointed to by AMI_parameters_out is allocated and
| de-allocated by the model.  This is a pointer to a string.  All the input
| from the IBIS AMI parameter file are passed using a string that been
| formatted as a parameter tree.
|
| Examples of the tree parameter passing is:
|
|   (dll
|     (tx
|       (taps 4)
|       (spacing sync)
|     )
|   )
|
```

with these lines:

```
|* 3.1.2.6 AMI_parameters_in
|* ========================
```

```
|*
|* The AMI_parameters_in argument is a pointer to a string.  Memory for the
|* string is allocated and de-allocated by the EDA platform.  All the input
|* from the IBIS AMI parameter file are passed to the algorithmic model using
|* a string that has been formatted as a parameter tree.
|*
|** The AMI_parameters_in argument must always be present in the AMI_Init
|** function call and it must always contain the address of a valid string.
|** The sting must always contain at least the root name of the parameter
|** tree, even if there are no parameters to pass to the algorithmic model.
|**
|** Examples of the tree parameter passing:
|**
|**
|** (RootName)
|**
|
|   (dll
|     (tx
|       (taps 4)
|       (spacing sync)
|     )
|   )
|
```

On pg. 186, replace these lines:

```
| The syntax for this string is:
|
| 1. Neither names nor individual values can contain white space characters.
```

with these lines:

```
|** The syntax for the parameter string is:
|
|** 1. The parameter and message strings passed through the AMI_parameters_in,
|**    AMI_parameters_out and msg arguments must not be enclosed in double
|**    quotes.  An empty parameter or message string is valid pointer to a
|**    null byte.
|** 2. Neither names nor individual values can contain white space characters.
```

and bump up each bullet number by one in the remaining part of this section.

On pg. 187, insert before these lines:

```
| 3.1.2.7 AMI_memory_handle
```

the following lines:

```
|* 3.1.2.7 AMI_parameters_out
|* ==========================
|*
|* The AMI_parameters_out argument is a pointer to a string pointer.  Memory
|* for the string is allocated and de-allocated by the algorithmic model.
```

```
|* The model returns a pointer to the string as the contents of this argument.
|* The string must be formatted as a parameter tree, as described in 3.1.2.6.
|* The AMI_Init function may use this string to return parameters to the EDA
|* platform.
|*
|* While the AMI_parameters_out argument must always be present in the
|* AMI_Init function call and the EDA platform must always provide a valid
|* (non-zero) address value in it, algorithmic models are not required to
|* return anything at that address to the EDA platform.  For this reason,
|* the EDA platform must also initialize the memory content at that address
|* to zero (null pointer) prior to calling the AMI_Init function, so that
|* after the execution of the function it can determine whether or not the
|* function returned a valid string pointer at that address.  If the AMI_Init
|* function does not wish to return a parameter string to the EDA platform
|** in this argument, it may take the following actions:
|**    - ignore the address provided in this argument (leaving the EDA tool
|**      provided null pointer at that address)
|**    - return a null pointer at the address provided in this argument
|**      (redundantly rewriting the EDA tool provided null pointer at that
|**      address)
|**    - return a non-zero pointer at the address provided in this argument
|**      to a null (pointer to an empty string)
|**    - return a non-zero pointer at the address provided in this argument
|**      to a string which contains nothing but the root name
|** Note that when parameters are returned to the EDA platform in this
|** argument, the root name must always be included in the string.
|*
```

On pg. 187, replace:

```
| 3.1.2.7 AMI_memory_handle
```

with:

```
| 3.1.2.8 AMI_memory_handle
```

On pg. 187, replace:

```
| 3.1.2.8 msg (optional)
| ======================
|
| Provides descriptive, textual message from the algorithmic model to the EDA
| platform.  It must provide a character string message that can be used by
| EDA platform to update log file or display in user interface.
|
```

with:

```
|* 3.1.2.9 msg
|* ===========
|*
|* The msg argument is a pointer to a string pointer.  Memory for the string
|* is allocated and de-allocated by the algorithmic model.  The model returns
|* a pointer to the string as the contents of this argument.  The AMI_Init
|* function may use this string to send a descriptive, textual message to the
|* EDA platform to be displayed in the user interface and/or to be saved in
|* a log file.
|*
```

```
|* While the msg argument must always be present in the AMI_Init function
|* call and the EDA platform must always provide a valid (non-zero) address
|* value in it, algorithmic models are not required to return anything at that
|* address to the EDA platform.  For this reason, the EDA platform must also
|* initialize the memory content at that address to zero (null pointer) prior
|* to calling the AMI_Init function, so that after the execution of the
|* function it can determine whether or not the function returned a valid
|** string pointer at that address.  If the AMI_Init function does not wish
|** to return a message string to the EDA platform in this argument, it may
|** take the following actions:
|**    - ignore the address provided in this argument (leaving the EDA tool
|**      provided null pointer at that address)
|**    - return a null pointer at the address provided in this argument
|**      (redundantly rewriting the EDA tool provided null pointer at that
|**      address)
|**    - return a non-zero pointer at the address provided in this argument
|**      to a null (pointer to an empty string)
|*
```

On pg. 189, replace these lines:

```
| 3.2.2.4 AMI_parameters_out (optional)
| ====================================
|
| A handle to a 'tree string' as described in 1.3.1.2.6.  This is used by the
| algorithmic model to return dynamic information and parameters.  The memory
| for this string is to be allocated and deleted by the algorithmic model.
|
```

with these lines:

```
|* 3.2.2.4 AMI_parameters_out
|* ==========================
|*
|* The AMI_parameters_out argument is a pointer to a string pointer.  Memory
|* for the string is allocated and de-allocated by the algorithmic model.
|* The model returns a pointer to the string as the contents of this argument.
|* The string must be formatted as a parameter tree, as described in 3.1.2.6.
|* The AMI_GetWave function may use this string to return parameters to the
|* EDA platform.
|*
|* While the AMI_parameters_out argument must always be present in the
|* AMI_GetWave function call and the EDA platform must always provide a valid
|* (non-zero) address value in it, algorithmic models are not required to
|* return anything at that address to the EDA platform.  For this reason,
|* the EDA platform must also initialize the memory content at that address
|* to zero (null pointer) prior to calling the AMI_GetWave function, so that
|* after the execution of the function it can determine whether or not the
|** function returned a valid string pointer at that address.  If the
|** AMI_GetWave function does not wish to return a parameter string to the
|** EDA platform in this argument, it may take the following actions:
|**    - ignore the address provided in this argument (leaving the EDA tool
|**      provided null pointer at that address)
|**    - return a null pointer at the address provided in this argument
|**      (redundantly rewriting the EDA tool provided null pointer at that
|**      address)
|**    - return a non-zero pointer at the address provided in this argument
|**      to a null (pointer to an empty string)
|**    - return a non-zero pointer at the address provided in this argument
```

```
|**      to a string which contains nothing but the root name
|** Note that when parameters are returned to the EDA platform in this
|** argument, the root name must always be included in the string.
|*
```

Questions:
==========

My notes from the last meeting say:

Is the root name always required even if there are no parameters?
In 5.1 say: Never be a null pointer, always point to a string
which contains the root name.

This conflicts the notes I took in the same meeting:

Null pointer,
Pointer to a null,
String with root name only

Which rule do we want?

****

ANALYSIS PATH/DATA THAT LED TO SPECIFICATION:

The changes documented in this BIRD are based on the discussions
which took place in the IBIS ATM teleconference on May 17, 2011 and in
subsequent emails on the ATM email reflector.

****

ANY OTHER BACKGROUND INFORMATION:

****