**BUFFER ISSUE RESOLUTION DOCUMENT (BIRD)**

**BIRD NUMBER:** 215

**ISSUE TITLE:** Back-channel Statistical Optimization Editorial Update

**REQUESTOR:**  Randy Wolff, Micron Technology; Bob Ross, Teraspeed Labs

**DATE SUBMITTED:** September 22, 2021

**DATE REVISED:**

**DATE ACCEPTED:**

**DEFINITION OF THE ISSUE:**

During integration of BIRD201.1 into the IBIS 7.1 draft specification, the IBIS Editorial task group made significant editorial edits to Section 10.9 to expand the training flow descriptions. This BIRD captures those editorial changes. No technical changes were made to the BIRD201.1 content including the AMI\_Impulse function and the BCI\_Training\_Mode parameter.

This BIRD is written to replace and supersede BIRD201.1, originally authored by Walter Katz of SiSoft (MathWorks). The rest of this “Definition of the Issue” section, the “Solution Requirements” section, and the “Summary of Proposed Changes” section below are copied from BIRD201.1 for completeness.

IC vendors would like back-channel to support training in the Statistical flow iterating between the Tx and Rx doing optimization based on impulse responses.

This BIRD defines one new BCI reserved parameter and how it determines what the EDA tool needs to do to support training in the Statistical Flow.

This BIRD also defines one new executable model function AMI\_Impulse. After the initial calls to Tx and Rx AMI\_Init, and if Statistical Training is enabled, the EDA tool shall repeatably call the Tx and Rx AMI\_Impulse functions until the Rx AMI\_Impulse returns BCI\_State Converged.

**SOLUTION REQUIREMENTS:**

The IBIS specification must meet these requirements:

|  |  |
| --- | --- |
| Requirement | Notes |
| 1. Enable back-channel link training messages between the Tx and Rx executable models to enable the Rx or Tx executable model to control the equalization of the Tx or Rx during statistical simulations. |  |
| 1. Allow the user and tool to know when statistical link training has ended, and normal operation has begun. |  |

**SUMMARY OF PROPOSED CHANGES:**

For review purposes, the proposed changes are summarized as follows:

|  |  |  |
| --- | --- | --- |
| Specification Item | New/Modified/Other | Notes |
| New AMI Reserved Parameter  BCI\_Training\_Mode | All are new AMI Parameters | All affect the operation of the AMI function AMI\_Impulse. |
| New executable function AMI\_Impulse | New |  |

**PROPOSED CHANGES:**

ADD TO SECTION 10.2.3 AFTER *Function* AMI\_Init and above *Function* AMI\_GetWave:

*Function:* **AMI\_Impulse**

*Required:* No, and illegal before AMI\_Version 7.1

*Declaration:* long AMI\_Impulse (double \*impulse\_matrix,

char \*BCI\_parameters\_in,

char \*\*BCI\_parameters\_out,

char \*\*AMI\_parameters\_out,

void \*AMI\_memory)

*Arguments:*

impulse\_matrix

Same as impulse\_matrix argument defined under AMI\_Init above.

Note that since both AMI\_Init and AMI\_Impulse modify the impulse\_matrix in place, the EDA tool could maintain the original impulse\_matrix and use different memory for the impulse\_matrix input to the AMI\_Init and AMI\_Impulse functions.

Note that the AMI\_Impulse function uses the number\_of\_rows, aggressors, sample\_interval, and bit\_time that were passed to the AMI\_Init call.

BCI\_parameters\_in

The BCI\_parameters\_in argument is a pointer to a string. This pointer is returned in the BCI\_parameter\_out argument by a previous call to an AMI\_Impulse function in another executable model in the channel. Memory for the string is allocated and de-allocated by the previous call to an AMI\_Impulse function in another executable model in the channel. The string must be formatted as defined by the BCI\_Protocol. On the first call to the primary Tx AMI\_Impulse function, this pointer shall be the Null pointer (0).

BCI\_parameters\_out

The BCI\_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 defined by the BCI\_Protocol.

The EDA tool must initialize the memory content at this address to zero (null pointer) prior to calling the AMI\_Impulse 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.

AMI\_parameters\_out

Same as AMI\_Parameters\_out argument defined under AMI\_Init above.

**AMI\_memory**

This is the memory pointer which was allocated during the AMI\_Init call.

Return Value

1 for success

0 for failure

Algorithmic models shall return a failure code (0) if and only if the function call fails due to a program execution error. In all other cases the return code shall be "success" (1), even if the function cannot operate properly due to some functional problems. For example, if a function includes a CDR which is unable to get into a stable mode, the function shall still return a success code (1). Examples for returning a failure code (0) may include an invalid data type, a null pointer during run time, or anything that prevents the successful execution of the model’s code.

The authors of Algorithmic Models are encouraged to provide feedback to the EDA tool’s users through the various available messaging options about any difficulties the model encounters during execution, regardless of what the value of the function’s return code is.

ADD SECTION SUBTITLES FOR REPEATERS

Page 262 ABOVE “The time-domain simulation flow for a Repeater link shown in Figure 41 is defined below.”

**10.8.2 REPEATER TIME DOMAIN SIMULATION FLOW FOR A REPEATER LINK**

Page 264, ABOVE “The statistical simulation flow for a Repeater link shown in Figure 41 is defined below.”

**10.8.3 REPEATER STATISTICAL DOMAIN SIMULATION FLOW FOR A REPEATER LINK**

REPLACE:

## AMI Reserved Parameter Definitions For Link Training Communications

In this section, the parameters BCI\_Protocol, BCI\_State, BCI\_ID, BCI\_Message\_Interval\_UI and BCI\_Training\_UI are documented, to enable link training communication. These Reserved Parameters are in the AMI file and positioned under the Reserved\_Parameters branch.

WITH:

## AMI Reserved Parameter Definitions For Link Training Communications

With the information provided in this section, IC vendors and EDA tool vendors should be able to develop models that support Back Channel Training and develop enhancements EDA tools will need to support these models. The following Reserved Parameters are in the AMI file and positioned under the Reserved\_Parameters branch.

ADD AFTER *Parameter* BCI\_Training\_UI SECTION AND ABOVE 10.9.1:

*Parameter:* **BCI\_Training\_Mode**

*Required:* No, and illegal before AMI\_Version 7.1

*Direction:* Rx, Tx

*Descriptors*:

Usage: In

Type: String

Format: Value, List

Default: <string literal>

Description:<string>

*Definition:* This parameter tells the EDA tool whether the model supports statistical (AMI\_Init-based, using the “Impulse” argument) link training only, time-domain (AMI\_GetWave-based, using the “GetWave” argument) link training only, or “Both” (both statistical link training followed by time-domain link training). The only allowed values of BCI\_Training\_Mode are “Impulse”, “GetWave”, or “Both”. If “Both” is present, then “Impulse” and “GetWave” shall also be present.

Allowed Formats:

(Value “Impulse”)

(Value “GetWave”)

(List “Impulse”)

(List “GetWave”)

(List “Impulse” “GetWave”)

(List “Impulse” “GetWave” “Both”)

Illegal Formats:

(Value “Both”)

(List “Both” “Impulse”)

(List “Both” “GetWave”)

*Usage Rules:* The user or EDA tool can only choose a BCI\_Training\_Mode value if it is available on both the Tx and the Rx (and must be set the same for Tx and Rx). If BCI\_Protocol is present but BCI\_Training\_Mode is not present, then the EDA tool shall use a BCI training mode of “GetWave”.

To run a BCI statistical training simulation, the Tx and Rx model (or the Terminal Tx, Terminal Rx, and all Repeater Rx and Tx models) must have BCI\_Training\_Mode as either “Impulse” or “Both”.

To run a BCI time-domain training simulation, the Tx and Rx model (or the Terminal Tx, Terminal Rx, and all Repeater Rx and Tx models) must have BCI\_Training\_Mode as either “GetWave” or “Both”.

Training and channel simulation need not use the same mode (e.g., training may be performed statistically, but channel simulation using those training results may be performed in the time domain).

*Example:*

(BCI\_Training\_Mode (Usage In)(Type String)(List “Both” “Impulse” “GetWave”)

(Description "This Device supports back-channel statistical and time-domain optimization”))

REPLACE SECTIONS 10.9.1 AND 10.9.2 WITH:

### 10.9.1 Training Flows

**A Unified Description of all AMI Training and Simulation Flows**

An IBIS-AMI channel consists of:

* A Terminal Tx
* Zero or more Repeaters with each Repeater consisting of:
  + Repeater Rx
  + Repeater Tx
* A Terminal Rx

This forms a daisy chain list of Tx and Rx models.

Every simulation will sequentially execute the AMI\_Init function of each model in this daisy chain list.

* The input to each Tx AMI\_Init function shall be in accordance with the value of its Tx\_Impulse\_Input
* The input to each Rx AMI\_Init function shall be the cumulative upstream impulse response

If BCI\_Training\_Mode is set to “Impulse” or “Both” then the simulator will sequentially execute the AMI\_Impulse function of each model in this daisy chain list repeatedly.

* The input to each Tx AMI\_Impulse function shall be in accordance with the value of its Tx\_Impulse\_Input and whether the Tx is a Terminal Tx or a Repeater Tx
* The input to each Rx AMI\_Impulse function shall be the cumulative upstream impulse response
* The BCI training terminates when BCI\_State returns “Converged”, “Fail”, or “Error” in the AMI\_parameters\_out of the Terminal Rx AMI\_Impulse function

If doing time-domain simulation, the simulator will sequentially execute the AMI\_GetWave function of each model in this daisy chain list repeatedly. If BCI\_Training\_Mode is set to “Both” or “GetWave” then the training will continue until the conditions described below have been reached.

* If BCI\_Training\_Mode is set to “Both” or “GetWave” then the training will finish when BCI\_State returns “Converged”, “Fail”, or “Error” in the AMI\_parameters\_out parameter string of the Terminal Rx AMI\_GetWave function. Then the EDA tool can start to accumulate waveform statistics.
* If BCI\_Training\_Mode is not set to “Both” then the EDA tool can start to accumulate waveform statistics after the EDA tool has completed simulation of a number of UI equal to Ignore\_Bits.

**Statistical Only Training Flow For Channels With No RepeaterS (BCI\_TRAINING\_MODE SET TO “IMPULSE”)**

The EDA tool shall make calls as described below to the Tx AMI\_Init and Rx AMI\_Init functions and Tx AMI\_Impulse and Rx AMI\_Impulse functions.

1. Tx AMI\_Init is called by the EDA tool using the following settings as part of the AMI parameter string:

(BCI\_State “Training”) (BCI\_Protocol “<name>”) (BCI\_ID “<my\_ ID>”)

(BCI\_Training\_Mode “Impulse”)

If BCI\_Training\_Mode is “GetWave”, follow the flow in Section 10.9.1. The impulse\_matrix argument contains the impulse response of the channel. If the Tx executable model does not implement the BCI\_Protocol and BCI\_Training\_Mode, it returns “Error” in BCI\_State.

1. Rx AMI\_Init is called by the EDA tool using the following settings as part of the AMI parameter string:

(BCI\_State “Training”) (BCI\_Protocol “<name>”) (BCI\_ID “<my\_ID>”) (BCI\_Training\_Mode “Impulse”)

The impulse\_matrix argument contains the impulse response output of Tx AMI\_Init. If the Rx executable model does not implement the BCI\_Protocol and BCI\_Training\_Mode, it returns “Error” in BCI\_State. The EDA tool may analyze the results of Rx AMI\_Init.

1. Tx AMI\_Impulse is called with the same impulse\_matrix used in the call to Tx AMI\_Init. The content of BCI\_parameters\_in shall be the empty string (“”) on the first call to Tx AMI\_Impulse. On subsequent calls to Tx AMI\_Impulse, the value of BCI\_parameters\_in shall be the value returned in BCI\_parameters\_out by the previous call to Rx AMI\_Impulse.
2. Rx AMI\_Impulse is called using the impulse\_matrix output of Tx AMI\_Impulse. The value of BCI\_parameters\_in shall be set to the value of BCI\_parameters\_out of the previous call to Tx AMI\_Impulse.
3. Steps 3 and 4 are repeated until the Rx AMI\_Impulse returns AMI Reserved Parameter BCI\_State “Converged”, “Failed”, or “Error”
   1. “Converged”: The impulse\_matrix and AMI\_parameters\_out returned by the receiver AMI\_Impulse function are used by the EDA tool to complete the simulation.
   2. “Fail”: This tells the EDA tool that the training failed to converge. The EDA tool can terminate the simulation or proceed using the outputs of AMI\_Impulse or AMI\_Init to complete the simulation at its own risk.
   3. “Error”: Tells the EDA tool that an error was detected that prevented optimization to continue. The EDA tool can terminate the simulation or proceed using the outputs of AMI\_Impulse or AMI\_Init to complete the simulation at its own risk.
4. Even if the user selected time-domain simulation after statistical (AMI\_Impulse) training, since BCI\_Training\_Mode is “Impulse”, the Tx and Rx AMI\_GetWave functions shall not perform training and shall return BCI\_State “Off”.
5. Tx AMI\_Close and Rx AMI\_Close are called.

**Statistical Only Training Flow For Channels With Repeaters (BCI\_Training\_Mode Set to “Impulse”)**

1. The AMI\_Init flow is identical to the flow defined for the statistical simulation flow for a link containing a Repeater as shown in Figure 44 in Section 10.8.2 and documented in Section 10.8.3.
2. This same flow is repeated with the calls to AMI\_Init replaced by calls to AMI\_Impulse. Note that the EDA tool shall set the value of BCI\_parameters\_in to the value of BCI\_parameters\_out of the previous call to an AMI\_Impulse function in the channel.
3. The BCI training terminates when BCI\_State returns “Converged”, “Fail”, or “Error” in the AMI\_parameters\_out of the Terminal Rx AMI\_Impulse function.

**Time-Domain Only Training Flow for Channels with No RepeaterS (BCI\_TRAINING\_MODE SET TO “GETWAVE”)**

The EDA tool shall make calls as described below to the Tx AMI\_Init and Rx AMI\_Init functions and Tx AMI\_GetWave and Rx AMI\_GetWave functions.

1. Tx AMI\_Init is called by the EDA tool using the following settings in the AMI parameter string:

(BCI\_State “Training”) (BCI\_Protocol “<name>“) (BCI\_ID “<my\_ ID>“)

If the Tx executable model does not implement the BCI\_Protocol, it returns “Error” in BCI\_State. The Tx executable model may write a message file in the BCI\_ID namespace under BCI\_Protocol.

1. Rx AMI\_Init is called by the EDA tool using the following settings in the AMI parameter string:

(BCI\_State “Training”) (BCI\_Protocol “<name>“) (BCI\_ID “<my\_ID>“) (BCI\_Training\_UI <# Training Bits>)

If the Rx executable model does not implement BCI\_Protocol, it returns “Error” in BCI\_State. The Rx executable model may read, write, modify and/or delete message files in the BCI\_ID namespace under BCI\_Protocol.

1. Tx AMI\_GetWave is called by the EDA tool with the stimulus pattern passed in. The Tx executable model may read, write, modify and/or delete message files in the BCI\_ID namespace under BCI\_Protocol.
2. Rx AMI\_GetWave is called by the EDA tool with a waveform that is the convolution of the output of Tx AMI\_GetWave and the impulse response of the channel. The Rx executable model may read, write, modify and/or delete message files under BCI\_Protocol.
3. Steps 3 and 4 are repeated until the EDA tool stops the simulation. The EDA tool should start processing the output of Rx AMI\_GetWave after Ignore\_Bits and either:

after BCI\_Training\_UI, or

when the Rx AMI\_GetWave function returns BCI\_State “Converged” or “Failed” or either the Tx or Rx executable model returns “Error”.

1. Tx and Rx AMI\_Close are called.

Note that the EDA tool does not need to perform any operations specifically assisting the BCI communication between the Tx and the Rx executable models beyond passing the BCI parameters to both executable models on AMI\_Init.

**Time-Domain Only Training Flow for Channels with RepeaterS (BCI\_TRAINING\_MODE SET TO “GETWAVE”)**

The EDA tool shall make the following calls to the Upstream Tx, Repeater Rx, Repeater Tx, Downstream Rx AMI\_Init and AMI\_GetWave functions.

1. Upstream Tx AMI\_Init is called by the EDA tool using the following settings as part of the AMI parameter string:

(BCI\_State “Training”) (BCI\_Protocol “<name>”) (BCI\_ “<my\_ID>”)

If the executable model does not implement the BCI\_Protocol, it returns “Error” in BCI\_State. The executable model may write a message file in the BCI\_ID namespace under BCI\_Protocol.

1. Repeater Rx AMI\_Init is called by the EDA tool using the following settings as part of the AMI parameter string:

(BCI\_State “Training”) (BCI\_Protocol “<name>”) (BCI\_ID “<my\_ID>”)

If the executable model does not implement the BCI\_Protocol, it returns “Error” in BCI\_State. The executable model may read, write, modify and/or delete message files in the BCI\_ID namespace under BCI\_Protocol.

1. Repeater Tx AMI\_Init is called by the EDA tool using the following settings as part of the AMI parameter string:

(BCI\_State “Training”) (BCI\_Protocol “<name>”) (BCI\_ID “<my\_ID>”)

If the executable model does not implement the BCI\_Protocol, it returns “Error” in BCI\_State. The executable model may read, write, modify and/or delete message files in the BCI\_ID namespace under BCI\_Protocol.

1. Downstream Rx AMI\_Init is called by the EDA tool using the following settings as part of the AMI parameter string:

(BCI\_State “Training”) (BCI\_Protocol “<name>”) (BCI\_ID “<my\_ID>”) (BCI\_Training\_UI <# Training Bits>)

If the executable model does not implement the BCI\_Protocol, it returns “Error” in BCI\_State. The executable model may read, write, modify and/or delete message files in the BCI\_ID namespace under BCI\_Protocol.

1. Upstream Tx AMI\_GetWave is called with the stimulus pattern. The executable model may read, write, modify and/or delete message files in the BCI\_ID namespace under BCI\_Protocol.
2. Repeater Rx AMI\_GetWave is called with the Upstream Tx AMI\_GetWave function’s output waveform combined with the Upstream Channel Impulse Response. The executable model may read, write, modify and/or delete message files in the BCI\_ID namespace under BCI\_Protocol.
3. Repeater Tx AMI\_GetWave is called with the waveform output of the Repeater Rx AMI\_GetWave. The executable model may read, write, modify and/or delete message files in the BCI\_ID namespace under BCI\_Protocol.
4. Downstream Rx AMI\_GetWave is called with the Repeater Tx AMI\_GetWave function’s output waveform combined with the Downstream Channel Impulse Response. The executable model may read, write, modify and/or delete message files in the BCI\_ID namespace under BCI\_Protocol
5. Steps 5 through 8 are repeated until the EDA tool stops the simulation.
   1. The EDA tool should start processing the output of Rx AMI\_GetWave after Ignore\_Bits and either:

after BCI\_Training\_UI, or

when the downstream Rx AMI\_GetWave function returns BCI\_State “Converged” or “Failed” or any executable model in the channel returns “Error”.

Note that it is the responsibility of the BCI \_Protocol to define the BCI message files and contents therein so that each executable model in the channel can determine its role/position in the channel optimization.

**Combined statistical and Time-Domain Training Flow for Channels with no Repeater (BCI\_TRAINING\_MODE SET TO “both”)**

The EDA tool shall make calls as described below to the Tx AMI\_Init and Rx AMI\_Init functions and Tx AMI\_Impulse and Rx AMI\_Impulse functions.

1. Tx AMI\_Init is called by the EDA tool using the following settings as part of the AMI parameter string:

(BCI\_State “Training”) (BCI\_Protocol “<name>”) (BCI\_ID “<my\_ ID>”) (BCI\_Training\_Mode “Both” or “Impulse”)

If (BCI\_Training\_Mode “GetWave”) follow the flow in Section 10.9.1. The impulse\_matrix argument contains the impulse response of the channel. If the Tx executable model does not implement the BCI\_Protocol and BCI\_Training\_Mode, it returns “Error” in BCI\_State.

1. Rx AMI\_Init is called by the EDA tool using the following settings as part of the AMI parameter string:

(BCI\_State “Training”) (BCI\_Protocol “<name>”) (BCI\_ID “<my\_ID>”) (BCI\_Training\_Mode “Both” or “Impulse”)

The impulse\_matrix argument contains the impulse response output of Tx AMI\_Init. If the Rx executable model does not implement the BCI\_Protocol and BCI\_Training\_Mode, it returns “Error” in BCI\_State. The EDA tool may analyze the results of Rx AMI\_Init.

1. Tx AMI\_Impulse is called with the same impulse\_matrix used in the call to Tx AMI\_Init. The value of BCI\_parameters\_in shall be Null (0) on the first call to Tx AMI\_Impulse and the value of BCI\_parameters\_out of the previous call to Rx AMI\_Impulse on subsequent calls to Tx AMI\_Impulse.
2. Rx AMI\_Impulse is called using the impulse\_matrix output of Tx AMI\_Impulse. The value of BCI\_parameters\_in shall be set to the value of BCI\_parameters\_out of the previous call to Tx AMI\_Impulse.
3. Steps 3 and 4 are repeated until the Rx AMI\_Impulse returns AMI Reserved Parameter BCI\_State “Converged”, “Failed”, or “Error”
   1. “Converged”: The impulse\_matrix and AMI\_parameters\_out returned by the receiver AMI\_Impulse function are used by EDA tool to complete the simulation.
   2. “Fail”: This tells the EDA tool that the training failed to converge. The EDA tool can terminate the simulation or proceed using the outputs of AMI\_Impulse or AMI\_Init to complete the simulation at its own risk.
   3. “Error”: Tells the EDA tool that an error was detected that prevented optimization to continue. The EDA tool can terminate the simulation or proceed using the outputs of AMI\_Impulse or AMI\_Init to complete the simulation at its own risk.
4. As BCI\_Training\_Mode is “Both”, the time-domain training as described above in this section is invoked.
5. Tx AMI\_Close and Rx AMI\_Close are called.

**Combined Statistical and Time-Domain Training Flow For Channels With RepeaterS (BCI\_TRAINING\_MODE SET TO “BOTH”)**

The EDA tool shall make calls as described below to the Tx AMI\_Init and Rx AMI\_Init functions and Tx AMI\_Impulse and Rx AMI\_Impulse functions and the Tx AMI\_GetWave and Rx AMI\_GetWave functions

1. The AMI\_Init flow is identical to the flow defined for the statistical simulation flow for a link containing a Repeater as shown in Figure 44 in Section 10.8.2 and documented in Section 10.8.3.
2. This same flow is repeated with the calls to AMI\_Init replaced by calls to AMI\_Impulse.
3. The BCI training terminates when BCI\_State returns “Converged”, “Fail”, or “Error” in the AMI\_parameters\_out of the Terminal Rx AMI\_Impulse function.
4. Upstream Tx AMI\_GetWave is called with the stimulus pattern. The executable model may read, write, modify and/or delete message files in the BCI\_ID namespace under BCI\_Protocol.
5. Repeater Rx AMI\_GetWave is called with the Upstream Tx AMI\_GetWave fucntion’s output waveform combined with the Upstream Channel Impulse Response. The executable model may read, write, modify and/or delete message files in the BCI\_ID namespace under BCI\_Protocol.
6. Repeater Tx AMI\_GetWave is called with the waveform output of the Repeater Rx AMI\_GetWave. The executable model may read, write, modify and/or delete message files in the BCI\_ID namespace under BCI\_Protocol.
7. Downstream Rx AMI\_GetWave is called with the Repeater Tx AMI\_GetWave function’s output waveform combined with the Downstream Channel Impulse Response. The executable model may read, write, modify and/or delete message files in the BCI\_ID namespace under BCI\_Protocol
8. Steps 5 through 8 are repeated until the EDA tool stops the simulation.
   1. The EDA tool should start processing the output of Rx AMI\_GetWave after Ignore\_Bits and either:

after BCI\_Training\_UI, or

when the downstream Rx AMI\_GetWave function returns BCI\_State “Converged” or “Failed” or any executable model in the channel returns “Error”.

Note that it is the responsibility of the BCI \_Protocol to define the BCI message files and contents therein so that each executable model in the channel can determine its role/position in the channel optimization.

CHANGE SECTION NUMBER FROM SECTION 10.9.3 TO 10.9.2 AND ADD BCI\_Training\_Mode to TABLES 33-35:

### Summary Tables for Usage, Type and Format

Table 33 – General Rules and Allowable Usage for BCI Reserved Parameters

| **Reserved Parameter** | **General Rules** | | **Allowable Usage** | | | | |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Required** | **Default2,4** | **Info** | **In** | **Out** | **Dep1** | **InOut** |
| BCI\_Message\_Interval\_UI | No3 | -- | X |  |  |  |  |
| BCI\_ID | No3 | -- |  | X |  |  |  |
| BCI\_Protocol | No | -- |  | X |  |  |  |
| BCI\_State | No3 | -- |  |  |  |  | X |
| BCI\_Training\_UI | No3 | -- |  | X |  |  |  |
| BCI\_Training\_Mode | No5 | “GetWave” |  | X |  |  |  |
| Notes:   1. Illegal for AMI\_Version 6.0 and earlier 2. “Default” in this context means “behavior if Reserved Parameter is absent” 3. Required if BCI\_Protocol is present 4. “--" means that an entry must be provided if the parameter is present; no default is assumed or permitted 5. Illegal for AMI\_Version 7.0 and earlier, required if BCI\_Protocol supports statistical training | | | | | | | |

**Table 34 – Allowable Data Types for BCI Reserved Parameters**

| **Reserved Parameter** | **Data Type** | | | | |
| --- | --- | --- | --- | --- | --- |
| **Float** | **UI** | **Integer** | **String** | **Boolean** |
| BCI\_Message\_Interval\_UI |  |  | X |  |  |
| BCI\_ID |  |  |  | X |  |
| BCI\_Protocol |  |  |  | X |  |
| BCI\_State |  |  |  | X |  |
| BCI\_Training\_UI |  |  | X |  |  |
| BCI\_Training\_Mode |  |  |  | X |  |

Table 35 – Allowable Data Formats for BCI Reserved Parameters

| **Reserved Parameter** | **Data Format** | | | | | | | | | |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Value** | **Range** | **Corner** | **List** | **Increment** | **Steps** | **Gaussian** | **Both-Dirac** | **DjRj** | **Table** |
| BCI\_Message\_Interval\_UI | X |  |  |  |  |  |  |  |  |  |
| BCI\_ID | X |  |  |  |  |  |  |  |  |  |
| BCI\_Protocol | X |  |  | X |  |  |  |  |  |  |
| BCI\_State |  |  |  | X |  |  |  |  |  |  |
| BCI\_Training\_UI | X |  |  |  |  |  |  |  |  |  |
| BCI\_Training\_Mode | X |  |  | X |  |  |  |  |  |  |

ADD TO TABLE 39 ABOVE BCI\_Training\_UI IN SECTION 10.12 FOR FIRST SUPPORTED AMI VERSION:

BCI\_Training\_Mode 7.1

**BACKGROUND INFORMATION/HISTORY:**

BIRD215 captures many editorial and organizational changes (including new subsections under 10.9) that the Editorial Task Group made while adding BIRD201.1 to the IBIS 7.1 specification draft.