API Documentation

rev 5.0

November 6th, 2019

Table of content

mlBert_createInstance

This API call is responsible to create instances.

This API is a prerequisite to all other API calls.

mlBert_Connect

This API call is responsible to establish a connection between the client application and the BERT. The API will return 1 upon successful connection or 0 if a problem occurs while connecting.

This API can be used to connect simultaneously with many BERTs using the instance parameter, each connected BERT will have a unique instance.

This API is a prerequisite to all other API calls.

Used for all BERTs.

Example of use:

```
/*
 * mlBert_Connect EntryPoint inside the DLL is responsible to connect to the BERT
*/

[DllImport(@"DLL/MLBert_API.dll", EntryPoint = " mlBert_Connect")]
 private static extern int mlBert_Connect (mlbertapi* instance, char* IP);

/*
* Param mlbertapi* instance
* Param String ipAddress
* return int
* Public Interface to connect with DLL, use IP address of the instrument in order to open an Ethernet
Connection between the client host (running the DLL) and this instrument
* Instance parameter allows to connect Multiple BERTs, every BERT will have a unique instance
*/

public int Initialize(mlbertapi* instance, char* IP)
{
  return mlBert_Connect (instance, IP);
 }
```

mlBert_ReadBoardID

This API call is used to read the ID stored on the board. Each board will have its own unique ID number programmed by Multilane.

API returns ID as 32 bit integer.

Used for all BERTs.

Example of use:

```
/*
* mlBert_ReadBoardID EntryPoint inside the DLL allows to get the Board ID number stored on the
hardware

*/

 [DllImport(@"DLL/MLBert_API.dll", EntryPoint = "mlBert_ReadBoardID")]
private static extern int mlBert_ReadBoardID (mlbertapi* instance);



/*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Returns 32 bit integer
*/

public int ReadBoardID (mlbertapi* instance)
 {
return mlBert_ReadBoardID (instance);

}
```

mlBert_ConfigureApplication

This API call is used to configure the application by setting clock location files, bathtub curve and eye report saves location.

API returns 1 upon success, 0 if failure.

Used for all BERTs.

Example of use:

```
/*
 * mlBert_ConfigureApplication EntryPoint inside the DLL is responsible to set the Line Rate
*/

[DllImport(@"DLL/MLBert_API.dll", EntryPoint = " mlBert_ConfigureApplication ")]
private static extern int mlBert_ConfigureApplication (mlbertapi* instance, string saveConfig, string
saveBathtub, string saveEye, int saveBathtubEnable, int saveEyeEnable);

/*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Param string saveConfig (Location of the Clock File required for the APILineRateConfiguration)
* Param string saveBathtub (Save location of the BathTube Curve files)
* Param string saveEye (Save location of the Eye files)
* Param int saveBathtubEnable (Enable BathTub save, Enable = 1 or Disable = 0)
* Param int saveEyeEnable (Enable Eye save, Enable = 1 or Disable = 0)
* return int
* Allows to configure the application
* Returns 1 on success, 0 on failure
* Boards connected are identified using the instance parameter
*/

public int ConfigureApplication(string saveConfig, string saveBathtub, string saveEye, int
saveBathtubEnable, int saveEyeEnable)
 {
return mlBert_ConfigureApplication (instance, saveConfig, saveBathtub, saveEye, saveBathtubEnable,
saveEyeEnable);
 }
```

mlBert_LineRateConfiguration

This API call is used to set the Line Rate of the connected BERT. It returns 1 if successful, 0 if a problem occurs while setting the line Rate.

For every Line Rate a File should be provided or generated that will configure the Silab, location of the file will be set by using “mlBert_ConfigureApplication”.

To generate the clock file the user should use a special GUI provided by MultiLane, or a C# API (ClockLibrary.dll) that generate clock files also provided by multilane.

Used for Berts except Pam boards

Example of use:

```
/*
 * mlBert_LineRateConfiguration EntryPoint inside the DLL is responsible to set the Line Rate
*/
[DllImport(@"DLL/MLBert_API.dll", EntryPoint = " mlBert_LineRateConfiguration ")]
private static extern int mlBert_LineRateConfiguration (mlbertapi* instance, double lineRate, int
clockSource);

/*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Param double lineRate
* Param int clockSource (1 = internal, 0 = external)
* return int
* Allows to set Line Rate generated by the connected board
* DLL use lineRate value to set the Line Rate of the connected board
* Boards connected are identified using the instance parameter
*/

public int LineRateConfiguration(mlbertapi* instance, double lineRate, int clockSource)
 {
return mlBert_LineRateConfiguration (instance, lineRate, clockSource);
 }
```

mlBert_LineRateConfiguration_04

This API call is used to set the Line Rate of the connected BERT Pam4. It returns 1 if successful, 0 if a problem occurs while setting the line Rate.

For every Line Rate a File should be provided or generated that will configure the Silab, location of the file will be set by using “mlBert_ConfigureApplication”.

To generate the clock file the user should use a special GUI provided by MultiLane, or a C# API (ClockLibrary.dll) that generate clock files also provided by multilane.

Used for Pam boards only

Example of use:

```
/*
 * mlBert_LineRateConfiguration_04 EntryPoint inside the DLL is responsible to set the Line Rate
*/
[DllImport(@"DLL/MLBert_API.dll", EntryPoint = " mlBert_LineRateConfiguration_04")]

private static extern int mlBert_LineRateConfiguration_04 (mlbertapi* instance, double lineRate, int
EyeMode, int GrayMapping, int PRECoding, int BER_FECMode, int clockSource, int FECOperationMode,
int HostSide, int HostLineDebug);
/*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Param double lineRate
* Param int EyeMode (0= Pam4, 1= NRZ)
* Param GrayMapping (0 = disabled, 1 = enabled)
* Param PRECoding (0 = disabled, 1 = enabled)
* Param BER_FECMode (0 = disabled, 1 = enabled)
* Param int clockSource (1 = internal, 0 = external)
* Param int FECOperationMode: 0 RP4, 1 KP4
* Param int HostSide: 1 for HostSide, 0 for LineSide
* Param int HostLineDebug: value = 0
* return int
* Allows to set Line Rate generated by the connected board
* DLL use lineRate value to set the Line Rate of the connected board
* Boards connected are identified using the instance parameter
*/
public int LineRateConfiguration_04(mlbertapi* instance, double lineRate, int EyeMode, int
GrayMapping, int PRECoding, int BER_FECMode, int clockSource, int FECOperationMode, int HostSide,
int HostLineDebug)
 {
    return mlBert_LineRateConfiguration_04 (instance, lineRate, EyeMode, FECMode, GrayMapping,
PRECoding, BER_FECMode, clockSource, FECOperationMode, HostSide, HostLineDebug);
 }
```

mlBert_SetPRBSPattern

This API call is used to set the PRBS Pattern on TX and RX, in addition to setting the TX and RX Invert status. If TX inverted is TRUE the generated pattern will have an inverted polarity, If RX inverted is TRUE the incoming pattern polarity will be inverted. Each channel should have a separate call. API returns 1 upon success, 0 if an error occurred.

Used for all BERTs.

Example of Use:

```
/*
* mlBert_SetPRBSPattern EntryPoint inside the DLL responsible to Set the PRBS Pattern
*/

[DllImport(@"DLL/MLBert_API.dll", EntryPoint = " mlBert_SetPRBSPattern")]
private static extern bool mlBert_SetPRBSPattern (mlbertapi* instance, int channel, int txPattern, int
rxPattern, int txInvert, int rxInvert);

/*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Param int channel
* Param int txPattern
* Param int
* Param int txInvert (Normal Mode = 0, Inverted Mode = 1)
 *Param int rxInvert (Normal Mode = 0, Inverted Mode = 1)
* return int
* Allows to set the PRBS Pattern generated by each channel
* If RX/TX inverted, the polarity of sequence received/generated will be inverted
* DLL use pattern enumeration send in order to generate PRBS
* Channels are differentiated using the channel parameter
* Boards connected are identified using the instance parameter
*/


public int SetPRBSPattern(mlbertapi* instance, int channel, int txPattern, int rxPattern, int txInvert, int
rxInvert)
 {
  return mlBert_SetPRBSPattern (instance, channel, txPattern, rxPattern, txInvert, rxInvert);
 }
```

TX patterns index:

4039	4039D/4079D	4039E/4039EN	4039B
PRBS 7 = 0	PRBS 7 = 0	PRBS 7 = 0	PRBS 7 = 0
PRBS 9 = 1	PRBS 9 = 1	PRBS 9 = 1	PRBS 9 = 1
PRBS 15 = 2	PRBS 11 = 2	PRBS 11 = 2	PRBS 11 = 2
PRBS 23 = 3	PRBS 13 = 3	PRBS 15 = 3	PRBS 15 = 3
PRBS 31 = 4	PRBS 15 = 4	PRBS 23 = 4	PRBS 23 = 4
User defined = 5	PRBS 16 = 5	PRBS 31 = 5	PRBS 31 = 5
8:8 pattern= 6	PRBS 23 = 6	SQ16 = 6	PRBS 13 = 6
	PRBS 31 = 7	SQ32 = 7	PRBS 9_4 = 10
	JP03B = 10	User defined = 8	PRBS 58 = 11
	LIN = 11	PRBS 13 = 9	JP03B = 12
	CJT = 12	PRBS 9_4 = 10	IEEE 802.3bs = 13
	SSPRQ = 13	PRBS 58 = 11	OIF-CEI-3.1 = 14
	User defined = 14	JP03B = 12	User defined = 15
		LIN = 13	
		CJT = 14	
		SSPRQ = 15	

mlBert_PRBSVerfication

This API call is used to verify the PRBS pattern received by the RX channel match the pattern set using APISetPRBSPattern. It does not work on userDefined Patterns.

Each channel should have a separate call.

Returns 1 if the pattern received on RX channel match the pattern set using mlBert_SetPRBSPattern.

Used for all BERTs.

Example of use:

```
/*
*  mlBert_PRBSVerfication EntryPoint inside the DLL responsible to verify the PRBS pattern
* received by the RX channel.
*  Does not work on userDefined Pattern
*/

 [DllImport(@"DLL/MLBert_API.dll", EntryPoint = " mlBert_PRBSVerfication ")]
 static extern int mlBert_PRBSVerfication (mlbertapi* instance, int channel);



/*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Param int channel
* return int
* Allows to verify the PRBS pattern received by the RX channel.
* Return 1 if the pattern received on the RX channel match the pattern set using APISetPRBSPattern.
* Does not work on userDefined Pattern
* Channels are identified using the channel parameter
* Boards connected are identified using the instance parameter
*/

public int PRBSVerfication(mlbertapi* instance, int channel)
 {
   return mlBert_PRBSVerfication (instance, channel);
 }
```

mlBert_SetTxUserPattern

This API call is used to set a user defined PRBS to be generated by the selected channel.
APISetPRBSPattern should be set to “user Defined ” in order for this API to work.
Each channel should have a separate call.

API returns 1 upon success, 0 if failure.

Example of Use:

```
/*
* mlBert_SetTxUserPattern EntryPoint inside the DLL allows to set user Defined Pattern to be generated
*/

[DllImport(@"DLL/MLBert_API.dll", EntryPoint = " mlBert_SetTxUserPattern ")]
private static extern int mlBert_SetTxUserPattern (mlbertapi* instance, int channel, ulong
UserDefinedPattern);

/*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Param int channel
* Param ulong Pattern (ex: 0x000000f0f0f0f0f0) hexadecimal
* return int
* Allows setting a user defined PRBS in order to be generated by the channel.
* APISetPRBSPattern should be set in order for this API to work
* Returns 1 on success, 0 on failure.
* Channels are identified using the channel parameter
* Boards connected are identified using the instance parameter
*/

public int SetTxUserPattern(mlbertapi* instance, int channel, ulong pattern)
 {
  return mlBert_SetTxUserPattern (instance, channel, pattern);
 }
```

mlBert_SetTxUserPattern39B

This API call is used to set a user defined PRBS to be generated by the selected channel. SetPRBSPattern should be set to 5 “user Defined” in order for this API to work. The user defined pattern length is equal to 80bit, they are divided to 40 bit in 2 parameters.

Each channel should have a separate call.

API returns 1 upon success, 0 if failure.

Used for ML4039B-EQL, ML4039B and ML4039B-EQL-HV.

Example of Use:

```
/*
* mlBert_SetTxUserPattern39B EntryPoint inside the DLL allows to set user Defined Pattern to be
generated
*/

[DllImport(@"DLL/MLBert_API.dll", EntryPoint = " mlBert_SetTxUserPattern39B ")]
private static extern int mlBert_SetTxUserPattern39B (mlbertapi* instance, int channel, ulong
UserDefinedPatternMSB, ulong UserDefinedPatternLSB);

/*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Param int channel
* Param ulong PatternMSB (ex: 0xxxxxxxf0f0f0f0f0) hexadecimal
* Param ulong PatternLSB (ex: 0xxxxxxxf0f0f0f0f0) hexadecimal
* return int
* Allows setting a user defined PRBS in order to be generated by the channel.
* APISetPRBSPattern should be set to (user Defined = 5) in order for this API to work
* Returns 1 on success, 0 on failure.
* Channels are identified using the channel parameter
* Boards connected are identified using the instance parameter
*/

public int SetTxUserPattern39B(mlbertapi* instance, int channel, ulong patternMSB, ulong patternLSB)
 {
  return mlBert_SetTxUserPattern39B (instance, channel, patternMSB, patternLSB);
 }
```

mlBert_OutputLevel

This API call is used to set the amplitude per channel. Amplitude values sent could be between 0 and 1000 mV.

Each channel should have a separate call.

API returns 1 upon success, 0 if failure.

In case of Error in values the Board requires calibration.

Used for all BERTs.

Example of use:

```
/*
* mlBert_OutputLevel EntryPoint inside the DLL allows to set the Amplitude
*/

 [DllImport(@"DLL/MLBert_API.dll", EntryPoint = " mlBert_OutputLevel ")]
private static extern int mlBert_OutputLevel (mlbertapi* instance, int channel, double Amplitude);

/*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Param int channel
* Param double Amplitude
* return int
* Allows to set the amplitude per channel
* Returns 1 on success, 0 on failure.
* In case of Errors in values the Board requires calibration.
* Channels are identified using the channel parameter
* Boards connected are identified using the instance parameter
*/

public int OutputLevel(mlbertapi* instance, int channel, double Amplitude)
{
   return mlBert_OutputLevel (instance, channel, Amplitude);
 }
```

mlBert_CML_OutputDriver

This API call is used to set the CML amplitude for Pam4 for each channel. Amplitude values sent could be between 0 and 3 steps.

Each channel should have a separate call.

API returns 1 upon success, 0 if failure.

In case of Error in values the Board requires calibration.

Used for 4004PAM and 4039PAM-ATE.

Example of use:

```
/*
* mlBert_CML_OutputDriver EntryPoint inside the DLL allows to set the Amplitude
*/

 [DllImport(@"DLL/MLBert_API.dll", EntryPoint = " mlBert_CML_OutputDriver ")]
private static extern int mlBert_CML_OutputDriver (mlbertapi* instance, int channel, double value);

/*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Param int channel
* Param double Value (from 0 to 4 steps)
* return int
* Allows to set the CML amplitude per channel
* Returns 1 on success, 0 on failure.
* Channels are identified using the channel parameter
* Boards connected are identified using the instance parameter
*/

public int CML_OutputDrivre(mlbertapi* instance, int channel, double Amplitude)
{
   return mlBert_CML_OutputDriver (instance, channel, Amplitude);
 }
```

mlBert_PreEmphasis

This API call is used to set the Pre Emphasis level on a channel, Pre Emphasis level could be set between 0 -100 for Gearbox2, and from -1000 to 1000 for others.

Each channel should have a separate call.

API returns 1 upon success, 0 if failure.

Used for ML4009, ML4039, ML4004Pam4, ML4004, ML4039b, ML4039D, ML4079D, ML4039E, ML4039EN, ML4039EML.

Example of use:

```
/*
* mlBert_PreEmphasis EntryPoint inside the DLL allows to set the PreEmphasis level
*/

 [DllImport(@"DLL/MLBert_API.dll", EntryPoint = " mlBert_PreEmphasis ")]
private static extern int mlBert_PreEmphasis (mlbertapi* instance, int channel, int value);



/*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Param int channel
* Param int value : 0 -100 for Gearbox2, and from -1000 to 1000 for others.
* return int
* Returns 1 on success, 0 on failure.
* Allows setting the Pre Emphasis level on a channel
* Channels are identified using the channel parameter
* Boards connected are identified using the instance parameter
*/

public int PreEmphasis(mlbertapi* instance, int Channel, int value)
 {
  return mlBert_PreEmphasis (instance, Channel, value);
 }
```

mlBert_PostEmphasis

This API call is used to set the Post Emphasis level on a channel, Post Emphasis level could be set 0 -100 for Gearbox2, and from -1000 to 1000 for others.

Each channel should have a separate call.

API returns 1 upon success, 0 if failure.

Used for ML4009, ML4039, ML4004Pam4, ML4004, ML4039b, ML4039D, ML4079D, ML4039E, ML4039EN, ML4039EML.

Example of use:

```
/*
* mlBert_PostEmphasis EntryPoint inside the DLL allows to set the Post Emphasis level
*/

 [DllImport(@"DLL/MLBert_API.dll", EntryPoint = "mlBert_PostEmphasis")]
private static extern int mlBert_PostEmphasis (mlbertapi* instance, int channel, int value);



/*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Param int channel
* Param int value : 0 -100 for Gearbox2, and from -1000 to 1000 for others.
* return int
* Returns 1 on success, 0 on failure.
* Allows to set the Post Emphasis level on a channel
* Channels are identified using the channel parameter
* Boards connected are identified using the instance parameter
*/

public int PostEmphasis(mlbertapi* instance, int Channel, int value)
{
  return mlBert_PostEmphasis (instance, Channel, value);
 }
```

mlBert_DFEEnable

This API call is used to enable/disable DFE (Decision Feedback Equalizer) per channel, it should be called to enable DFE before setting it using “mlBert_DFESetValue”. It's also used to disable DFE after setting it to zero.

Each channel should have a separate call.

API returns 1 on success, 0 on failure.

Used for all BERTs.

Example of use:

```
/*
* mlBert_DFEEnable EntryPoint inside the DLL allows to enable/disable DFE (Decision Feedback
Equalizer),
* should be called before setting the DFE
*/

 [DllImport(@"DLL/MLBert_API.dll", EntryPoint = " mlBert_DFEEnable")]
 static extern int mlBert_DFEEnable (mlbertapi* instance, int channel, int status);



/*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Param int channel
* Param int status (DFE disabled = 0, DFE enabled= 1)
* return int
* Allows to enable/Disable DFE (Decision Feedback Equalizer)
* Returns 1 on success, 0 on failure.
* Channels are identified using the channel parameter
* Boards connected are identified using the instance parameter
*/

public int DFEEnable(mlbertapi* instance, int channel, int status)
 {
   return mlBert_DFEEnable (instance, channel, status);
}
```

mlBert_DFESetValue

This API is used to set the value of DFE (Decision Feedback Equalizer) . DFE should be enabled using "mlBert_DFEEnable" in order for this API to work properly

DFE varies between 0 and 14.

Each channel should have a separate call.

API returns 1 on success, 0 on failure.

Used for all BERTs.

Example of Use:

```
/*
* mlBert_DFESetValue EntryPoint inside the DLL allows to set the value of DFE (Decision Feedback
Equalizer)
*/

[DllImport(@"\dll\MLBert_API.dll", EntryPoint = " mlBert_DFESetValue")]
private static extern bool mlBert_DFESetValue (mlbertapi* instance, int channel, int value);

 /*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Param int channel
* Param int value (values from: 0 - 14)
* return int
* Allows to change the DFE value (Decision Feedback Equalizer)
* Returns 1 on success, 0 on failure.
* Channels are identified using the channel parameter
* Boards connected are identified using the instance parameter
*/

public int DFESetValue(mlbertapi* instance,int channel, int value)
{
return mlBert_DFESetValue (instance, channel, value);
 }
```

mlBert_Monitor

This API call is used to read the BERT configuration saved on the EEprom, it’s usually used after establishing a BERT connection. It returns 1 if successful, 0 if it was not able to read the value. Each channel should have a separate call.

Return 1 on success, 0 on failure.

Used for all BERTs except ML4039B-EQL, ML4039B and ML4039B-HV.

Example of use:

```
/*
* mlBert_Monitor EntryPoint inside the DLL allows to read the configuration set on the BERT connected
*/

[DllImport(@"DLL/MLBert_API.dll", EntryPoint = " mlBert_Monitor")]
private static extern int apiMonitor(mlbertapi* instance, byte Channel, byte StatusID,ref double Data);

/*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Param int channel
* Param byte StatusID
** StatusID 0 return 1 if clock source is set to Internal, and 0 for external
** StatusID 1 return the LineRate value
** StatusID 2 return The clock Out settings (0 = clock out, 1 for CDR/4 and 2 for CDR/16)
** StatusID 3 return for TX enable status
** StatusID 4 return for RX enable status
** StatusID 5 return pattern set on TX
** StatusID 6 return Custom pattern set on TX ( in case of ML4039B-EQL it returns the Cust Pattern LSB)
** StatusID 8 return if TX is inverted
** StatusID 9 return amplitude set on TX
** StatusID 10 return pattern set on RX
** StatusID 11 return if RX is inverted
** StatusID 13 return if channel is locked
** StatusID 14 return PreEmphasis value
** StatusID 15 return PostEmphasis value
** StatusID 16 return if error inseration is enabled or disabled
** StatusID 17 return DFE Value
** StatusID 18 return BER Timer
** StatusID 19 return BER Phase offset
** StatusID 20 return BER vertical offset
** StatusID 21 return Phase skew
** StatusID 22 return PM Frquency
** StatusID 23 return PM amplitude
** StatusID 24 return PM RJ amplitude
** StatusID 25 return AM RJ amplitude
** StatusID 29 return Internal loop back Status
** StatusID 30 return reference clock out value
** StatusID 31 return inner amplitude value(only for ML4004 pam4)
** StatusID 32 return PRE Coding (only for ML4004 pam4)
** StatusID 33 return GrayMapping (only for ML4004 pam4)
** StatusID 34 return FEC status (only for ML4004 pam4)
** StatusID 35 return Tx mode (only for ML4004 pam4)
** StatusID 36 return CML amplitude step (only for ML4004 pam4)
* ref UInt64 Data
* Return int
* Allows to read configuration set on the BERT, the associated value will be set to the Data param
* Return 1 on success, 0 on failure.
* Channels are identified using the channel parameter
* Boards connected are identified using the instance parameter
*/
public int Monitor(mlbertapi* instance, byte Channel, byte StatusID, ref double Data)
 {
return mlBert_Monitor (instance, Channel, StatusID,ref Data);
 }
```

mlBert_ErrorInserationMode

This API call is used to insert specific count of errors into a signal generated on a TX channel.

Each channel should have a separate call.

Return 1 on success, 0 on failure.

Used for all BERTs, except ML4039B-EQL and ML4039B.

Example of use:

```
/*
* mlBert_ErrorInserationMode EntryPoint inside the DLL allows to insert specific count of errors into a
* signal generated on a TX channel
*/

[DllImport(@"DLL/MLBert_API.dll", EntryPoint = " mlBert_ErrorInserationMode")]
static extern int mlBert_ErrorInserationMode (mlbertapi* instance, int channel, UInt16
errorCountToInject);

/*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Param int channel
* Param UInt16 errorCountToInject (number of error to be inserted)
** If Linerate > 20 pattern is 40 at a time
** If Linerate < 20 pattern is 16 at a time
* Return int
* Allows to insert specific count of errors into a signal generated on a TX channel
* Return 1 on success, 0 on failure.
* Channels are identified using the channel parameter
* Boards connected are identified using the instance parameter
*/

public int ErrorInserationMode(mlbertapi* instance, int channel, UInt16 errorCountToInject)
 {
return mlBert_ErrorInserationMode (instance, channel, errorCountToInject);
 }
```

mlBert_DoInstantBER

This API call is used to run BER test and return the BER for the specified BitCount in SetBERCounter function.

API returns 1 upon success, 0 if failure.

Used for all BERTs.

Example of use:

```
/*
* mlBert_DoInstantBER EntryPoint inside the DLL allows to run BER test*/

 [DllImport(@"DLL/MLBert_API.dll", EntryPoint = " mlBert_DoInstantBER")]
private static extern int mlBert_DoInstantBER (mlbertapi* instance, double berValues[]);

/*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Param double berValues: Return BER Value in a table for all 8 channels, need to send double table[8]
* Return int
* Return 1 on success, 0 on failure
* Boards connected are identified using the instance parameter
*/
public int DoInstantBER (mlbertapi* instance, double berValues[])
 {
  return mlBert_DoInstantBER (instance, berValues[]);
 }
```

mlBert_ChangeBERPhase

This API call is used to change the sampling point position of the RX checker.

API returns 1 upon success, 0 if failure.

Used for all BERTs.

Example of use:

```
/*
* mlBert_ChangeBERPhase EntryPoint inside the DLL allows to set the sampling point of the RX checker.
API should be called before “mlBert_DoInstantBER”
*/

 [DllImport(@"DLL/MLBert_API.dll", EntryPoint = " mlBert_ChangeBERPhase")]
private static extern int mlBert_ChangeBERPhase (mlbertapi* instance, byte Phase, byte VerticalOffset);

/*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Param byte Phase (0 to 127) and (0 to 64 for ML4039B-EQL)
* Param byte VerticalOffset (0 to 255)
* Return int
* Return 1 on success, 0 on failure
* Boards connected are identified using the instance parameter
*/
public int ChangeBERPhase(mlbertapi* instance, byte Phase, byte VerticalOffset)
 {
   return mlBert_ChangeBERPhase (instance, Phase, VerticalOffset);
 }
```

mlBert_RealTimeBEREnable

This API call is used to enable/disable Real Time BER. Results will be collected using mlBert_RealTimeBER.

Return 1 upon success, 0 if failure.

Used for all BERTs.

Example of use:

```
/*
* mlBert_RealTimeBEREnable EntryPoint inside the DLL allows to enable/disable real time BER
* Results will be collected using mlBert_RealTimeBEREnable
*/

 [DllImport(@"DLL/MLBert_API.dll", EntryPoint = " mlBert_RealTimeBEREnable")]
 static extern int APIRealTimeBEREnable (mlbertapi* instance, int enable);

/*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Param int enable (Enable = 1, Disable = 0)
* Return int
* Allows to enable/disable real time BER
* Return 1 on success, 0 on failure
* Boards connected are identified using the instance parameter
*/

public int RealTimeBEREnable (mlbertapi* instance, int enable)
{
  mlBert_RealTimeBEREnable (instance, enable);
 }
```

mlBert_ReadRealTimeBER

This API call is used to read BER values from the time the “mlBert_RealTimeBEREnable” was called. The API will keep track of the elapsed time through a reference value time that get updated every call

This call should be repeated periodically. The returned result will be cumulative of BER array on all channels.

Return 1 upon success, 0 if failure.

Used for all BERTs.

Example of use:

```
/*
* mlBert_ReadRealTimeBER EntryPoint inside the DLL allows to read Real Time BER Collected from the
time the “mlBert_RealTimeBEREnable” was started
*/

 [DllImport(@"DLL/MLBert_API.dll", EntryPoint = " mlBert_ReadRealTimeBER ")]
private static extern int mlBert_ReadRealTimeBER (mlbertapi* instance, UInt64[] errorCount, ref UInt32
Time);

/*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* UInt64 [8] errorCount (Cumulative BER values will be stored is this array)
* ref UInt32 Time (reference value that return the value from when the APIRealTimeBEREnable has
started)
* Return int
* Allows to collect real time BER with time tracking
* Return 1 on success, 0 on failure.
* Measurements are collected on 4 RX channels
* Boards connected are identified using the instance parameter
*/

public int ReadRealTimeBER (mlbertapi* instance, UInt64[] errorCount, ref UInt32 Time)
 {
   mlBert_ReadRealTimeBER (instance, errorCount, ref Time);
 }
```

mlBert_GetEye

This API call is used to read the Eye from the board for a specific RX channel.

Each channel should have a separate call.

API returns 1 on success, 0 on failure.

Used for all BERTs except PAM.

Example of use:

```
/*
* mlBert_GetEye EntryPoint inside the DLL allows to read the Eye from a specific RX channel
 */

[DllImport(@"DLL/MLBert_API.dll", EntryPoint = " mlBert_GetEye ")]
private static extern int mlBert_GetEye (mlbertapi* instance, int Channel, double[] ScopeSamples);



/*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Param int channel
* double[131072] ScopeSamplesEye (EYE values will be stored is this array)
* Return int
* Allows to read the Eye from the board for a specific channel
* Returns 1 on success, 0 on failure.
* Channels are identified using the channel parameter
* Boards connected are identified using the instance parameter
*/

public int GetEye(mlbertapi* instance, byte channel, double[] ScopeSamplesEye)
 {
  return mlBert_GetEye (instance, channel, ScopeSamplesEye);
 }
```

mlBert_BathTubVerticalOffset

This API call is used to set the vertical point for which the BathTub is measured from the board for a specific RX channel.

Each channel should have a separate call.

API returns 1 upon success, 0 if failure.

Used for all BERTs.

Example of use:

```
/*
* mlBert_BathTubVerticalOffset EntryPoint inside the DLL allows to set the vertical sampling point of the
bathtub measurement for a specific RX channel
*/

 [DllImport(@"DLL/MLBert_API.dll", EntryPoint = " mlBert_BathTubVerticalOffset ")]
private static extern int mlBert_BathTubVerticalOffset (mlbertapi* instance, byte VerticalOffset);



/*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Param int channel
* Param byte vertical offset
* Return int
* Allows to set the vertical sampling point of the bathtub measurement for a specific channel
* Returns 1 on success, 0 on failure.
* Channels are identified using the channel parameter
* Boards connected are identified using the instance parameter
*/

public int SetBathTubVerticalOffset(mlbertapi* instance, int channel, byte vertical_offset)
 {
return mlBert_BathTubVerticalOffset (instance, channel, vertical_Offset);
}
```

mlBert_FastDiamond

This API call is used to get Fast eye Width and Height from received signal.

Each channel should have a separate call.

API returns 1 upon success, 0 if failure.

Used for all BERTs except PAM.

Example of use:

```
/*
* mlBert_FastDiamond EntryPoint inside the DLL allows to get the eye width and eye height of the
received signal for a specific RX channel.
*/

 [DllImport(@"DLL/MLBert_API.dll", EntryPoint = " mlBert_FastDiamond ")]
private static extern int mlBert_FastDiamond (mlbertapi* instance, int channel ,double ref EyeWidth,
Double ref EyeHeight );

/*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Param int channel
* Param ref double EyeWidth in ps
* Param ref double EyeHeight in mV
* Return int
* Allows to get the eye width directly in ps and eye height in mV, Returns 1 on success, 0 on failure.
* Channels are identified using the channel parameter
* Boards connected are identified using the instance parameter
*/

public int FastDiamond (mlbertapi* instance, int channel, double ref width, double ref Height)
 {
return mlBert_FastDiamond (instance, channel, ref width , ref Height);
}
```

mlBert_GetBathTub

This API call is used to read the BathTub from the board for a specific RX channel.

Each channel should have a separate call.

API returns 1 upon success, 0 if failure.

Used for all BERTs.

Example of use:

```
/*
* mlBert_GetBathTub EntryPoint inside the DLL allows to read the BathTub from a specific RX channel
*/

 [DllImport(@"DLL/MLBert_API.dll", EntryPoint = " mlBert_GetBathTub ")]
private static extern int mlBert_GetBathTub (mlbertapi* instance, int channel, double xValues, double
BERValues);



/*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Param int channel
* double[1024] xValues (BathTub values will be stored is this array)
* double[1024] BERValues: return the BERValues of a specific xValue in an array of 1024 values. In case
the user needs to implement the dual dirac, he needs to pass these 2 arrays into the dual dirac function.
* Return int
* Allows to read the BathThub curves from the BERT for a specific channel
* Returns 1 on success, 0 on failure.
* Channels are identified using the channel parameter
* Boards connected are identified using the instance parameter
*/

public int GetBathTubCurve(mlbertapi* instance, int channel, double xValues, double BERValues)
 {
return mlBert_GetBathTub (instance, channel, xValues, BERValues);
}
```

mlBert_TempRead

This API call is used to read the board's temperature. There are two sensors to read temperature identified by the index parameter

Return double.

Used for all BERTs.

Example of use:

```
/*
* mlBert_TempRead EntryPoint inside the DLL allows to read Board temperature
*/

 [DllImport(@"DLL/MLBert_API.dll", EntryPoint = " mlBert_TempRead ")]
private static extern double mlBert_TempRead (mlbertapi* instance, int index);

/*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Param int Index (0 first Sensor, 1 Second Sensor)
* Return double
* Allows to read to read temperature
* Boards connected are identified using the instance parameter
*/

public double TempRead(mlbertapi* instance, int index)
 {
   return mlBert_TempRead (instance, index);
 }
```

mlBert_Disconnect

This API is used to disconnect the BERT.

Return 1 on success, 0 on failure.

Used for all BERTs.

Example of use:

```
/*
* mlBert_Disconnect EntryPoint inside the DLL is responsible to disconnect to the BERT
*/

 [DllImport(@"DLL/MLBert_API.dll", EntryPoint = " mlBert_Disconnect ")]
private static extern int mlBert_Disconnect (mlbertapi* instance);

/*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Return int
* Allows to Disconnect from the Board
* Return 1 on success, 0 on failure.
* Boards connected are identified using the instance parameter
*/

public int Disconnect(mlbertapi* instance)
 {
return mlBert_Disconnect (instance);
}
```

mlBert_LoadDefaultCalibration (Debug use only)

This API call is used to Just for Testing in case the board is not calibrated, It will load default calibration, the “APIOutputLevel” max value will be 100 instead of 1000

Return 1 on success, 0 on failure.

Used for all BERTs.

Example of use:

```
/*
* mlBert_LoadDefaultCalibration EntryPoint inside the DLL that will load the board default calibration in
case * the BERT is not calibrated
*/

 [DllImport(@"DLL/MLBert_API.dll", EntryPoint = " mlBert_LoadDefaultCalibration ")]
private static extern int mlBert_LoadDefaultCalibration (mlbertapi* instance);

/*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Return int
* Allows to will load the board default calibration
* Return 1 on success, 0 on failure.
* Boards connected are identified using the instance parameter
*/

public int LoadDefaultCalibration(mlbertapi* instance)
{
return mlBert_LoadDefaultCalibration (instance);
 }
```

mlBert_AccessBoardRegister (Debug use only)

This API call is used to direct access BERT registery in order to read and write values. This API is for used only for debugging

Return 1 on success, 0 on failure.

Used for all BERTs.

Example of use:

```
/*
* mlBert_AccessBoardRegister EntryPoint inside the DLL that allows direct access to BERT registery in order * to read and write values
*/

[DllImport(@"DLL/MLBert_API.dll", EntryPoint = " mlBert_AccessBoardRegister ")]
private static extern int mlBert_AccessBoardRegister (mlbertapi* instance, byte Read_Write, UInt16 channel, UInt16 Reg, ref UInt16 Data);

/*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Param byte Read_Write (0 = Read, 1 = Write)
* Param UInt16 channel
* Param UInt16 Reg (Register address that want to read/write from)
* Param ref UInt16 Data (Case Write will contain the data to write, Case Read will return the data inside the register)
* Return int
* Allows direct access to BERT registery in order to read and write values
* Return 1 on success, 0 on failure.
* Boards connected are identified using the instance parameter
*/

public int AccessBoardRegister(mlbertapi* instance, byte Read_Write, UInt16 channel, UInt16 Reg, ref UInt16 Data)
 {
return mlBert_AccessBoardRegister (instance, Read_Write, channel, Reg, ref Data);
  }
```

mlBert_GetVBathTub

This API call is used to read the vertical BathTub from a specific channel.

Returns 1 on success, 0 on failure.

Used for all BERTs.

Example of use:

```
/*
* mlBert_GetVBathTub EntryPoint inside the DLL allows to read the verical BathTub from a specific
channel
*/

 [DllImport(@"DLL/MLBert_API.dll", EntryPoint = " mlBert_GetVBathTub ")]
private static extern int mlBert_GetVBathTub (mlbertapi* instance, int channel, byte HorizontalOffset,
double *xValues, double *BERValues);



/*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Param int channel
* Param byte HorizontalOffset
* Param double xValues
* Param double BERValues
* Return int
* Allows to read the VerticalBathThub curves from the BERT for a specific channel
* Returns 1 on success, 0 on failure.
* Channels are identified using the channel parameter
*/

Public int GetVBathTub(mlbertapi* instance, int channel, byte HorizontalOffset, double *xValues, double
*BERValues)
 {
return mlBert_GetVBathTub (instance, channel, HorizontalOffset, xValues, BERValues);
}
```

mlBert_ReadRJDJ

This API call is used to read the DJ and RJ for a specific channel. The user must call a prerequisite function "mlBert _CalculateBathtubDualDirac" after reading a bathtub curve using this function "mlBert _GetBathtub".

Returns true on success, false on failure.

Used for all BERTs except PAM.

Example of use:

```
/*
* mlBert_ReadRJDJ EntryPoint inside the DLL allows to read the DJ and RJ
*/

 [DllImport(@"DLL/MLBert_API.dll", EntryPoint = " mlBert_ReadRJDJ ")]
private static extern bool mlBert_ReadRJDJ (mlbertapi* instance, int channel, double ref DJ_array,
double ref RJ_array);

/*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Param int channel
* Param DJ array for the 4 channels
* Param RJ array for the 4 channels
* Return bool
* Allows to read the DJ and RJ
* Returns true on success, false on failure.
* Channels are identified using the channel parameter
*/

Public bool ReadRJDJ(mlbertapi* instance, int channel, double *DJ_array, double *RJ_array)

{
return mlBert_ReadRJDJ (instance, channel, DJ_array, RJ_array);
}
```

mlBert_CalculateBathtubDualDirac

This API call is used to apply Dual Dirac on the pre-captured bathtub.

This function required using mlBert_GetBathtub, the X and Y values for the Bathtub extracted from the APIGetBathtub should be used as parameters in this function (second and third parameter).
After passing this function, the user should use these two parameters to draw the BathTub in Dual Dirac mode.

Returns 1 on success, 0 on failure.

Used for all BERTs.

Example of use:

```
/*
* mlBert_CalculateBathtubDualDirac EntryPoint inside the DLL apply Dual Dirac on the pre-captures
bathtub.
*/

 [DllImport(@"DLL/MLBert_API.dll", EntryPoint = " mlBert_CalculateBathtubDualDirac")]
private static extern int mlBert_CalculateBathtubDualDirac (mlbertapi* instance, int channel, double[]
xValues, double[] BERValues);

/*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Param int channel
* Param double[] xValues
* Param double[] BERValues
* Return int
* Allows to apply Dual Dirac on the pre-captures bathtub
* Returns 1 on success, 0 on failure.
* Channels are identified using the channel parameter
*/

Public int CalculateBathtubDualDirac (mlbertapi* instance, int channel, double *xValues, double
*BERValues)

{
return mlBert_CalculateBathtubDualDirac (instance, channel, xValues, BERValues);
}
```

mlBert_AMEnable

This API call is used to enable and disable the AM.

Returns 1 on success, 0 on failure.

Used for ML4009-JIT, ML4039-JIT, ML4039B-EQL, ML4039B and ML4004-JIT.

Example of use:

```
/*
* mlBert_AMEnable EntryPoint inside the DLL allows enabling and disabling the AM.
*/

 [DllImport(@"DLL/MLBert_API.dll", EntryPoint = " mlBert_AMEnable ")]
private static extern int mlBert_AMEnable (mlbertapi* instance, int channel, int status);

/*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Param int channel
* Param int status: 0=disable, 1=enable
* Return int
* Allows to enable and disable the AM
* Returns 1 on success, 0 on failure.
* Channels are identified using the channel parameter
*/

public int AMEnable (mlbertapi* instance, int channel, int status)

{
return mlBert_AMEnable (instance, channel, status);
}
```

mlBert_PMEnable

This API call is used to enable and disable the PM.

Returns 1 on success, 0 on failure.

Used for ML4009-JIT, ML4039-JIT, ML4039B-EQL, ML4039B and ML4004-JIT.

Example of use:

```
/*
* mlBert_PMEnable EntryPoint inside the DLL allows enabling and disabling the PM.
*/

 [DllImport(@"DLL/MLBert_API.dll", EntryPoint = " mlBert_PMEnable ")]
private static extern int mlBert_PMEnable (mlbertapi* instance, int channel, int status);

/*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Param int channel
* Param int status: 0=disable, 1=enable
* Return int
* Allows to enable and disable the PM
* Returns 1 on success, 0 on failure.
* Channels are identified using the channel parameter
*/

Public int PMEnable (mlbertapi* instance, int channel, int status)

{
return mlBert_PMEnable (instance, channel, status);
}
```

mlBert_SetPMFrequency

This API call is used to set the PM Frequency.

Returns 1 on success, 0 on failure.

Used for ML4009-JIT, ML4039-JIT, ML4039B-EQL, ML4039B and ML4004-JIT.

Example of use:

```
/*
* mlBert_SetPMFrequency EntryPoint inside the DLL allows setting the PM Frequency.
*/

 [DllImport(@"DLL/MLBert_API.dll", EntryPoint = " mlBert_SetPMFrequency ")]
private static extern int mlBert_SetPMFrequency (mlbertapi* instance, int channel, double frequency);

/*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Param int channel
* Param double frequency
* Return int
* Allows to set the PM Frequency
* Returns 1 on success, 0 on failure.
* Channels are identified using the channel parameter
*/

Public int SetPMFrequency (mlbertapi* instance, int channel, double frequency)

{
return mlBert_SetPMFrequency (instance, channel, frequency);
}
```

mlBert_PMSJAmplitude

This API call is used to change the SJ value.

Returns 1 on success, 0 on failure.

Used for ML4009-JIT, ML4039-JIT, ML4039B-EQL, ML4039B and ML4004-JIT.

Example of use:

```
/*
* mlBert_PMSJAmplitude EntryPoint inside the DLL allows changing the SJ value.
*/

 [DllImport(@"DLL/MLBert_API.dll", EntryPoint = " mlBert_PMSJAmplitude ")]
private static extern int mlBert_PMSJAmplitude (mlbertapi* instance, int channel, double value);

/*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Param int channel
* Param double value between 0 and 4095
* Return int
* Allows to change the SJ value
* Returns 1 on success, 0 on failure.
* Channels are identified using the channel parameter
*/

Public int PMSJAmplitude (mlbertapi* instance, int channel, double value)

{
return mlBert_PMSJAmplitude (instance, channel, value);
}
```

mlBert_PMRJEnable

This API call is used to enable and disable the RJ.

Returns 1 on success, 0 on failure.

Used for ML4009-JIT, ML4039-JIT, ML4039B-EQL, ML4039B and ML4004-JIT.

Example of use:

```
/*
* mlBert_PMRJEnable EntryPoint inside the DLL allows enabling and disabling the RJ.
*/

 [DllImport(@"DLL/MLBert_API.dll", EntryPoint = " mlBert_PMRJEnable ")]
private static extern int mlBert_PMRJEnable (mlbertapi* instance, int channel, int status);

/*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Param int channel
* Param int status: 0=disable, 1=enable
* Return int
* Allows to enable and disable the RJ
* Returns 1 on success, 0 on failure.
* Channels are identified using the channel parameter
*/

Public int PMRJEnable (mlbertapi* instance, int channel, int status)

{
return mlBert_PMRJEnable (instance, channel, status);
}
```

mlBert_PMRJAmplitude

This API call is used to change the Diff Noise value.

Returns 1 on success, 0 on failure.

Used for ML4009-JIT, ML4039-JIT, ML4039B-EQL, ML4039B and ML4004-JIT.

Example of use:

```
/*
* mlBert_PMRJAmplitude EntryPoint inside the DLL allows changing the Diff Noise value.
*/

 [DllImport(@"DLL/MLBert_API.dll", EntryPoint = " mlBert_PMRJAmplitude ")]
private static extern int mlBert_PMRJAmplitude (mlbertapi* instance, int channel, double value);

/*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Param int channel
* Param double value between 0 and 4095
* Return int
* Allows to change the Diff Noise value
* Returns 1 on success, 0 on failure.
* Channels are identified using the channel parameter
*/

Public int PMRJAmplitude (mlbertapi* instance, int channel, double value)

{
return mlBert_PMRJAmplitude (instance, channel, value);
}
```

mlBert_AMRJAmplitude

This API call is used to change the RJ value.

Returns 1 on success, 0 on failure.

Used for ML4009-JIT, ML4039-JIT, ML4039B-EQL, ML4039B and ML4004-JIT.

Example of use:

```
/*
* mlBert_AMRJAmplitude EntryPoint inside the DLL allows changing the RJ value.
*/

 [DllImport(@"DLL/MLBert_API.dll", EntryPoint = " mlBert_AMRJAmplitude ")]
private static extern int mlBert_AMRJAmplitude (mlbertapi* instance, int channel, double value);

/*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Param int channel
* Param double value between 0 and 4095
* Return int
* Allows to change the RJ value
* Returns 1 on success, 0 on failure.
* Channels are identified using the channel parameter
*/

Public int AMRJAmplitude (mlbertapi* instance, int channel, double value)

{
return mlBert_AMRJAmplitude (mlbertapi* instance, int channel, double value);
}
```

mlBert_PhaseSkew

This API call is used to change the TX Phase Skew of the signal.

Returns 1 on success, 0 on failure.

Used for ML4009-JIT, ML4039-JIT, ML4039B-EQL, ML4039B and ML4004-JIT.

Example of use:

```
/*
* mlBert_PhaseSkew EntryPoint inside the DLL allows changing the TX Phase Skew of the signal.
*/

 [DllImport(@"DLL/MLBert_API.dll", EntryPoint = " mlBert_PhaseSkew ")]
private static extern int mlBert_PhaseSkew (mlbertapi* instance, int channel, double amplitude);

/*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Param int channel
* Param double amplitude between 0 and 4096
* Return int
* Allows to change TX Phase Skew of the signal
* Returns 1 on success, 0 on failure.
* Channels are identified using the channel parameter
*/

Public int PhaseSkew (mlbertapi* instance, int channel, double amplitude)

{
return mlBert_PhaseSkew (instance, channel, amplitude);
}
```

mlBert_ClockOut

This API call is used to specify clock out source, the user can set the source to be form the internal refernce clock, or form the CDR. In case of CDR the user can select the channel source and the divsion rate (/8 or /16).

Returns 1 on success, 0 on failure.

Used for all BERTs except ML4039B-EQL and ML4039B.

Example of use:

```
/*
* mlBert_ClockOut EntryPoint inside the DLL allows specifying clock out.
*/

 [DllImport(@"DLL/MLBert_API.dll", EntryPoint = " mlBert_ClockOut ")]
private static extern int mlBert_ClockOut (mlbertapi* instance, int channel, int clockIndex);

/*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Param int channel
* Param int clockIndex: 0 refers to internal refernce clock, 1 refers to CDR /8, 2 refers to CDR/
* Return int
* Allows to specify clock out
* Returns 1 on success, 0 on failure
* Channels are identified using the channel parameter
*/
Public int ClockOut (mlbertapi* instance, int channel, int clockIndex)

{
return mlBert_ClockOut (instance, channel, clockIndex);
}
```

mlBert_SetBERCounter

This API call is used to set the BER Timer in ms.

Returns 1 on success, 0 on failure.

Used for all BERTs.

Example of use:

```
/*
* mlBert_SetBERCounter EntryPoint inside the DLL allows setting the BER Timer in ms.
*/

 [DllImport(@"DLL/MLBert_API.dll", EntryPoint = " mlBert_SetBERCounter ")]
private static extern int mlBert_SetBERCounter (mlbertapi* instance, Uint32 packetCounter);

/*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Param Uint32 packetCounter
* Return int
* Allows to set the BER Timer in ms
* Returns 1 on success, 0 on failure
* Channels are identified using the channel parameter
*/

Public int SetBERCounter (mlbertapi* instance, Uint32 packetCounter)

{
return mlBert_SetBERCounter (instance, packetCounter);
}
```

mlBert_DoInstantBER

This API call is used to run BER Test and return the BER for the specified BitCount in SetBERCounter function.

Returns 1 on success, 0 on failure.

Used for all BERTs.

Example of use:

```
/*
* mlBert_DoInstantBER EntryPoint inside the DLL allows running BER Test and return the BER for the
specified BitCount in SetBERCounter function.
*/

 [DllImport(@"DLL/MLBert_API.dll", EntryPoint = " mlBert_DoInstantBER ")]
private static extern int mlBert_DoInstantBER (mlbertapi* instance, double berValues[]);

/*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Param double berValues
* Return int
* Allows to run BER Test and return the BER for the specified BitCount in SetBERCounterfunction
* Returns 1 on success, 0 on failure
* Channels are identified using the channel parameter
*/

Public int DoInstantBER (mlbertapi* instance, double berValues[])

{
return mlBert_DoInstantBER (instance, berValues[]);

}
```

mlBert_GetFullSweepBathTub

This API call is used to scan the BathTub in full sweep mode where each phase scan take the exact same time and this time is set by the user.

Returns 1 on success, 0 on failure.

Used for all BERTs, except ML4039B-EQL and ML4039B.

Example of use:

```
/*
* mlBert_GetFullSweepBathTub EntryPoint inside the DLL allows scanning the BathTub in full sweep
mode*/

 [DllImport(@"DLL/MLBert_API.dll", EntryPoint = " mlBert_GetFullSweepBathTub ")]
private static extern int mlBert_GetFullSweepBathTub (mlbertapi* instance, int channel, unsigned long
long packetWaitCount, byte start, byte stop, double *xValues, double *BERValues);

/*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Param int channel
* Param unsigned packetWaitCount
* Param byte start
* Param byte stop
* Param double xValues
* Param double BERValues
* Return int
* Allows to scan the BathTub in full sweep mode
* Returns 1 on success, 0 on failure
* Channels are identified using the channel parameter
*/

Public int GetFullSweepBathTub (mlbertapi* instance, int channel, unsigned long long packetWaitCount,
byte start, byte stop, double *xValues, double *BERValues)

{
return mlBert_GetFullSweepBathTub (instance, channel, packetWaitCount, start, stop, xValues,
BERValues);

}
```

mlBert_CalculateJitter

This API call is used to calculate the jitter from current BathTub.

Returns 1 on success, 0 on failure.

Used for all BERTs.

Example of use:

```
/*
* mlBert_CalculateJitter EntryPoint inside the DLL allows calculating the jitter from current BathTub*/

 [DllImport(@"DLL/MLBert_API.dll", EntryPoint = " mlBert_CalculateJitter ")]
private static extern int mlBert_CalculateJitter (mlbertapi* instance, int channel, double targetBER,
double ref jitterMeasurements);

/*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Param int channel
* Param double targetBER
* Param double jitterMeasurements
* Return int
* Allows to calculate the jitter from current BathTub
* Returns 1 on success, 0 on failure
* Channels are identified using the channel parameter
*/

Public int CalculateJitter (mlbertapi* instance, int channel, double targetBER, double
*jitterMeasurements)

{
return mlBert_CalculateJitter (instance, channel, targetBER, jitterMeasurements);

}
```

mlBert_GetEyeMeasurements

This API call is used to measure the eye width and eye height for a specific BER.

Returns 1 on success, 0 on failure.

Used for all BERTs.

Example of use:

```
/*
* mlBert_GetEyeMeasurements EntryPoint inside the DLL allows measuring the eye width and eye
height for a specific BER*/

 [DllImport(@"DLL/MLBert_API.dll", EntryPoint = " mlBert_GetEyeMeasurements ")]
private static extern int mlBert_GetEyeMeasurements (mlbertapi* instance,int channel , double
targetBER, double *eyeWidthStart, double *eyeWidthEnd, double *eyeHeightStart, double
*eyeHeightEnd);

/*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Param int channel
* Param double targetBER
* Param double eyeWidthStart
* Param double eyeWidthEnd
* Param double eyeHeightStart
* Param double eyeHeightEnd
* Return int
* Allows to measure the eye width and eye height for a specific BER
* Returns 1 on success, 0 on failure
* Channels are identified using the channel parameter
*/

Public int GetEyeMeasurements (mlbertapi* instance,int channel , double targetBER, double
*eyeWidthStart, double *eyeWidthEnd, double *eyeHeightStart, double *eyeHeightEnd)

{
return mlBert_GetEyeMeasurements (instance, channel , targetBER, eyeWidthStart, eyeWidthEnd,
eyeHeightStart, eyeHeightEnd);

}
```

mlBert_ReadFirmwareRevision

This API call is used to read Firmware Revision loaded in the board.

Returns Firmware Revision as double.

Used for all BERTs.

Example of use:

```
/*
* mlBert_ReadFirmwareRevision EntryPoint inside the DLL allows reading Firmware Revision loaded in
the board.
*/

 [DllImport(@"DLL/MLBert_API.dll", EntryPoint = " mlBert_ReadFirmwareRevision ")]
private static extern double mlBert_ReadFirmwareRevision (mlbertapi* instance);

/*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Return double
* Allows to read Firmware Revision
* Returns Firmware Revision
* Channels are identified using the channel parameter
*/

Public int ReadFirmwareRevision (mlbertapi* instance)

{
return mlBert_ReadFirmwareRevision (instance);
}
```

mlBert_RXEnable

This API call is used to enable or disable the RX Line.

Returns 1 on success, 0 on failure.

Used for all BERTs.

Example of use:

```
/*
* mlBert_RXEnable EntryPoint inside the DLL allows enabling or disabling the RX line.
*/

 [DllImport(@"DLL/MLBert_API.dll", EntryPoint = " mlBert_RXEnable ")]
private static extern int mlBert_RXEnable (mlbertapi* instance, int channel, bool status);

/*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Param int channel
* Param bool status: indicates the RX Line status
* Return int
* Allows to enable and disable the RX
* Returns 1 on success, 0 on failure.
* Channels are identified using the channel parameter
*/

Public int RXEnable (mlbertapi* instance, int channel, bool status)

{
return mlBert_RXEnable (instance, channel, status);
}
```

mlBert_TXEnable

This API call is used to enable or disable the TX Line.

Returns 1 on success, 0 on failure.

Used for all BERTs.

Example of use:

```
/*
* mlBert_TXEnable EntryPoint inside the DLL allows enabling or disabling the TX line.
*/

 [DllImport(@"DLL/MLBert_API.dll", EntryPoint = " mlBert_TXEnable ")]
private static extern int mlBert_TXEnable (mlbertapi* instance, int channel, bool status);

/*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Param int channel
* Param bool status: indicates the TX Line status
* Return int
* Allows to enable and disable the TX
* Returns 1 on success, 0 on failure.
* Channels are identified using the channel parameter
*/

Public int TXEnable (mlbertapi* instance, int channel, bool status)

{
return mlBert_TXEnable (instance, channel, status);
}
```

mlBert_CalculateMaskMeasurements

This API call is used to apply the mask test on the measured eye.

It required running GetEye first.

Returns the number of failing points.

Used for all BERTs.

Example of use:

```
/*
* mlBert_CalculateMaskMeasurements EntryPoint inside the DLL allows making the mask test on the
measured eye.
*/

 [DllImport(@"DLL/MLBert_API.dll", EntryPoint = " mlBert_CalculateMaskMeasurements ")]
private static extern int mlBert_CalculateMaskMeasurements (mlbertapi* instance, int channel, double
x1, double x2, double x3, double x4, double x5, double x6, double y0, double y1, double y2, double y3,
double y4, double y5, double y6);

/*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Param int channel
* Param double x1, x2, x3, x4, x5, x6 (Mask X values in ps)
*Param double y0, y1, y2, y3, y4, y5, y6 (Mask Y values in mV)
* Return int
* Allows to count the number of failing points
* Returns 1 on success, 0 on failure.
* Channels are identified using the channel parameter
*/

Public int CalculateMaskMeasurements (mlbertapi* instance, int channel, double x1, double x2, double
x3, double x4, double x5, double x6, double y0, double y1, double y2, double y3, double y4, double y5,
double y6)

{
return mlBert_CalculateMaskMeasurements (instance, channel, x1, x2, x3, x4, x5, x6, y0, y1, y2, y3, y4,
y5, y6);
}
```

mlBert_ReadCalibrationValues

This API call is used to read the calibration values in the EEPROM.

Returns 1 on success, 0 on failure.

Used for all BERTs.

Example of use:

```
/*
* mlBert_ReadCalibrationValues EntryPoint inside the DLL allows reading the calibration values in the
EEPROM*/

 [DllImport(@"DLL/MLBert_API.dll", EntryPoint = " mlBert_ReadCalibrationValues ")]
private static extern int mlBert_ReadCalibrationValues (mlbertapi* instance, double* values);

/*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Param values: array of double that read the calibration values
* Return int
* Allows to read the calibration values in the EEPROM
* Returns 1 on success, 0 on failure.
*/

Public int ReadCalibrationValues (mlbertapi* instance, double* values)

{
return mlBert_ReadCalibrationValues (instance, values);
}
```

mlBert_CheckEQLRevision

This API call is used to check the 4009-EQL Revision number.

Returns true on success, false on failure.

Used for ML4009-EQL.

Example of use:

```
/*
* mlBert_CheckEQLRevision EntryPoint inside the DLL allows checking 4009-EQL Revision number*/

 [DllImport(@"DLL/MLBert_API.dll", EntryPoint = " mlBert_CheckEQLRevision ")]
private static extern bool mlBert_CheckEQLRevision (mlbertapi* instance, int*Revision);

/*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Param int Revision, if 0: Rev A, 0xb: Rev B
* Return bool
* Allows to check 4009-EQL Revision number
* Returns true on success, false on failure.
*/

Public bool CheckEQLRevision (mlbertapi* instance, int*Revision)

{
return mlBert_CheckEQLRevision (instance, Revision);
}
```

mlBert_EnableInternalLoopback

This API call is used to enable the Internal Loopback.

Returns true on success, false on failure.

Used for all Gearbox2 berts.

Example of use:

```
/*
* mlBert_EnableInternalLoopbcak EntryPoint inside the DLL allow enabling the internal Loopback*/

 [DllImport(@"DLL/MLBert_API.dll", EntryPoint = " mlBert_EnableInternalLoopbcak ")]
private static extern bool mlBert_EnableInternalLoopbcak (mlbertapi* instance, int Enable);

/*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Param int Enable, if 0: Disabled, 1: Enabled
* Return bool
* Allows to enable the internal Loopback
* Returns true on success, false on failure.
*/

Public bool EnableInternalLoopback (mlbertapi* instance, int Enable)

{
return mlBert_EnableInternalLoopbcak (instance, enable);
}
```

mlBert_LineRateConfigurationBert39B

This API call is used to set the Line Rate of the ML4039B EQL. It returns 1 if successful, 0 if a problem occurs while setting the line Rate.

For every Line Rate a File should be provided or generated that will configure the Silab, location of the file will be set by using “APIConfigureApplication”.

To generate the clock file the user should use a special GUI provided by MultiLane.

Used for ML4039B-EQL only.

Example of use:

```
/*
 *  mlBert_LineRateConfigurationBert39B EntryPoint inside the DLL is responsible to set the Line Rate
*/
[DllImport(@"DLL/MLBert_API.dll", EntryPoint = " mlBert_LineRateConfigurationBert39B ")]
private static extern int mlBert_LineRateConfigurationBert39B (mlbertapi* instance, int channelIndex,
double linerate1, double linerate2, int clockSource1, int clockSource2);
/*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Param int channelIndex : if 0 the linerate1 and linerate2 refers to channel1 and channel2, if 1 they
refers to channel3 and channel4
* Param double lineRate1 : contains the linerate (for channel1 if channelindex=0, channel3 if
channelindex=1) required in Gbps ranging from 1-15 and 15-33
* Param double lineRate2 : contains the linerate (for channel2 if channelindex=0, channel4 if
channelindex=1) required in Gbps ranging from 1-15 and 15-33
*Param int clockSource: External=0, Internal=1
NB: -The clock file should be included in the specified directory in ConfigureApplication:saveConfig
parameter (default:clk)
-Configuration will be lost after applying linerate, and can be restored automatically using
RestoreAllConfig() function
* return int
* Allows to set Line Rate generated by the ML4039B EQL
* DLL use lineRate value to set the Line Rate of the ML4039B EQL
*/
public int LineRateConfigurationBert39B(mlbertapi* instance, int channelIndex, double linerate1, double
linerate2, int clockSource1, int clockSource2)
 {
return mlBert_LineRateConfigurationBert39B (instance, channelIndex, linerate1, linerate2,
clockSource1, clockSource2);
 }
```

mlBert_LoadDynamicStressers

This API call is used to set the Dynamic stressers configuration

API returns 1 upon success, 0 if failure.

Used for ML4009-EQL and ML4009-JIT EQL only.

Example of Use:

```
/*
* mlBert_LoadDynamicStressers EntryPoint inside the DLL allows to set the dynamic stressers */

[DllImport(@"DLL/MLBert_API.dll", EntryPoint = "APILoadDynamicStressers")]
private static extern int mlBert_LoadDynamicStressers (mlbertapi* instance, int *Data);

/*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Param int *Data: array of stressers data
* return int
* Allows setting the dynamic stressers configuration
* Returns 1 on success, 0 on failure.
*/

public int LoadDynamicStressers (mlbertapi* instance, int *Data)
 {
  return mlBert_LoadDynamicStressers (instance, *Data);
 }
```

mlBert_EnableDynamicStressers

This API call is used to enable the Dynamic stressers

API returns 1 upon success, 0 if failure.

Used for ML4009-EQL and ML4009-JIT-EQL only.

Example of Use:

```
/*
* mlBert_EnableDynamicStressers EntryPoint inside the DLL allows to enable the dynamic stressers
configuration*/

[DllImport(@"DLL/MLBert_API.dll", EntryPoint = " mlBert_EnableDynamicStressers ")]
private static extern int mlBert_EnableDynamicStressers (mlbertapi* instance, int channel, int timer, int
Interpolator_Step, int enable);

/*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Param int channel
* Param int timer: time for each stresser
* Param int Interpolator_Step: the interpolation step makes the tap values change smoothly from 1
stressor file to another, this will allow the DUT to adapt to the incoming signal with any problems.
Example: to go from one stressor (taps: 0,10,4,...) to another stressor (taps: 0,20,14,...) with
interpolation steps = 1, the EQL will pass by all the values in between (0,10,4)-->(0,11,5,...)-->(0,12,6,...)--
> (0,20,14) for interpolation step = 2 it will become (0,10,4)-->(0,12,6,...)-->(0,14,8,...)--> (0,20,14)
* Param int enable: 1 to enable it and 0 to disable it
* return int
* Allows enabling the dynamic stressers
* Returns 1 on success, 0 on failure.
*/

public int EnableDynamicStressers (mlbertapi* instance, int channel, int timer, int Interpolator_Step, int
enable)
 {
   return mlBert_EnableDynamicStressers (instance, channel, timer, Interpolator_Step, enable);
 }
```

mlBert_WriteEqualizerRegister

This API call is used to write into an EQL tab

API returns 1 upon success, 0 if failure.

Used for EQL BERTs only.

Example of Use:

```
/*
* mlBert_WritEqualizerRegister EntryPoint inside the DLL allows to write into an EQL tab*/

[DllImport(@"DLL/MLBert_API.dll", EntryPoint = " mlBert_WriteEqualizerRegister ")]
private static extern int mlBert_WriteEqualizerRegister (mlbertapi* instance, int channel, uint16
Register, uint16 data);

/*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Param int channel
* Param int Register
* Param int data
* return int
* Allows writing into an EQL tab
* Returns 1 on success, 0 on failure.
*/

public int WriteEqualizerRegister (mlbertapi* instance, int channel, uint16 Register, uint16 data)
 {
  return mlBert_WriteEqualizerRegister (instance, channel, Register, data);
 }
```

mlBert_ReadEqualizerRegister

This API call is used to read from an EQL tab

API returns 1 upon success, 0 if failure.

Used for EQL BERTs only.

Example of Use:

```
/*
* mlBert_ReadEqualizerRegister EntryPoint inside the DLL allows to read from an EQL tab*/

[DllImport(@"DLL/MLBert_API.dll", EntryPoint = " mlBert_ReadEqualizerRegister ")]
private static extern int mlBert_ReadEqualizerRegister (mlbertapi* instance, int channel, uint16 Register,
uint16 *data);

/*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Param int channel
* Param int Register
* Param int data: data extracted
* return int
* Allows reading from an EQL tab
* Returns 1 on success, 0 on failure.
*/

public int ReadEqualizerRegister (mlbertapi* instance, int channel, uint16 Register, uint16 *data)
 {
   return mlBert_ReadEqualizerRegister (instance, channel, Register, *data);
 }
```

mlBert_GetBathTubparallel

This API call is used to read the Bathtub for one or 4 channels in parallel mode.

Each channel should have a separate call.

API returns 1 upon success, 0 if failure.

Used for ML4039B-EQL and ML4039B only.

Example of use:

```
/*
* mlBert_GetBathTubparallel EntryPoint inside the DLL allows to read the Bathtub for one or 4 channels
in parallel mode
*/

 [DllImport(@"DLL/MLBert_API.dll", EntryPoint = " mlBert_GetBathTubparallel ")]
private static extern int mlBert_GetBathTubparallel (mlbertapi* instance, int []channel, double[]
xValues, double[] berValues);

/*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Param int channel: is an array of 4 values (1 or 0). For ex. If the user wants to read the bathtub for
channel 1 and 2, the param channel will be 1100
* Param xValues: return the xValues for each bathtub in an array of 64 values each
* Param BERValues: return the BERValues for each bathtub in an array of 64 values each
* Return int
* Returns 1 on success, 0 on failure.
* Channels are identified using the channel parameter
*/

public int GetBathTubparallel(mlbertapi* instance, int []channel, double[] xValues, double[] berValues)
 {
return mlBert_GetBathTubparallel (instance, channel, xValues, berValues);
}
```

Implementation:

```
PointPairList[] Data = new PointPairList[4];

double[] xValues = new double[1024];

double[] berValues = new double[1024];

apiGetBathTubparallel(classInstance, channel, xValues, berValues);

for (int j = 0; j < 4; j++)

{

 if (channel[j] == 1)

  {

     for (int i = 0; i < 64; i++)

     {

        Data[j].Add(xValues[(j * 64) + i], berValues[(j * 64) + i]);

     }

  }

}
```

mlBert_GetEyeparallel

This API call is used to read the Eye for one or 4 channels in parallel mode. It returns the matrix with xValues, yValues and BERValues for every specific location.

Each channel should have a separate call.

API returns 1 upon success, 0 if failure.

Used for ML4039B-EQL and ML4039B only.

Example of use:

```
/*
* mlBert_GetEyeparallel EntryPoint inside the DLL allows to read the Eye for one or 4 channels in
parallel mode
*/

 [DllImport(@"DLL/MLBert_API.dll", EntryPoint = " mlBert_GetEyeparallel ")]
private static extern int mlBert_GetEyeparallel (mlbertapi* instance, int[] Channel, int PacketNumber,
double[] xValues, double[] yValues, double[] berValues);

/*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Param int channel: is an array of 4 values (1 or 0). For ex. If the user wants to read the bathtub for
channel 1 and 2, the param channel will be 1100
* Param int PacketNumber: the user sends each time from 0 to 64 packets
* Param xValues: X location in the matrix
* Param yValues: Y location in the matrix
* Param BERValues: BERValues on this specific X and Y locations
* Return int
* Returns 1 on success, 0 on failure.
* Channels are identified using the channel parameter
*/

public int GetEyeparallel(mlbertapi* instance, int []channel, double[] xValues, double[] berValues)
 {
return mlBert_GetEyeparallel (instance, channel, xValues, berValues)

}
```

Implementation:

```
PointPairList []Parallelpoints = new PointPairList[4];

public class EyeData {

    public double[] xvalues;

    public double[] yvalues;

    public double[] berValues;

    public EyeData()

    {

      xvalues = new double[16384];

      yvalues = new double[16384];

      berValues = new double[16384];

    }

  }

     for (int i = 0; i < 64; i++)

     {

      GetEyeParallel(channel, i, ref eyeData);

      for (int k = 0; k < 4; k++)

      {

        if (channel[k] == 1)

          {

            pointer = Parallelpoints[k].Count();

            for (int j = 0; j < 256; j++)

             {

               Parallelpoints[k].Add(eyeData[k].xvalues[i + (j * 64)], eyeData[k].yvalues[i + (j * 64)],
(eyeData[k].berValues[i + (j * 64)] < 0.08 || eyeData[k].berValues[i + (j * 64)] == 1) ?
eyeData[k].berValues[i + (j * 64)] : 0);

               sharedVariables.Parallel_eyeValuesAcqumulated[k][counter[k]] =
eyeData[k].berValues[i + (j * 64)];

               counter[k]++;
```

```
                    }

                  sharedVariables.graphManager.DrawEyeParallel(k, Parallelpoints[k], pointer,
0);//sharedVariables.ShiftingToZeroValue[sharedVariables.eyeBathtubChannel]);

                }

            }

            // draw each packet

          }

      public void GetEyeParallel(int[] Channel, int PacketNumber, ref EyeData[] ChannelData)

      {

          apiGetEyeparallel(classInstance, Channel, PacketNumber, Parallel_xValues, Parallel_yValues,
Parallel_berValues);

          // each time we need to filtering the data that belong to the specific packet number


          for (int i = 0; i < 4; i++)

          {

              if (Channel[i] == 1)

              {

                  int pointer = (i * 16384) + PacketNumber;


                  for (int j = 0; j < 256; j++)

                  {

                      ChannelData[i].xvalues[PacketNumber + (j * 64)] = Parallel_xValues[pointer + (j * 64)];

                      ChannelData[i].yvalues[PacketNumber + (j * 64)] = Parallel_yValues[pointer + (j * 64)];

                      ChannelData[i].berValues[PacketNumber + (j * 64)] = Parallel_berValues[pointer + (j * 64)];

                  }

              }

          }

      }
```

mlBert_Check39B_EQLHVRevision

This API call is used to check the ML4039B-EQL board type.

API returns 1 upon success, 0 if failure.

Used for ML4039B-EQL and ML4039B only.

Example of Use:

```
/*
* mlBert_Check39B_EQLHVRevision EntryPoint inside the DLL allow to check the ML4039-B-EQL board
type*/

[DllImport(@"DLL/MLBert_API.dll", EntryPoint = " mlBert_Check39B_EQLHVRevision ")]
private static extern int mlBert_Check39B_EQLHVRevision (mlbertapi* instance, ref int Revision);

/*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Param int Revision: contains the revision number, 0: for no HV, 1: for HV
* return int
* Allows checking the ML4039B-EQL board type
* Returns 1 on success, 0 on failure.
*/

public int Check39B_EQLHVRevision (mlbertapi* instance, ref int Revision)
 {
   return mlBert_Check39B_EQLHVRevision (instance, ref Revision);
 }
```

mlBert_Check4039Jit_Revision

This API call is used to check the ML4039Jit revision and if it contains BUJ option.

API returns 1 upon success, 0 if failure.

Used for ML4039JIT and ML4039JIT-BUJ.

Example of Use:

```
/*
* mlBert_Check4039Jit_Revision EntryPoint inside the DLL allow to check the ML4039Jit revision and if it
contains BUJ option */

[DllImport(@"DLL/MLBert_API.dll", EntryPoint = " mlBert_Check4039Jit_Revision ")]
private static extern int mlBert_Check4039Jit_Revision (mlbertapi* instance, int *Revision, int
*WithBUJ);

/*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Param int Revision: contains the revision number, 0= revB or 1=revC
* Param int WithBUJ: 1= withBUJ, 0=withoutBUJ

*return int
* Allows checking the ML4039Jit revision and if it contains BUJ option
* Returns 1 on success, 0 on failure.
*/

public int Check4039Jit_Revision (mlbertapi* instance, int *Revision, int *WithBUJ)
 {
   return mlBert_Check4039Jit_Revision (instance, Revision, WithBUJ);
 }
```

mlBert_CalculateVerticalBathtubDualDirac

This API call is used to apply Dual Dirac on the vertical bathtub.

This function required using APIGetVBathtub, the X and Y values for the vertical Bathtub extracted from the APIGetVBathtub should be used as parameters in this function (second and third parameter).
After passing this function, the user should use these two parameters to draw the vertical BathTub in Dual Dirac mode.

Returns 1 on success, 0 on failure.

Used for all BERTs.

Example of use:

```
/*
* mlBert_CalculateVerticalBathtubDualDirac EntryPoint inside the DLL apply Dual Dirac on the vertical
bathtub.
*/

 [DllImport(@"DLL/MLBert_API.dll", EntryPoint = " mlBert_CalculateVerticalBathtubDualDirac ")]
private static extern int mlBert_CalculateVerticalBathtubDualDirac (mlbertapi* instance, int channel,
double[] xValues, double[] BERValues);

/*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Param int channel
* Param double[] xValues
* Param double[] BERValues
* Return int
* Allows to apply Dual Dirac on the vertical bathtub
* Returns 1 on success, 0 on failure.
* Channels are identified using the channel parameter
*/

Public int CalculateVerticalBathtubDualDirac (mlbertapi* instance, int channel, double *xValues, double
*BERValues)

{
return mlBert_CalculateVerticalBathtubDualDirac (instance, channel, xValues, BERValues);
}
```

mlBert_ReadVerticalRJDJ

This API call is used to read the vertical DJ and RJ for a specific channel. The user should use this function after using the dual dirac function.

Returns true on success, false on failure.

Used for ML4009-JIT and ML4039-JIT.

Example of use:

```
/*
* mlBert_ReadVerticalRJDJ EntryPoint inside the DLL allows to read the vertical DJ and RJ
*/

 [DllImport(@"DLL/MLBert_API.dll", EntryPoint = " mlBert_ReadVerticalRJDJ ")]
private static extern bool mlBert_ReadVerticalRJDJ (mlbertapi* instance, int channel, double ref
DJ_array, double ref RJ_array);

/*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Param int channel
* Param DJ array for the 4 channels
* Param RJ array for the 4 channels
* Return bool
* Allows to read the vertical DJ and RJ
* Returns true on success, false on failure.
* Channels are identified using the channel parameter
*/

Public bool ReadVerticalRJDJ (mlbertapi* instance, int channel, double *DJ_array, double *RJ_array)

{
return mlBert_ReadVerticalRJDJ (instance, channel, DJ_array, RJ_array);
}
```

mlBert_CalculateVerticalJitter

This API call is used to calculate the vertical jitter from current BathTub.

Returns 1 on success, 0 on failure.

Used for ML4009-JIT and ML4039-JIT.

Example of use:

```
/*
* mlBert_CalculateVerticalJitter EntryPoint inside the DLL allows calculating the vertical jitter from
current BathTub*/

 [DllImport(@"DLL/MLBert_API.dll", EntryPoint = " mlBert_CalculateVerticalJitter ")]
private static extern int mlBert_CalculateVerticalJitter (mlbertapi* instance, int channel, double
targetBER, double ref jitterMeasurements);

/*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Param int channel
* Param double targetBER
* Param double jitterMeasurements
* Return int
* Allows to calculate the vertical jitter from current BathTub
* Returns 1 on success, 0 on failure
* Channels are identified using the channel parameter
*/

Public int CalculateVerticalJitter (mlbertapi* instance, int channel, double targetBER,
double*jitterMeasurements)

{
return mlBert_CalculateVerticalJitter (instance, channel, targetBER, jitterMeasurements);

}
```

mlBert_SkewCalibration

This API call is used to convert skew steps to UI and vice versa (depending on mode).

Returns the equivalent steps in UI.

Used for ML4009-JIT and ML4039-JIT.

Example of use:

```
/*
* mlBert_SkewCalibration EntryPoint inside the DLL allows converting skew steps to UI*/

 [DllImport(@"DLL/MLBert_API.dll", EntryPoint = " mlBert_SkewCalibration ")]
private static extern double mlBert_SkewCalibration (mlbertapi* instance, int channel, int mode, double
Steps);

/*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Param int channel
* Param int mode: 0 to convert steps to UI(returns the steps equivalent value in UI), 1 to convert UI to
steps(returns the UI equivalent value in steps)
* Param double steps: step or UI value
* Return double
* Allows to convert skew steps to UI
* Returns the equivalent steps in UI
* Channels are identified using the channel parameter
*/

Public double SkewCalibration (mlbertapi* instance, int channel, int mode, double Steps)

{
return mlBert_SkewCalibration (instance, channel, mode, Steps);

}
```

mlBert_ML4039BEQLAmplitudeCalibration_Step_To_MV

This API call is used to convert skew steps to MV.

Returns the equivalent steps in MV.

Used for ML4039B-EQL and ML4039B.

Example of use:

```
/*
* mlBert_ML4039BEQLAmplitudeCalibration_Step_T0_MV EntryPoint inside the DLL allows converting
skew steps to MV*/

 [DllImport(@"DLL/MLBert_API.dll", EntryPoint = "
mlBert_ML4039BEQLAmplitudeCalibration_Step_T0_MV ")]
private static extern double mlBert_ML4039BEQLAmplitudeCalibration_Step_T0_MV (mlbertapi*
instance, int channel, int step, double LineRate);

/*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Param int channel
* Param int step: step value
* Param double LineRate: linerate value for the channel
* Return double
* Allows to convert skew steps to MV
* Returns the equivalent steps in MV
* Channels are identified using the channel parameter
*/

Public double ML4039BEQLAmplitudeCalibration_Step_To_MV (mlbertapi* instance, int channel, int
step, double LineRate)

{
return mlBert_ML4039BEQLAmplitudeCalibration_Step_T0_MV (instance, channel, step, LineRate);

}
```

mlBert_ML4039BEQLAmplitudeCalibration_MV_To_Step

This API call is used to convert MV to skew steps.

Returns the equivalent MV in steps.

Used for ML4039B-EQL and ML4039B.

Example of use:

```
/*
* mlBert_ML4039BEQLAmplitudeCalibration_MV_T0_Step EntryPoint inside the DLL allows converting
MV to skew steps*/

 [DllImport(@"DLL/MLBert_API.dll", EntryPoint = "
mlBert_ML4039BEQLAmplitudeCalibration_MV_T0_Step ")]
private static extern double mlBert_ML4039BEQLAmplitudeCalibration_MV_T0_Step (mlbertapi*
instance, int channel, int MV, double LineRate);

/*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Param int channel
* Param int MV: MV value
* Param double LineRate: linerate value for the channel
* Return double
* Allows to convert MV to steps
* Returns the equivalent MV in steps
* Channels are identified using the channel parameter
*/

Public double ML4039BEQLAmplitudeCalibration_ MV _To_Step (mlbertapi* instance, int channel, int
MV, double LineRate)

{
return mlBert_ML4039BEQLAmplitudeCalibration_MV_T0_Step (instance, channel, MV, LineRate);

}
```

mlBert_ML4039BEQLAmplitudeHVCalibration_Step_To_MV

This API call is used to convert skew steps to MV for high voltage.

Returns the equivalent steps in MV.

Used for ML4039B-EQL-HV.

Example of use:

```
/*
* mlBert_ML4039BEQLAmplitudeHVCalibration_Step_T0_MV EntryPoint inside the DLL allows
converting skew steps to MV for high voltage*/

 [DllImport(@"DLL/MLBert_API.dll", EntryPoint = "
mlBert_ML4039BEQLAmplitudeHVCalibration_Step_T0_MV ")]
private static extern double mlBert_ML4039BEQLAmplitudeHVCalibration_Step_T0_MV (mlbertapi*
instance, int channel, int step, double LineRate);

/*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Param int channel
* Param int step: step value
* Param double LineRate: linerate value for the channel
* Return double
* Allows to convert skew steps to MV
* Returns the equivalent steps in MV
* Channels are identified using the channel parameter
*/

Public double ML4039BEQLAmplitudeHVCalibration_Step_To_MV (mlbertapi* instance, int channel, int
step, double LineRate)

{
return mlBert_ML4039BEQLAmplitudeHVCalibration_Step_T0_MV (instance, channel, step, LineRate);

}
```

mlBert_ML4039BEQLAmplitudeHVCalibration_MV_To_Step

This API call is used to convert MV to skew steps for high voltage.

Returns the equivalent MV in steps.

Used for ML4039B-EQL-HV.

Example of use:

```
/*
* mlBert_ML4039BEQLAmplitudeHVCalibration_MV_T0_Step EntryPoint inside the DLL allows
converting MV to skew steps for high voltage*/

 [DllImport(@"DLL/MLBert_API.dll", EntryPoint = "
mlBert_ML4039BEQLAmplitudeHVCalibration_MV_T0_Step ")]
private static extern double mlBert_ML4039BEQLAmplitudeHVCalibration_MV_T0_Step (mlbertapi*
instance, int channel, int MV, double LineRate);

/*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Param int channel
* Param int MV: MV value
* Param double LineRate: linerate value for the channel
* Return double
* Allows to convert MV to steps
* Returns the equivalent MV in steps
* Channels are identified using the channel parameter
*/

Public double ML4039BEQLAmplitudeHVCalibration_ MV _To_Step (mlbertapi* instance, int channel, int
MV, double LineRate)

{
return mlBert_ML4039BEQLAmplitudeHVCalibration_MV_T0_Step (instance, channel, MV, LineRate);

}
```

mlBert_CalculateDeep_EyeWidth

This API call is used to calculate eye width at a certain BER level.

It requires running GetBathTub first and dual dirac calulation.

Returns 1 on success, 0 on failure.

Used for all BERTs except PAM.

Example of use:

```
/*
* mlBert_CalculateDeep_EyeWidth EntryPoint inside the DLL allows calculating the eye width at a
certain BER level*/

 [DllImport(@"DLL/MLBert_API.dll", EntryPoint = " mlBert_CalculateDeep_EyeWidth ")]
private static extern int mlBert_CalculateDeep_EyeWidth (mlbertapi* instance, int channel, double
targetBER, ref double Eyewidth);

/*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Param int channel
* Param double targetBER: defined by the user
* Param double Eyewidth
* Return int
* Allows to calculate the eye width at a certain BER level
* Returns 1 on success, 0 on failure
* Channels are identified using the channel parameter
*/

Public int CalculateDeep_EyeWidth (mlbertapi* instance, int channel, double targetBER, ref double
Eyewidth)

{
return mlBert_CalculateDeep_EyeWidth (instance, channel, targetBER, Eyewidth);

}
```

mlBert_CalculateDeep_EyeHeight

This API call is used to calculate eye height at a certain BER level.

It requires running GetVBathTub first dual dirac calulation.

Returns 1 on success, 0 on failure.

Used for all BERTs except PAM.

Example of use:

```
/*
* mlBert_CalculateDeep_eyeHeight EntryPoint inside the DLL allows calculating the eye height at a
certain BER level*/

 [DllImport(@"DLL/MLBert_API.dll", EntryPoint = " mlBert_CalculateDeep_eyeHeight ")]
private static extern int mlBert_CalculateDeep_eyeHeight (mlbertapi* instance, int channel, double
targetBER, ref double Eyeheight);

/*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Param int channel
* Param double targetBER: defined by the user
* Param double Eyeheight
* Return int
* Allows to calculate the eye height at a certain BER level
* Returns 1 on success, 0 on failure
* Channels are identified using the channel parameter
*/

Public int CalculateDeep_EyeHeight (mlbertapi* instance, int channel, double targetBER, ref double
Eyeheight)

{
return mlBert_CalculateDeep_eyeHeight (instance, channel, targetBER, Eyeheight);

}
```

mlBert_BathTubVerticalOffset_mv

This API call is used to set vertical offset for instant Bathtub

Returns 1 on success, 0 on failure.

Used for all BERTs except PAM.

Example of use:

```
/*
mlBert_BathTubVerticalOffset_mv EntryPoint inside the DLL allows setting vertical offset for instant
bathtub */

 [DllImport(@"DLL/MLBert_API.dll", EntryPoint = "APIBathTubVerticlOffset_mv ")]
private static extern int APIBathTubVerticlOffset_mv (mlbertapi* instance, int channel, byte
verticaloffset);

/*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Param int channel
* Param byte vertical offset in mv
* Return int
* Allows to set vertical offset for instant bathtub
* Returns 1 on success, 0 on failure
* Channels are identified using the channel parameter
*/

Public int BathTubVerticlOffset_mv (mlbertapi* instance, int channel, byte verticaloffset)

{
return APIBathTubVerticlOffset_mv (instance, channel, verticaloffset);



}
```

mlBert_ChangeBERPhaseAndOffset_pS_mV

This API call is used to change the phase and amplitude decision point

Returns 1 on success, 0 on failure.

Used for all BERTs.

Example of use:

```
/*
* mlBert_ChangeBERPhaseAndOffset_pS_mV EntryPoint inside the DLL allows changing the phase and
amplitude decision point */

 [DllImport(@"DLL/MLBert_API.dll", EntryPoint = " mlBert_ChangeBERPhaseAndOffset_pS_mV ")]
private static extern int mlBert_ChangeBERPhaseAndOffset_pS_mV (mlbertapi* instance, int channel,
int phase, int amplitude);

/*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Param int channel
* Param int phase: phase value in ps
* Param int amplitude: amplitude level in mV
* Return int
* Allows to change the phase and amplitude decision point
* Returns 1 on success, 0 on failure
* Channels are identified using the channel parameter
*/

Public int ChangeBERPhaseAndOffset_pS_mv (mlbertapi* instance, int channel, int phase, int amplitude)

{
return mlBert_ChangeBERPhaseAndOffset_pS_mV (instance, channel, phase, amplitude);



}
```

mlBert_ClockOut39A

This API call is used to specify clock out for ML4039A-JIT.

Returns 1 on success, 0 on failure.

Used for ML4039A-JIT.

Example of use:

```
/*
* mlBert_ClockOut39A EntryPoint inside the DLL allows specifying clock out for ML4039A-JIT */

 [DllImport (@"DLL/MLBert_API.dll", EntryPoint = " mlBert_ClockOut39A ")]
private static extern int mlBert_ClockOut39A (mlbertapi* instance, int TriggerOut, int CDRchannel, int
CDRRate);

/*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Param int TriggerOut: 0=reference clock, 1= reference clock/2, 2=CDR.
* Param int CDRchannel: 0= CDR channel 1 output (if TriggerOut = 2 channel 0 from clean chip and if 1
channel 0 from jit chip), 1= CDR channel 2 output (if TriggerOut = 2 channel 1 from clean chip and if 1
channel 1 from jit chip), 2= CDR channel 3 output (if TriggerOut = 2 channel 2 from clean chip and if 1
channel 2 from jit chip), 3= CDR channel 4 output (if TriggerOut = 2 channel 3 from clean chip and if 1
channel 3 from jit chip)
* Param int CDRRate: 0=Datarate/8, 1=Datarate/16, 2= Datarate/32, 3= Datarate/64, 4=Datarate/128,
5= Datarate/256.
* Return int
* Allows to specify clock out for ML4039A-JIT
* Returns 1 on success, 0 on failure
*/

Public int ClockOut39A (mlbertapi* instance, int TriggerOut, int CDRchannel, int CDRRate)

{
return int mlBert_ClockOut39A (instance, TriggerOut, CDRchannel, CDRRate);

}
```

mlBert_Check39A_CleanJitRevision

This API call is used to check the ML4039A and ML4039A-JIT revision number.

Returns 1 on success, 0 on failure.

Used for ML4039A and ML4039A-JIT.

Example of use:

```
/*
* mlBert_Check39A_CleanJitRevision EntryPoint inside the DLL allows checking the ML4039A rev
number*/

 [DllImport (@"DLL/MLBert_API.dll", EntryPoint = " mlBert_Check39A_CleanJitRevision ")]
private static extern mlBert_Check39A_CleanJitRevision (mlbertapi* instance, int *Revision);

/*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Param int Revision: contains the revision number; 0 for clean, 1 for Jit revision, 2 for clean and jit
* Return int
* Allows to checking the ML4039A revision number
* Returns 1 on success, 0 on failure
*/

Public int Check39A_CleanJitRevision (mlbertapi* instance, int *Revision)

{
return int mlBert_Check39A_CleanJitRevision (instance, Revision);

}
```

mlBert_OptimizedSJ

This API call is used to enable the SJ optimization for a specific channel.
Returns calibration version.

Used for ML4009-JIT, ML4039-JIT, ML4039A-JIT and ML4004-JIT.

Example of use:

```
/*
* mlBert_OptimizedSJ EntryPoint inside the DLL allows enabling the SJ optimization for a specific
channel*/

 [DllImport (@"DLL/MLBert_API.dll", EntryPoint = " mlBert_OptimizedSJ ")]
private static extern int mlBert_OptimizedSJ (mlbertapi* instance, int channel, double *Version, double
*MinMax);

/*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Param int channel: channel index
* Param double version: calibration version number
* Param double MinMax: array that contains the min and max for each SJ frequency {5 MHz, 10 MHz, 20
MHz, 40 MHz, 80 MHz}
* Return int
* Allows to enable the SJ optimization for a specific channel
* Returns 1 on calibrated (recently calibration value=1), 0 on not calibrated
*/

Public int OptimizedSJ (mlbertapi* instance, int channel, double *Version, double *MinMax)

{
return int mlBert_OptimizedSJ (instance, channel, Version, MinMax);

}
```

mlBert_UnOptimizedSJ

This API call is used to disable the SJ optimization for a specific channel.
Returns 1 on success, 0 on failure.

Used for ML4009-JIT, ML4039-JIT, ML4039A-JIT and ML4004-JIT.

Example of use:

```
/*
* mlBert_UnOptimizedSJ EntryPoint inside the DLL allows disabling the SJ optimization for a specific
channel*/

 [DllImport (@"DLL/MLBert_API.dll", EntryPoint = " mlBert_UnOptimizedSJ ")]
private static extern int mlBert_UnOptimizedSJ (mlbertapi* instance, int channel);

/*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Param int channel: channel index
* Return int
* Allows to disable the SJ optimization for a specific channel
* Returns 1 on success, 0 on failure
*/

Public int UnOptimizedSJ (mlbertapi* instance, int channel)

{
return int mlBert_UnOptimizedSJ (instance, channel);

}
```

mlBert_ReadSerialNumber

This API call is used to read board serial number.
Returns 1 on success, 0 on failure.

Used for all BERTs.

Example of use:

```
/*
* mlBert_ReadSerialNumber EntryPoint inside the DLL allows reading board SN*/

 [DllImport (@"DLL/MLBert_API.dll", EntryPoint = " mlBert_ReadSerialNumber ")]
private static extern int mlBert_ReadSerialNumber (mlbertapi* instance,UInt64 *SN, int
*SerialNumberVersion);

/*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Param UInt64 SN: the serial number
* Param int SerialNumberVersion: =0 represents the old serial number system, =1 represents the new
SN version
* Return int
* Allows to read the board SN
* Returns 1 on success, 0 on failure
*/

Public int ReadSerialNumber (mlbertapi* instance, UInt64 *SN, int *SerialNumberVersion)

{
return int mlBert_ReadSerialNumber (instance, SN, SerialNumberVersion);

}
```

mlBert_LineRateConfigurationBert39Brv2

This API call is used to set the Line Rate of the ML4039B No EQL. It returns 1 if successful, 0 if a problem occurs while setting the line Rate.

Used for ML4039B no EQL only.

Example of use:

```
/*
 * mlBert_LineRateConfigurationBert39Brv2 EntryPoint inside the DLL is responsible to set the Line Rate
*/
[DllImport(@"DLL/MLBert_API.dll", EntryPoint = " mlBert_LineRateConfigurationBert39Brv2")]
private static extern int mlBert_LineRateConfigurationBert39Brv2 (mlbertapi* instance, int
channelIndex, double linerate1, double linerate2, int clockSource, int clockSource_ConfigIndex);
/*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Param int channelIndex : if 0 the linerate1 and linerate2 refers to channel1 and channel2, if 1 they
refers to channel3 and channel4
* Param double lineRate1 : contains the linerate (for channel1 if channelindex=0, channel3 if
channelindex=1) required in Gbps ranging from 1-15 and 15-33
* Param double lineRate2 : contains the linerate (for channel2 if channelindex=0, channel4 if
channelindex=1) required in Gbps ranging from 1-15 and 15-33
*Param int clockSource: External=0, Internal=1
NB: -The clock file should be included in the specified directory in ConfigureApplication:saveConfig
parameter (default:clk)
-Configuration will be lost after applying linerate, and can be restored automatically using
RestoreAllConfig() function
*Param int clockSource_ConfigIndex: When connecting the ML407 as an external clock to ML4039B to
work on external clock mode, the user needs to specify the configuration index. Note that this feature is
a GUI feature (probably a purely dll in the future). The user can choose the corresponding ML407
configuration such as the VCO frequency and the divider, that will provide the same linerate chosen
after some calculations made by the GUI. At this mode the 4 channels of ML4039B will have the same
linerate, and the Avago chip will work on specific divider. (To get more info about this feature, please
contact support@multilaneinc.com).
* return int
* Allows to set Line Rate generated by the ML4039B No EQL
*/

public int LineRateConfigurationBert39Brv2 (mlbertapi* instance, int channelIndex, double linerate1,
double linerate2, int clockSource, int clockSource_ConfigIndex)
 {
return mlBert_LineRateConfigurationBert39Brv2 (instance, channelIndex, linerate1, linerate2,
clockSource, clockSource_ConfigIndex);
 }
```

mlBert_TXClockOut_RateOverEight

This API call is used to specify clock out equal to line rate over eight.
Returns 1 on success, 0 on failure.

Used for ML4039, ML4039-JIT, ML4009, ML4009-JIT, ML4009-EQL and ML4039A.

Example of use:

```
/*
* mlBert_TXClockOut_RateOverEight EntryPoint inside the DLL allows specifying clock out equal to line
rate over eight*/

 [DllImport (@"DLL/MLBert_API.dll", EntryPoint = " mlBert_TXClockOut_RateOverEight ")]
private static extern int mlBert_TXClockOut_RateOverEight (mlbertapi* instance);

/*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Return int
* Allows specifying clock out equal to line rate over eight
* Returns 1 on success, 0 on failure
*/

Public int TXClockOut_RateOverEight (mlbertapi* instance)

{
return int mlBert_TXClockOut_RateOverEight (instance);

}
```

mlBert_Check39Brv2_CleanJitRevision

This API call is used to check the Ml4039BNoEQL revision type.
Returns 1 on success, 0 on failure.

Used for ML4039B No EQL.

Example of use:

```
/*
* mlBert_Check39Brv2_CleanJitRevision EntryPoint inside the DLL allows checking the Ml4039BNoEQL
revision type */

 [DllImport (@"DLL/MLBert_API.dll", EntryPoint = " mlBert_Check39Brv2_CleanJitRevision")]
private static extern int mlBert_Check39Brv2_CleanJitRevision (mlbertapi* instance, int *Revision);

/*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Param int Revision
* Return int
* Allows checking the Ml4039BNoEQL revision type
* Returns 1 on success, 0 on failure
*/

Public int Check39Brv2_CleanJitRevision (mlbertapi* instance, int *Revision)

{
return int mlBert_Check39Brv2_CleanJitRevision (instance, *Revision);

}
```

mlBert_CheckTDA_Revision

This API call is used to check the TDA revision number.
Returns 1 on success, 0 on failure.

Used for ML4004, ML4004-JIT and ML4004-PAM only.

Example of use:

```
/*
* mlBert_CheckTDA_Revision EntryPoint inside the DLL allows checking the TDA revision number */

 [DllImport (@"DLL/MLBert_API.dll", EntryPoint = " mlBert_CheckTDA_Revision ")]
private static extern int mlBert_CheckTDA_Revision (mlbertapi* instance, int Revision, int BoardRV, int
*IsPortable);

/*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Param int Revision: contains the Revision number, 0- for no HV or jit, 1 for HV version, 2 for jit revision
and 3 for JIT and HV revision
* Param int BoardRV: contains the Board Revision 1 Rev B and 0 Rev A
* Param int IsPortable : if true that means this is a Portable version
* Return int
* Allows checking the TDA revision number
* Returns 1 on success, 0 on failure
*/

Public int CheckTDA_Revision (mlbertapi* instance, int Revision, int BoardRV, int *IsPortable)

{
return int mlBert_CheckTDA_Revision (instance, Revision, BoardRV, *IsPortable );

}
```

mlBert_ClockOutTDArevB

This API call is used to specify clock input and output for TDA rev B.
Returns 1 on success, 0 on failure.

Used for ML4004, ML4004-JIT and ML4004-PAM only.

Example of use:

```
/*
* mlBert_ClockOutTDArevB EntryPoint inside the DLL allows specifying clock input and output for TDA
rev B */

 [DllImport (@"DLL/MLBert_API.dll", EntryPoint = " mlBert_ClockOutTDArevB ")]
private static extern int mlBert_ClockOutTDArevB (mlbertapi* instance, int ClockInSource, int
ClockOutSource, int ClockOutDividers);

/*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Param int ClockInSource: 0 = InputClock silab source, 1 = InputClock FM Circuit
* Param int ClockOutSource: 1 = LPCDR, 0 = LPCDR TX Clock (LineRate/ ClockOutDividers), 2 = LPCDR RX
Clock (LineRate/ ClockOutDividers)

* Param ClockOutDividers: 0 = Datarate/8, 1 = Datarate/16, 2 = Datarate/32, 3 = Datarate/64, 4 =
Datarate/128, 5 = Datarate/256
* Return int
* Allows specifying clock input and output for TDA rev B
* Returns 1 on success, 0 on failure
*/

Public int ClockOutTDArevB (mlbertapi* instance, int ClockInSource, int ClockOutSource, int
ClockOutDividers)

{
return int mlBert_ClockOutTDArevB (instance, ClockInSource, ClockOutSource, ClockOutDividers);

}
```

mlBert_SetVCOFrequency

This API call is used to set VCO Frequency output.
Returns 1 on success, 0 on failure.

Used for TDA rev B only.

Example of use:

```
/*
* mlBert_SetVCOFrequency EntryPoint inside the DLL allows setting VCO Frequency output */

 [DllImport (@"DLL/MLBert_API.dll", EntryPoint = " mlBert_SetVCOFrequency ")]
private static extern int mlBert_SetVCOFrequency (mlbertapi* instance, double VCOFrequency);

/*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Param double VCOFrequency: The VCO frequency desired (range 2.43 GHz to 5.585 GHz)

* Return int
* Allows setting VCO Frequency output for TDA rev B
* Returns 1 on success, 0 on failure
*/

Public int SetVCOFrequency (mlbertapi* instance, double VCOFrequency)

{
return mlBert_SetVCOFrequency (instance, VCOFrequency);

}
```

mlBert_Enable_DisableFM

This API call is used to enable/disable FM modulation.
Returns 1 on success, 0 on failure.

Used for TDA rev B only.

Example of use:

```
/*
* mlBert_Enable_DisableFM EntryPoint inside the DLL allows enabling/disabling FM modulation */

 [DllImport (@"DLL/MLBert_API.dll", EntryPoint = " mlBert_Enable_DisableFM")]
private static extern int mlBert_Enable_DisableFM (mlbertapi* instance , int FM_Status);

/*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Param int FM_Status: 0 to disable and 1 to enable

* Return int
* Allows enabling/disabling FM modulation
* Returns 1 on success, 0 on failure
*/

Public int Enable_DisableFM (mlbertapi* instance, int FM_Status)

{
return mlBert_Enable_DisableFM (instance, FM_Status);

}
```

mlBert_SetFMFRequency

This API call is used to set FM Frequency.
Returns 1 on success, 0 on failure.

Used for TDA rev B only.

Example of use:

```
/*
* mlBert_SetFMFrequency EntryPoint inside the DLL allows setting FM frequency */

 [DllImport (@"DLL/MLBert_API.dll", EntryPoint = "mlBert_SetFMFrequency ")]
private static extern int mlBert_SetFMFrequency (mlbertapi* instance , double FM_Frequency);

/*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Param int FM_Frequency: FM Frequency Value (Uint in MHz)

* Return int
* Allows setting FM frequency
* Returns 1 on success, 0 on failure
*/

Public int SetFMFRequency (mlbertapi* instance, double FM_Frequency)

{
return mlBert_SetFMFrequency (instance, FM_Frequency);

}
```

mlBert_SetFMAmplitude

This API call is used to set FM Amplitude.
Returns 1 on success, 0 on failure.

Used for TDA rev B only.

Example of use:

```
/*
* mlBert_SetFMAmplitude EntryPoint inside the DLL allows setting FM Amplitude */

 [DllImport (@"DLL/MLBert_API.dll", EntryPoint = "mlBert_SetFMAmplitude ")]
private static extern int mlBert_SetFMAmplitude (mlbertapi* instance , int FM_Amplitdue);

/*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Param int FM_ Amplitdue: FM Amplitude Value (0 to 4095)

* Return int
* Allows setting FM Amplitude
* Returns 1 on success, 0 on failure
*/

Public int SetFMAmplitude (mlbertapi* instance, int FM_Amplitdue)

{
return mlBert_SetFMAmplitude (instance, FM_Amplitdue);

}
```

mlBert_GetEyeScale

This API call is used to read the Eye Y scale also it read the min and max y value in steps.
Returns 1 on success, 0 on failure.

Used for ML4039B only.

Example of use:

```
/*
* mlBert_GetEyeScale EntryPoint inside the DLL allows reading the min and max y value in steps */

 [DllImport (@"DLL/MLBert_API.dll", EntryPoint = "mlBert_GetEyeScale ")]
private static extern int mlBert_GetEyeScale (mlbertapi* instance, int Channel, double *Margin);

/*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Param int channel: the channel number
* Param double *Margin: conversion value
* Return int
* Allows reading the min and max y value in steps
* Returns 1 on success, 0 on failure
*/

Public int GetEyeScale (mlbertapi* instance, int Channel, double *Margin)

{
return mlBert_GetEyeScale (instance, Channel, *Margin);

}
```

mlBert_CheckBertJitisATE

This API call is used to check if the ML4039 jit is an ATE Board.
Returns 1 on success, 0 on failure.

Used for ML4039-JIT-ATE only.

Example of use:

```
/*
* mlBert_CheckBertJitisATE EntryPoint inside the DLL allows checking if the ML4039 jit is an ATE Board
*/

 [DllImport (@"DLL/MLBert_API.dll", EntryPoint = "mlBert_CheckBertJitisATE ")]
private static extern int mlBert_CheckBertJitisATE (mlbertapi* instance);

/*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Return int
* Allows checking if the ML4039 jit is an ATE Board
* Returns 1 on success, 0 on failure
*/

Public int CheckBertJitisATE (mlbertapi* instance)

{
return mlBert_CheckBertJitisATE (instance);

}
```

mlBert_GetPhaseShiftValue

This API call is used to read the Phase Shift for a specific channel.
Used for all BERTs.

Example of use:

```
/*
* mlBert_GetPhaseShiftValue EntryPoint inside the DLL allows reading the Phase Shift for a specific
channel */

 [DllImport (@"DLL/MLBert_API.dll", EntryPoint = "mlBert_GetPhaseShiftValue ")]
private static extern double mlBert_GetPhaseShiftValue (mlbertapi* instance, int Channel);

/*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Param byte channel: channel number
* Return double
* Allows reading the Phase Shift for a specific channel
*/

Public double GetPhaseShiftValue (mlbertapi* instance, int channel)

{
return mlBert_GetPhaseShiftValue (instance, channel);

}
```

mlBert_EnableFMFrequencyCalibration

This API call is used to Enable FM Calibration Value.

Used for ML4039B that includes ML407.

Example of use:

```
/*
* mlBert_EnableFMFrequencyCalibration EntryPoint inside the DLL allows enabling FM Calibration Value
*/

 [DllImport (@"DLL/MLBert_API.dll", EntryPoint = "mlBert_EnableFMFrequencyCalibration ")]
private static extern int mlBert_EnableFMFrequencyCalibration (mlbertapi* instance, double *MinMax);

 /*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Param MinMax: array that contain all the min and max value for each FM calibration frequency
* Returns if the command was executed properly Success = 1, failed = 0
* Allows enabling FM Calibration Value
*/

Public int EnableFMFrequencyCalibration (mlbertapi* instance, double *MinMax)

{
return mlBert_EnableFMFrequencyCalibration (instance, *MinMax);

}
```

mlBert_DisableFMFrequencyCalibration

This API call is used to Disable FM Calibration Value.

Used for ML4039B that includes ML407.

Example of use:

```
/*
* mlBert_DisableFMFrequencyCalibration EntryPoint inside the DLL allows disabling FM Calibration
Value */

 [DllImport (@"DLL/MLBert_API.dll", EntryPoint = "mlBert_DisableFMFrequencyCalibration ")]
private static extern int mlBert_DisableFMFrequencyCalibration (mlbertapi* instance);

 /*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Returns if the command was executed properly Success = 1, failed =0
* Allows disabling FM Calibration Value
*/

Public int DisableFMFrequencyCalibration (mlbertapi* instance)

{
return mlBert_DisableFMFrequencyCalibration (instance);

}
```

mlBert_ClockSource_LoadDefaultCalibration

This API call is used to load FM Calibration Value.

Used for ML4039B that includes ML407.

Example of use:

```
/*
* mlBert_ClockSource_LoadDefaultCalibration EntryPoint inside the DLL allows loading FM Calibration
Value */

 [DllImport (@"DLL/MLBert_API.dll", EntryPoint = "mlBert_ClockSource_LoadDefaultCalibration ")]
private static extern int mlBert_ClockSource_LoadDefaultCalibration (mlbertapi* instance);

 /*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Returns if the command was executed properly Success = 1, failed =0
* Allows loading FM Calibration Value
*/

Public int ClockSource_LoadDefaultCalibration (mlbertapi* instance)

{
return mlBert_ClockSource_LoadDefaultCalibration (instance);

}
```

mlBert_SaveFMCalibrationValues

This API call is used to save FM Calibration Values.

Used for ML4039JIT BUJ that includes ML407.

Example of use:

```
/*
* mlBert_SaveFMCalibrationValues EntryPoint inside the DLL allows disabling FM Calibration Values */

 [DllImport (@"DLL/MLBert_API.dll", EntryPoint = "mlBert_SaveFMCalibrationValues ")]
private static extern int mlBert_SaveFMCalibrationValues (mlbertapi* instance, double Version, double
*Data);

 /*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Param double Version: FM calibration version
*Param double *Data: Data to be saved

* Return int
* Allows saving FM calibration values
* Returns 1 on success, 0 on failure
*/

Public int SaveFMCalibrationValues (mlbertapi* instance, double Version, double *Data)

{
return mlBert_SaveFMCalibrationValues (instance, Version, Data);

}
```

mlBert_ReadVCOLock

This API call is used to read the VCO Lock.

Used for ML4039B that includes ML407.

Example of use:

```
/*
* mlBert_ReadVCOLock EntryPoint inside the DLL allows reading the VCO lock */

 [DllImport (@"DLL/MLBert_API.dll", EntryPoint = "mlBert_ReadVCOLock ")]
private static extern int mlBert_ReadVCOLock (mlbertapi* instance);

 /*
* Public Interface to connect with DLL
* Param mlbertapi* instance

* Return int
* Allows reading the VCO lock
* Returns 1 on success, 0 on failure
*/

Public int ReadVCOLock (mlbertapi* instance)

{
return mlBert_ReadVCOLock (instance);

}
```

mlBert_EnableBUJ

This API call is used to enable/disable BUJ.

Used for ML4039JIT BUJ that includes ML407.

Example of use:

```
/*
* mlBert_EnableBUJ EntryPoint inside the DLL allows enabling/disabling BUJ*/

 [DllImport (@"DLL/MLBert_API.dll", EntryPoint = "mlBert_EnableBUJ ")]
private static extern void mlBert_EnableBUJ (mlbertapi* instance , int channel, int status);

 /*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Param int channel: channel number
* Param int status: 1 for enable, 0 for disable

* Allows enabling/disabling BUJ
*/
Public void EnableBUJ (mlbertapi* instance, int channel, int status)

{
return mlBert_EnableBUJ (instance, channel, status);

}
```

mlBert_SetBUJfreq

This API call is used to set the BUJ frequency for a specific channel.

Used for ML4039JIT BUJ that includes ML407.

Example of use:

```
/*
* mlBert_SetBUJfreq EntryPoint inside the DLL allows setting the BUJ frequency for a specific channel*/

 [DllImport (@"DLL/MLBert_API.dll", EntryPoint = "mlBert_SetBUJfreq ")]
private static extern void mlBert_SetBUJfreq (mlbertapi* instance , int channel, int frequency);

 /*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Param int channel: channel number
* Param int frequency: 1-->5G 2 --> 2.5G 3 --> 1.25G 4 --> 625M

* Allows setting the BUJ frequency for a specific channel
*/
Public void SetBUJfreq (mlbertapi* instance, int channel, int frequency)

{
return mlBert_SetBUJfreq (instance, channel, frequency);

}
```

mlBert_SetBUJamplitude

This API call is used to set the BUJ amplitude for a specific channel.

Used for ML4039JIT BUJ that includes ML407.

Example of use:

```
/*
* mlBert_SetBUJamplitude EntryPoint inside the DLL allows setting the BUJ amplitude for a specific
channel*/

 [DllImport (@"DLL/MLBert_API.dll", EntryPoint = " mlBert_SetBUJamplitude ")]
private static extern void mlBert_SetBUJamplitude (mlbertapi* instance , int channel, int amplitude, int
index);

 /*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Param int channel: channel number
* Param int amplitude: 0 to 4095
* Param int index: index of the rate
* Allows setting the BUJ amplitude for a specific channel
*/
Public void SetBUJamplitude (mlbertapi* instance, int channel, int amplitude, int index)

{
return mlBert_SetBUJamplitude (instance, channel, amplitude, index);

}
```

mlBert_OptimizedBUJ

This API call is used to load the BUJ calibration for a specific channel.
Returns 1 on success, 0 on failure.

Used for ML4039JIT BUJ that includes ML407.

Example of use:

```
/*
* mlBert_OptimizedBUJ EntryPoint inside the DLL allows optimizing BUJ for a specific channel*/

 [DllImport (@"DLL/MLBert_API.dll", EntryPoint = “mlBert_OptimizedBUJ ")]
private static extern int mlBert_OptimizedBUJ (mlbertapi* instance, int Channel, double *MinMax);

 /*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Param int channel: channel number
* Param double MinMax: min and max of calibration values
* Allows optimizing BUJ for a specific channel
*/
Public int OptimizedBUJ (mlbertapi* instance, int channel, double *MinMax)

{
return mlBert_OptimizedBUJ (instance, channel, MinMax);

}
```

mlBert_UnOptimizedBUJ

This API call is used to remove the BUJ calibration for a specific channel.
Returns 1 on success, 0 on failure.

Used for ML4039JIT BUJ that includes ML407.

Example of use:

```
/*
* mlBert_UnOptimizedBUJ EntryPoint inside the DLL allows unoptimizing BUJ for a specific channel*/

 [DllImport (@"DLL/MLBert_API.dll", EntryPoint = “mlBert_UnOptimizedBUJ ")]
private static extern int mlBert_UnOptimizedBUJ (mlbertapi* instance, int Channel);

 /*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Param int channel: channel number
* Allows unoptimizing BUJ for a specific channel
*/
Public int UnOptimizedBUJ (mlbertapi* instance, int channel)

{
return mlBert_UnOptimizedBUJ (instance, channel);

}
```

mlBert_SaveBUJCalibrationValues

This API call is used to save BUJ Calibration Values for each channel.
Returns 1 on success, 0 on failure.

Used for ML4039JIT BUJ that includes ML407.

Example of use:

```
/*
* mlBert_SaveBUJCalibrationValues EntryPoint inside the DLL allows saving BUJ Calibration Values for
each channel */

 [DllImport (@"DLL/MLBert_API.dll", EntryPoint = “mlBert_SaveBUJCalibrationValues ")]
private static extern int mlBert_SaveBUJCalibrationValues (mlbertapi* instance, int Channel, double
Version, double *Data);

 /*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Param int Channel
* Param double Version: BUJ calibration version
*Param double *Data: Data to be saved

* Return int
* Allows saving BUJ calibration values
* Returns 1 on success, 0 on failure
*/

Public int SaveBUJCalibrationValues (mlbertapi* instance, int Channel, double Version, double *Data)

{
return mlBert_SaveBUJCalibrationValues (instance, Channel, Version, Data);

}
```

mlBert_SetAvagoRxConnectionStatus

This API call is used to set the Avago RX connection status.

Used for ML4039B, ML4039B-EQL.

Example of use:

```
/*
* mlBert_SetAvagoRxConnectionStatus EntryPoint inside the DLL allows setting the Avago RX connection
status */

 [DllImport (@"DLL/MLBert_API.dll", EntryPoint = "mlBert_SetAvagoRxConnectionStatus ")]
private static extern void mlBert_SetAvagoRxConnectionStatus (mlbertapi* instance, bool
RxSingleEnded);

 /*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Param int channel: channel number
* Param bool RxSingleEnded: true if single ended, false if differential
* Allows setting the Avago RX connection status
*/
Public void SetAvagoRxConnectionStatus (mlbertapi* instance, bool RxSingleEnded)

{
return mlBert_SetAvagoRxConnectionStatus (instance, RxSingleEnded);

}
```

mlBert_LineRateConfiguration39EVB

This API call is used to set the linerate for ML4039-EVB.
Returns 1 on success, 0 on failure.

Used for ML4039-EVB only.

Example of use:

```
/*
* mlBert_LineRateConfiguration39EVB EntryPoint inside the DLL allows setting the linerate for ML4039-
EVB */

 [DllImport (@"DLL/MLBert_API.dll", EntryPoint = “mlBert_LineRateConfiguration39EVB ")]
private static extern int mlBert_LineRateConfiguration39EVB (mlbertapi* instance, double linerate, int
clockSource);

 /*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Param double linerate
* Param int clockSource: (1 = internal, 0 = external)
* Allows setting the linerate for ML4039-EVB
*/
Public int LineRateConfiguration39EVB (mlbertapi* instance, double linerate, int clockSource)

{
return mlBert_LineRateConfiguration39EVB (instance, linerate, clockSource);

}
```

mlBert_Check3BRevision

This API call is used to read the ML4003B configuration.
Returns 1 on success, 0 on failure.

Used for ML4003B only.

Example of use:

```
/*
* mlBert_Check3BRevision EntryPoint inside the DLL allows reading the ML4003B configuration */

 [DllImport (@"DLL/MLBert_API.dll", EntryPoint = “mlBert_Check3BRevision ")]
private static extern int mlBert_Check3BRevision (mlbertapi* instance, int *LineRateRevision, int
*SFPRevision, int *HVRevision, int *XFPRevision);

 /*
* Public Interface to connect with DLL
* Param mlbertapi* instance

* Param int LineRateRevision: linerate revision
* Param int SFPRevision: SFP revision
* Param int HVRevision: HV revision
* Param int XFPRevision: XFP revision
* Allows reading the ML4003B configuration

*/
Public int Check3BRevision (mlbertapi* instance, int *LineRateRevision, int *SFPRevision, int
*HVRevision, int *XFPRevision)

{
return mlBert_Check3BRevision (instance, LineRateRevision, SFPRevision, HVRevision, XFPRevision);

}
```

mlBert_SFP_ReadSlaveAddress

This API call is used to read SFP Slave Address.
Returns true on success, false on failure.

Used for ML4003B and ML4003BX only.

Example of use:

```
/*
* mlBert_SFP_ReadSlaveAddress EntryPoint inside the DLL allows reading the SFP Slave Address */

 [DllImport (@"DLL/MLBert_API.dll", EntryPoint = “mlBert_SFP_ReadSlaveAddress ")]
private static extern bool mlBert_SFP_ReadSlaveAddress (mlbertapi* instance, int *SlaveAddress);

 /*
* Public Interface to connect with DLL
* Param mlbertapi* instance

* Param int *SlaveAddress: Pointer to the Slave address
* Allows reading the SFP Slave Address

*/
Public bool SFP_ReadSlaveAddress (mlbertapi* instance, int * SlaveAddress)

{
return mlBert_SFP_ReadSlaveAddress (instance, SlaveAddress);

}
```

mlBert_SFP_ SetSlaveAddress

This API call is used to set SFP Slave Address.
Returns true on success, false on failure.

Used for ML4003B and ML4003BX only.

Example of use:

```
/*
* mlBert_SFP_SetSlaveAddress EntryPoint inside the DLL allows setting the SFP Slave Address */

 [DllImport (@"DLL/MLBert_API.dll", EntryPoint = “mlBert_SFP_SetSlaveAddress ")]
private static extern bool mlBert_SFP_SetSlaveAddress (mlbertapi* instance, int SlaveAddress);

 /*
* Public Interface to connect with DLL
* Param mlbertapi* instance

* Param int SlaveAddress: the Slave address
* Allows setting the SFP Slave Address

*/
Public bool SFP_SetSlaveAddress (mlbertapi* instance, int SlaveAddress)

{
return mlBert_SFP_SetSlaveAddress (instance, SlaveAddress);

}
```

mlBert_SFP_ SetRegister

This API call is used to set SFP Register.
Returns true on success, false on failure.

Used for ML4003B and ML4003BX only.

Example of use:

```
/*
* mlBert_SFP_SetRegister EntryPoint inside the DLL allows setting the SFP Register */

 [DllImport (@"DLL/MLBert_API.dll", EntryPoint = “mlBert_SFP_SetRegister ")]
private static extern bool mlBert_SFP_SetRegister (mlbertapi* instance, UInt16 Register, UInt16 Value);

 /*
* Public Interface to connect with DLL
* Param mlbertapi* instance

* Param UInt16 Register: Register address

* Param UInt16 Value: Value for this register
* Allows setting the SFP Register

*/

Public bool SFP_ SetRegister (mlbertapi* instance, UInt16 Register, UInt16 Value)

{
return mlBert_SFP_SetRegister (instance, Register, Value);

}
```

mlBert_SFP_ GetRegister

This API call is used to get SFP Register.
Returns true on success, false on failure.

Used for ML4003B and ML4003BX only.

Example of use:

```
/*
* mlBert_SFP_GetRegister EntryPoint inside the DLL allows getting the SFP Register */

 [DllImport (@"DLL/MLBert_API.dll", EntryPoint = "mlBert_SFP_GetRegister ")]
private static extern bool mlBert_SFP_GetRegister (mlbertapi* instance, UInt16 Register, UInt16
*Value);

 /*
* Public Interface to connect with DLL
* Param mlbertapi* instance

* Param UInt16 Register: Register address

* Param UInt16 *Value: pointer Value for this register
* Allows getting the SFP Register

*/

Public bool SFP_ GetRegister (mlbertapi* instance, UInt16 Register, UInt16 *Value)

{
return mlBert_SFP_GetRegister (instance, Register, *Value);

}
```

mlBert_SFP_TXFaultStatus

This API call is used to check SFP TX Fault status.
Returns true on success, false on failure.

Used for ML4003B and ML4003BX only.

Example of use:

```
/*
* mlBert_SFP_TXFaultStatus EntryPoint inside the DLL allows checking SFP TX Fault status */

 [DllImport (@"DLL/MLBert_API.dll", EntryPoint = "mlBert_SFP_TXFaultStatus ")]
private static extern bool mlBert_SFP_TXFaultStatus (mlbertapi* instance);

 /*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Allows checking SFP TX Fault status

*/

Public bool SFP_TXFaultStatus (mlbertapi* instance)

{
return mlBert_SFP_TXFaultStatus (instance);

}
```

mlBert_SFP_RXLOSStatus

This API call is used to check SFP RX LOS status.
Returns true on success, false on failure.

Used for ML4003B and ML4003BX only.

Example of use:

```
/*
* mlBert_SFP_RXLOSStatus EntryPoint inside the DLL allows checking SFP RX LOS status */

 [DllImport (@"DLL/MLBert_API.dll", EntryPoint = “mlBert_SFP_RXLOSStatus ")]
private static extern bool mlBert_SFP_RXLOSStatus (mlbertapi* instance);

 /*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Allows checking SFP RX LOS status

*/

Public bool SFP_RXLOSStatus (mlbertapi* instance)

{
return mlBert_SFP_RXLOSStatus (instance);

}
```

mlBert_SFP_TXDisable

This API call is used to disable TX.
Returns true on success, false on failure.

Used for ML4003B and ML4003BX only.

Example of use:

```
/*
* mlBert_SFP_TXDisable EntryPoint inside the DLL allows disabling TX */

 [DllImport (@"DLL/MLBert_API.dll", EntryPoint = "APISFP_TXDisable")]
private static extern bool mlBert_SFP_TXDisable (mlbertapi* instance, int Mode);

 /*
* Public Interface to connect with DLL
* Param mlbertapi* instance

* Param int Mode: Mode = 0 to enable TX and 1 to Disable TX
* Allows disabling TX

*/

Public bool SFP_TXDisable (mlbertapi* instance, int Mode)

{
return mlBert_SFP_TXDisable (instance, Mode);

}
```

mlBert_SFP_ModulePresent

This API call is used to check SFP Module Present status.
Returns true on success, false on failure.

Used for ML4003B and ML4003BX only.

Example of use:

```
/*
* mlBert_SFP_ModulePresent EntryPoint inside the DLL allows checking SFP Module Present status*/

 [DllImport (@"DLL/MLBert_API.dll", EntryPoint = "mlBert_SFP_ModulePresent ")]
private static extern bool mlBert_SFP_ModulePresent (mlbertapi* instance);

 /*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Allows checking SFP Module Present status

*/

Public bool SFP_ModulePresent (mlbertapi* instance)

{
return mlBert_SFP_ModulePresent (instance);

}
```

mlBert_SFP_RS0RateSelect

This API call is used to select SFP RS0 Mode.
Returns true on success, false on failure.

Used for ML4003B and ML4003BX only.

Example of use:

```
/*
* mlBert_SFP_RS0RateSelect EntryPoint inside the DLL allows selecting SFP RS0 Mode */

 [DllImport (@"DLL/MLBert_API.dll", EntryPoint = “mlBert_SFP_RS0RateSelect ")]
private static extern bool mlBert_SFP_RS0RateSelect (mlbertapi* instance, int mode);

 /*
* Public Interface to connect with DLL
* Param mlbertapi* instance

* Param int mode: 1 high mode, 0 low mode
* Allows selecting SFP RS0 Mode

*/

Public bool SFP_RS0RateSelect (mlbertapi* instance, int mode)

{
return mlBert_SFP_RS0RateSelect (instance, mode);

}
```

mlBert_SFP_RS1RateSelect

This API call is used to select SFP RS1 Mode.
Returns true on success, false on failure.

Used for ML4003B and ML4003BX only.

Example of use:

```
/*
* mlBert_SFP_RS1RateSelect EntryPoint inside the DLL allows selecting SFP RS1 Mode */

 [DllImport (@"DLL/MLBert_API.dll", EntryPoint = “mlBert_SFP_RS1RateSelect ")]
private static extern bool mlBert_SFP_RS1RateSelect (mlbertapi* instance, int mode);

 /*
* Public Interface to connect with DLL
* Param mlbertapi* instance

* Param int mode: 1 high mode, 0 low mode
* Allows selecting SFP RS1 Mode

*/

Public bool SFP_RS1RateSelect (mlbertapi* instance, int mode)

{
return mlBert_SFP_RS1RateSelect (instance, mode);

}
```

mlBert_SFP_GetTemperature

This API call is used to read SFP Module Temperature.
Returns true on success, false on failure.

Used for ML4003B and ML4003BX only.

Example of use:

```
/*
* mlBert_SFP_GetTemperature EntryPoint inside the DLL allows reading SFP Module Temperature */

 [DllImport (@"DLL/MLBert_API.dll", EntryPoint = “mlBert_SFP_GetTemperature ")]
private static extern bool mlBert_SFP_GetTemperature (mlbertapi* instance, double* Temperature);

 /*
* Public Interface to connect with DLL
* Param mlbertapi* instance

* Param double* Temperature: Temperature Pointer
* Allows reading SFP Module Temperature

*/

Public bool SFP_GetTemperature (mlbertapi* instance, double* Temperature)

{
return mlBert_SFP_GetTemperature (instance, Temperature);

}
```

mlBert_SFP_GetVolt_sens

This API call is used to read SFP Module Voltage level.
Returns true on success, false on failure.

Used for ML4003B and ML4003BX only.

Example of use:

```
/*
* mlBert_SFP_GetVolt_sens EntryPoint inside the DLL allows reading SFP Module Voltage level */

 [DllImport (@"DLL/MLBert_API.dll", EntryPoint = “mlBert_SFP_GetVolt_sens")]
private static extern bool mlBert_SFP_GetVolt_sens (mlbertapi* instance, double* Voltage);

 /*
* Public Interface to connect with DLL
* Param mlbertapi* instance

* Param double* Voltage: Temperature Voltage
* Allows reading SFP Module Voltage level

*/

Public bool SFP_GetVolt_sens (mlbertapi* instance, double* Voltage)

{
return mlBert_SFP_GetVolt_sens (instance, Voltage);

}
```

mlBert_SFP_GetCurrent_sens

This API call is used to read SFP Module current level.
Returns true on success, false on failure.

Used for ML4003B and ML4003BX only.

Example of use:

```
/*
* mlBert_SFP_GetCurrent_sens EntryPoint inside the DLL allows reading SFP Module current level */

 [DllImport (@"DLL/MLBert_API.dll", EntryPoint = "mlBert_SFP_GetCurrent_sens")]
private static extern bool mlBert_SFP_GetCurrent_sens (mlbertapi* instance, double* Current);

 /*
* Public Interface to connect with DLL
* Param mlbertapi* instance

* Param double* Current: Temperature Current
* Allows reading SFP Module current level

*/

Public bool SFP_GetCurrent _sens (mlbertapi* instance, double* Current)

{
return mlBert_SFP_GetCurrent_sens (instance, Current);

}
```

mlBert_XFP_ SetRegister

This API call is used to set XFP Register.
Returns true on success, false on failure.

Used for ML4003BX only.

Example of use:

```
/*
* mlBert_XFP_SetRegister EntryPoint inside the DLL allows setting the XFP Register */

 [DllImport (@"DLL/MLBert_API.dll", EntryPoint = "mlBert_XFP_SetRegister ")]
private static extern bool mlBert_XFP_SetRegister (mlbertapi* instance, UInt16 Register, UInt16 Value);

 /*
* Public Interface to connect with DLL
* Param mlbertapi* instance

* Param UInt16 Register: Register address

* Param UInt16 Value: Value for this register
* Allows setting the XFP Register

*/

Public bool XFP_ SetRegister (mlbertapi* instance, UInt16 Register, UInt16 Value)

{
return mlBert_XFP_SetRegister (instance, Register, Value);

}
```

mlBert_XFP_GetRegister

This API call is used to get XFP Register.
Returns true on success, false on failure.

Used for ML4003BX only.

Example of use:

```
/*
* mlBert_XFP_GetRegister EntryPoint inside the DLL allows getting the XFP Register */

 [DllImport (@"DLL/MLBert_API.dll", EntryPoint = “mlBert_XFP_GetRegister ")]
private static extern bool mlBert_XFP_GetRegister (mlbertapi* instance, UInt16 Register, UInt16
*Value);

 /*
* Public Interface to connect with DLL
* Param mlbertapi* instance

* Param UInt16 Register: Register address

* Param UInt16 *Value: pointer Value for this register
* Allows getting the XFP Register

*/

Public bool XFP_ GetRegister (mlbertapi* instance, UInt16 Register, UInt16 *Value)

{
return mlBert_XFP_GetRegister (instance, Register, *Value);

}
```

mlBert_XFP_Interrupt

This API call is used to Interrupt XFP module.
Returns true on success, false on failure.

Used for ML4003BX only.

Example of use:

```
/*
* mlBert_XFP_Interrupt EntryPoint inside the DLL allows Interrupting XFP module */

 [DllImport (@"DLL/MLBert_API.dll", EntryPoint = “mlBert_XFP_Interrupt")]
private static extern bool mlBert_XFP_Interrupt (mlbertapi* instance, UInt16 data);

 /*
* Public Interface to connect with DLL
* Param mlbertapi* instance

* Param UInt16 data: 0: XFP interrupt exists, 1: No interrupt
* Allows Interrupting XFP module

*/

Public bool XFP_Interrupt (mlbertapi* instance, UInt16 data)

{
return mlBert_XFP_Interrupt (instance, data);

}
```

mlBert_XFP_RXLOSStatus

This API call is used to check XFP RX LOS status.
Returns 1 on success, 0 on failure.

Used for ML4003BX only.

Example of use:

```
/*
* mlBert_XFP_RXLOSStatus EntryPoint inside the DLL allows checking XFP RX LOS status */

 [DllImport (@"DLL/MLBert_API.dll", EntryPoint = “mlBert_XFP_RXLOSStatus ")]
private static extern int mlBert_XFP_RXLOSStatus (mlbertapi* instance);

 /*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Allows checking XFP RX LOS status

*/

Public int XFP_RXLOSStatus (mlbertapi* instance)

{
return mlBert_XFP_RXLOSStatus (instance);

}
```

mlBert_XFP_ModulePresent

This API call is used to check XFP Module Present status.
Returns 1 on success, 0 on failure.

Used for ML4003BX only.

Example of use:

```
/*
* mlBert_XFP_ModulePresent EntryPoint inside the DLL allows checking XFP Module Present status*/

 [DllImport (@"DLL/MLBert_API.dll", EntryPoint = “mlBert_XFP_ModulePresent ")]
private static extern int mlBert_XFP_ModulePresent (mlbertapi* instance);

 /*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Allows checking XFP Module Present status

*/

Public int XFP_ModulePresent (mlbertapi* instance)

{
return mlBert_XFP_ModulePresent (instance);

}
```

mlBert_XFP_TXDisable

This API call is used to disable TX.
Returns true on success, false on failure.

Used for ML4003BX only.

Example of use:

```
/*
* mlBert_XFP_TXDisable EntryPoint inside the DLL allows disabling TX */

 [DllImport (@"DLL/MLBert_API.dll", EntryPoint = “mlBert_XFP_TXDisable")]
private static extern bool mlBert_XFP_TXDisable (mlbertapi* instance, int Mode);

 /*
* Public Interface to connect with DLL
* Param mlbertapi* instance

* Param int Mode: Mode = 0 to enable TX and 1 to Disable TX
* Allows disabling TX

*/

Public bool XFP_TXDisable (mlbertapi* instance, int Mode)

{
return mlBert_XFP_TXDisable (instance, Mode);

}
```

mlBert_XFP_ModuleSelection

This API call is used to select the XFP module.
Returns 1 on success, 0 on failure.

Used for ML4003BX only.

Example of use:

```
/*
* mlBert_XFP_ModuleSelection EntryPoint inside the DLL allows selecting the XFP module */

 [DllImport (@"DLL/MLBert_API.dll", EntryPoint = “mlBert_XFP_ModuleSelection")]
private static extern int mlBert_XFP_ModuleSelection (mlbertapi* instance, int Mode);

 /*
* Public Interface to connect with DLL
* Param mlbertapi* instance

* Param int Mode: Mode = 0 XFP Module is selected, Mode = 1 XFP module is not-selected


* Allows selecting the XFP module

*/

Public int XFP_ModuleSelection (mlbertapi* instance, int Mode)

{
return mlBert_XFP_ModuleSelection (instance, Mode);

}
```

mlBert_XFP_PowerDown

This API call is used to power down the XFP module.
Returns 1 on success, 0 on failure.

Used for ML4003BX only.

Example of use:

```
/*
* mlBert_XFP_PowerDown EntryPoint inside the DLL allows powering down the XFP module */

 [DllImport (@"DLL/MLBert_API.dll", EntryPoint = “mlBert_XFP_PowerDown ")]
private static extern int mlBert_XFP_PowerDown (mlbertapi* instance, int Mode);

 /*
* Public Interface to connect with DLL
* Param mlbertapi* instance

* Param int Mode: Mode = 0 XFP Module is powered, Mode = 1 XFP Module is powered down


* Allows powering down the XFP module

*/

Public int XFP_PowerDown (mlbertapi* instance, int Mode)

{
return mlBert_XFP_PowerDown (instance, Mode);

}
```

mlBert_XFP_GetVolt_sens

This API call is used to read XFP Module Voltage level.
Returns true on success, false on failure.

Used for ML4003BX only.

Example of use:

```
/*
* mlBert_XFP_GetVolt_sens EntryPoint inside the DLL allows reading XFP Module Voltage level */

 [DllImport (@"DLL/MLBert_API.dll", EntryPoint = "mlBert_XFP_GetVolt_sens")]
private static extern bool mlBert_XFP_GetVolt_sens (mlbertapi* instance, int mode, double* Voltage);

 /*
* Public Interface to connect with DLL
* Param mlbertapi* instance

* Param int mode: 0: disable regulator, 1: enable regulator

* Param double* Voltage: control Module 1.8V, 5V
* Allows reading XFP Module Voltage level

*/

Public bool XFP_GetVolt_sens (mlbertapi* instance, int mode, double* Voltage)

{
return mlBert_XFP_GetVolt_sens (instance, mode, Voltage);

}
```

mlBert_CoarseTuning

This API call is used to run the coarse tune algorithm.

Used for AVAGO like ML4039B, ML4039B-JIT, ML4039B-EQL, ML4003B and ML4003BX.

Example of use:

```
/*
* mlBert_CoarseTuning EntryPoint inside the DLL allows running the coarse tune algorithm */

 [DllImport (@"DLL/MLBert_API.dll", EntryPoint = “mlBert_CoarseTuning ")]
private static extern void mlBert_CoarseTuning (mlbertapi* instance, int channel);

 /*
* Public Interface to connect with DLL
* Param mlbertapi* instance

* Param int channel
* Allows running the coarse tune algorithm

*/

Public void CoarseTuning (mlbertapi* instance, int channel)

{
return mlBert_CoarseTuning (instance, channel);

}
```

mlBert_FineTuning

This API call is used to run the fine tune algorithm.

Used for AVAGO like ML4039B, ML4039B-JIT, ML4039B-EQL, ML4003B and ML4003BX.

Example of use:

```
/*
* mlBert_FineTuning EntryPoint inside the DLL allows running the fine tune algorithm */

 [DllImport (@"DLL/MLBert_API.dll", EntryPoint = "mlBert_FineTuning ")]
private static extern void mlBert_FineTuning (mlbertapi* instance, int channel);

 /*
* Public Interface to connect with DLL
* Param mlbertapi* instance

* Param int channel
* Allows running the fine tune algorithm

*/

Public void FineTuning (mlbertapi* instance, int channel)

{
return mlBert_FineTuning (instance, channel);

}
```

mlBert_HaltTuning

This API call is used to run the halt tune algorithm.

Used for AVAGO like ML4039B, ML4039B-JIT, ML4039B-EQL, ML4003B and ML4003BX.

Example of use:

```
/*
* mlBert_HaltTuning EntryPoint inside the DLL allows running the halt tune algorithm */

 [DllImport (@"DLL/MLBert_API.dll", EntryPoint = “mlBert_HaltTuning ")]
private static extern void mlBert_HaltTuning (mlbertapi* instance, int channel);

 /*
* Public Interface to connect with DLL
* Param mlbertapi* instance

* Param int channel
* Allows running the halt tune algorithm

*/

Public void HaltTuning (mlbertapi* instance, int channel)

{
return mlBert_HaltTuning (instance, channel);

}
```

mlBert_ContinuousTuning

This API call is used to run the Continuous tune algorithm.

Used for AVAGO like ML4039B, ML4039B-JIT, ML4039B-EQL, ML4003B and ML4003BX.

Example of use:

```
/*
* mlBert_ContinuousTuning EntryPoint inside the DLL allows running the continuous tune algorithm */

 [DllImport (@"DLL/MLBert_API.dll", EntryPoint = “mlBert_ContinuousTuning ")]
private static extern void mlBert_ContinuousTuning (mlbertapi* instance, int channel);

 /*
* Public Interface to connect with DLL
* Param mlbertapi* instance

* Param int channel
* Allows running the continuous tune algorithm

*/

Public void ContinuousTuning (mlbertapi* instance, int channel)

{
return mlBert_ContinuousTuning (instance, channel);

}
```

mlBert_FPGA_Write_PatternLength

This API call is used to write the pattern length for FPGA.

Used for ML4003BX.

Example of use:

```
/*
* mlBert_FPGA_Write_PatternLength EntryPoint inside the DLL allows writing the pattern length for
FPGA */

 [DllImport (@"DLL/MLBert_API.dll", EntryPoint = “mlBert_FPGA_Write_PatternLength ")]
private static extern void mlBert_FPGA_Write_PatternLength (mlbertapi* instance, int channel, UInt16
Data);

 /*
* Public Interface to connect with DLL
* Param mlbertapi* instance

* Param int channel

* Param UInt16 Data: pattern length to set
* Allows writing the pattern length for FPGA

*/

Public void FPGA_Write_PatternLength (mlbertapi* instance, int channel, UInt16 Data)

{
return mlBert_FPGA_Write_PatternLength (instance, channel, Data);

}
```

mlBert_FPGA_Read_PatternLength

This API call is used to read the pattern length for FPGA.
Returns 1 on success, 0 on failure.

Used for ML4003BX.

Example of use:

```
/*
* mlBert_FPGA_Read_PatternLength EntryPoint inside the DLL allows reading the pattern length for
FPGA */

 [DllImport (@"DLL/MLBert_API.dll", EntryPoint = “mlBert_FPGA_Read_PatternLength ")]
private static extern int mlBert_FPGA_Read_PatternLength (mlbertapi* instance, int channel, UInt16
*Data);

 /*
* Public Interface to connect with DLL
* Param mlbertapi* instance

* Param int channel

* Param UInt16 *Data: pattern length
* Allows reading the pattern length for FPGA

*/

Public int FPGA_Read_PatternLength (mlbertapi* instance, int channel, UInt16 *Data)

{
return mlBert_FPGA_Read_PatternLength (instance, channel, Data);

}
```

mlBert_FPGA_Pattern_Data

This API call is used to read or write the pattern data for FPGA.
Returns 1 on success, 0 on failure.

Used for ML4003BX.

Example of use:

```
/*
* mlBert_FPGA_Pattern_Data EntryPoint inside the DLL allows reading or writing the pattern data for
FPGA */

 [DllImport (@"DLL/MLBert_API.dll", EntryPoint = “mlBert_FPGA_Pattern_Data")]
private static extern int mlBert_FPGA_Pattern_Data (mlbertapi* instance, int channel, int mode, UInt64
*Pattern);

 /*
* Public Interface to connect with DLL
* Param mlbertapi* instance

* Param int channel

* Param int mode: 0 for reading, 1 for writing

* Param UInt64 * Pattern
* Allows reading or writing the pattern data for FPGA

*/

Public int FPGA_Pattern_Data (mlbertapi* instance, int channel, int mode, UInt64 *Pattern)

{
return mlBert_FPGA_Pattern_Data (instance, channel, mode, Pattern);

}
```

mlBert_FPGA_Clk_Output_Divider

This API call is used to read or write the clock output divider for FPGA.
Returns 1 on success, 0 on failure.

Used for ML4003BX.

Example of use:

```
/*
* mlBert_FPGA_Clk_Output_Divider EntryPoint inside the DLL allows reading or writing the clock output
divider for FPGA */

 [DllImport (@"DLL/MLBert_API.dll", EntryPoint = "mlBert_FPGA_Clk_Output_Divider ")]
private static extern int mlBert_FPGA_Clk_Output_Divider (mlbertapi* instance, int channel, int mode,
UInt16 *Data);

 /*
* Public Interface to connect with DLL
* Param mlbertapi* instance

* Param int channel

* Param int mode: 0 for reading, 1 for writing

* Param UInt16 * Data: divider
* Allows reading or writing the clock output divider for FPGA

*/

Public int FPGA_Clk_Output_Divider (mlbertapi* instance, int channel, int mode, UInt16 *Data)

{
return mlBert_FPGA_Clk_Output_Divider (instance, channel, mode, Data);

}
```

mlBert_Enable_Deep_System_Loopback_GB2

This API call is used to enable the loopback internally.

Used for GearBox2 like ML4039 and ML4039-JIT.

Example of use:

```
/*
* mlBert_Enable_Deep_System_Loopback_GB2 EntryPoint inside the DLL allows enabling the loopback
internally */

 [DllImport (@"DLL/MLBert_API.dll", EntryPoint = “mlBert_Enable_Deep_System_Loopback_GB2")]
private static extern void mlBert_Enable_Deep_System_Loopback_GB2 (mlbertapi* instance);

 /*
* Public Interface to connect with DLL

* Allows enabling the loopback internally

*/

Public int Enable_Deep_System_Loopback_GB2 (mlbertapi* instance)

{
return mlBert_Enable_Deep_System_Loopback_GB2 (instance);

}
```

mlBert_Disable_Deep_System_Loopback_GB2

This API call is used to disable the loopback internally.

Used for GearBox2 like ML4039 and ML4039-JIT.

Example of use:

```
/*
* mlBert_Disable_Deep_System_Loopback_GB2 EntryPoint inside the DLL allows disabling the loopback
internally */

 [DllImport (@"DLL/MLBert_API.dll", EntryPoint = “mlBert_Disable_Deep_System_Loopback_GB2")]
private static extern int mlBert_Disable_Deep_System_Loopback_GB2 (mlbertapi* instance);

 /*
* Public Interface to connect with DLL

* Allows disabling the loopback internally

*/

Public int Disable_Deep_System_Loopback_GB2 (mlbertapi* instance)

{
return mlBert_Disable_Deep_System_Loopback_GB2 (instance);

}
```

mlBert_QuickSettings

This API call is used to apply all the configuration settings.

Used for PAM boards.

Example of use:

```
/*
* mlBert_QuickSettings EntryPoint inside the DLL allows applying configuration settings */

 [DllImport (@"DLL/MLBert_API.dll", EntryPoint = "mlBert_QuickSettings ")]
private static extern bool mlBert_QuickSettings (mlbertapi* instance, byte Mode, byte GrayMapping,
byte PreCoding, byte FECenable, byte FECmode, UInt8 Rate, byte Clock, UInt8 TXPattern, UInt8
RXPattern, UInt8 CML, UInt16 Amplitude, UInt8 InnerAmplitude, UInt8 Pre, UInt8 Post, byte Side);

 /*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Param byte Mode: 1 for NRZ, 0 for PAM
* Param byte GrayMapping: 1 for enabling, 0 for disabling
* Param byte PreCoding: 1 for enabling, 0 for disabling
* Param byte FECenable: 1 for enabling, 0 for disabling
* Param byte FECmode: 1 for KP4, 0 for _802_3bj
* Param UInt8 Rate: Index of the rate
* Param byte Clock: 1 for internal clock
* Param UInt8 TXPattern:  0 for PRBS9, 1 for PRBS15, 2 for PRBSQ13
* Param UInt8 RXPattern:  0 for PRBS9, 1 for PRBS15, 2 for PRBSQ13
* Param UInt8 CML: 50, 100, 150 or 200
* Param UInt16 Amplitude: amplitude level
* Param UInt8 Pre: pre emphasis value
* Param UInt8 Post: post emphasis value
* Param byte Side: 0 for line side, 1 for host side

*/

Public bool QuickSettings (mlbertapi* instance, byte Mode, byte GrayMapping, byte PreCoding, byte
FECenable, byte FECmode, UInt8 Rate, byte Clock, UInt8 TXPattern, UInt8 RXPattern, UInt8 CML, UInt16
Amplitude, UInt8 InnerAmplitude, UInt8 Pre, UInt8 Post, byte Side);

{
return mlBert_QuickSettings (instance, Mode, GrayMapping, PreCoding, FECenable, FECmode, Rate,
Clock, TXPattern, RXPattern, CML, Amplitude, InnerAmplitude, Pre, Post, Side);

}
```

mlBert_Check4039EVBRev

This API call is used to check the ML4039-EVB revision.

Used for ML4039-EVB.

Example of use:

```
/*
* mlBert_Check4039EVBRev EntryPoint inside the DLL allows checking the ML4039-EVB revision */

 [DllImport (@"DLL/MLBert_API.dll", EntryPoint = “mlBert_Check4039EVBRev ")]
private static extern int mlBert_Check4039EVBRev (mlbertapi* instance, int *EVBRevision);

 /*
* Public Interface to connect with DLL
* Param EVBRevision: get the EVB revision (1: EVB rev B)
* Allows disabling the loopback internally

*Return EVB revision

*/

Public int Check4039EVBRev (mlbertapi* instance, int *EVBRevision)

{
return mlBert_Check4039EVBRev (instance, *EVBRevision);

}
```

mlBert_LineRateConfiguration_4070

This API call is used to set the Line Rate of the connected BERT. It returns true if successful, false if a problem occurs while setting the line Rate.

For every Line Rate a File should be provided or generated that will configure the Silab, location of the file will be set by using “APIConfigureApplication”.

To generate the clock file the user should use a special GUI provided by MultiLane, or a C# API (ClockLibrary.dll) that generate clock files also provided by multilane.

Used for ML4070-QSFP and ML4070-SFP.

Example of use:

```
/*
 * mlBert_LineRateConfiguration_4070 EntryPoint inside the DLL is responsible to set the Line Rate
*/
[DllImport(@"DLL/MLBert_API.dll", EntryPoint = " mlBert_LineRateConfiguration_4070")]
private static extern bool mlBert_LineRateConfiguration_4070 (mlbertapi* instance, double lineRate);


/*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Param double lineRate
* return bool
* Allows to set Line Rate generated by the connected board
* DLL use lineRate value to set the Line Rate of the connected board
* Boards connected are identified using the instance parameter
*/

public bool LineRateConfiguration_4070(mlbertapi* instance, double lineRate)
 {
return mlBert_LineRateConfiguration_4070 (instance, lineRate);
 }
```

mlBert_RealTimeBEREnableByQSFP

This API call is used to enable/disable Real Time BER by QSFP modules. Results will be collected using APIRealTimeBER.

Return 1 upon success, 0 if failure.

Used for ML ODVT , ML4070-QSFP and ML4070-QDD.

Example of use:

```
/*
* mlBert_RealTimeBEREnableByQSFP EntryPoint inside the DLL allows to enable/disable real time BER
* Results will be collected using APIReadRealTimeBER
*/

 [DllImport(@"DLL/MLBert_API.dll", EntryPoint = " mlBert_RealTimeBEREnableByQSFP")]
 static extern int mlBert_RealTimeBEREnableByQSFP (mlbertapi* instance, int qsfpIdx, int enable);

/*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Param int qsfpIdx: index of the module
* Param int enable (Enable = 1, Disable = 0)
* Return int
* Allows to enable/disable real time BER
* Return 1 on success, 0 on failure
* Boards connected are identified using the instance parameter
*/

public int RealTimeBEREnableByQSFP (mlbertapi* instance, int qsfpIdx, int enable)
{
  mlBert_RealTimeBEREnableByQSFP (instance, qsfpIdx, enable);
 }
```

mlBert_RealTimeBEREnableByChannel

This API call is used to enable/disable Real Time BER by channel for SFP/QFP and QDD modules. Results will be collected using APIRealTimeBER.

Return 1 upon success, 0 if failure.

Used for ML ODVT , ML4070-SFP, ML4070-QSFP and ML4070-QDD.

Example of use:

```
/*
* mlBert_RealTimeBEREnableByChannel EntryPoint inside the DLL allows to enable/disable real time
BER Results will be collected using APIReadRealTimeBER
*/

 [DllImport(@"DLL/MLBert_API.dll", EntryPoint = " mlBert_RealTimeBEREnableByChannel")]
 static extern int mlBert_RealTimeBEREnableByChannel (mlbertapi* instance, int ChannelIdx, int enable);

/*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Param int channelIdx: index of the channel
* Param int enable (Enable = 1, Disable = 0)
* Return int
* Allows to enable/disable real time BER
* Return 1 on success, 0 on failure
* Boards connected are identified using the instance parameter
*/

public int RealTimeBEREnableByChannel (mlbertapi* instance, int channelIdx, int enable)
{
  mlBert_RealTimeBEREnableByChannel (instance, channelIdx, enable);
 }
```

mlBert_QSFP_Block_Read

This API call is used to read a block of registers from the QSFP module.
It returns true if successful, false if a problem occurs while reading.

Used for ML ODVT and ML4070-QSFP.

Example of use:

```
/*
 * mlBert_QSFP_Block_Read EntryPoint inside the DLL is responsible to Read a block of register from
the QSFP module
*/
[DllImport(@"DLL/MLBert_API.dll", EntryPoint = " mlBert_QSFP_Block_Read")]
public static extern bool mlBert_QSFP_Block_Read (mlbertapi* instance, int Module, byte length, byte
address, byte[] calib);

/*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Param int Module: Select QSFP module
* Param byte length: Length of the data to read
* Param byte address: starting address
* Param byte[] Calib: to store the read data
* return bool true if success, false if failed
* Allows to read a block of register from the QSFP module
* Modules are identified using the module parameter
* Boards connected are identified using the instance parameter
*/

public bool QSFP_Block_Read(int Module, byte length, byte address, byte[] calib)
{
return mlBert_QSFP_Block_Read (instance, Module, length, address, calib);
}
```

mlBert_QSFP_Block_Write

This API call is used to write a block of data on the QSFP module. It returns true if successful, false if a problem occurs while writting.

Used for ML ODVT and ML4070-QSFP.

Example of use:

```
/*
 * mlBert_QSFP_Block_Write EntryPoint inside the DLL is responsible to Write a block of data to the
QSFP module's registers
*/

[DllImport(@"DLL/MLBert_API.dll", EntryPoint = " mlBert_QSFP_Block_Write")]
public static extern bool mlBert_QSFP_Block_Write (mlbertapi* instance, int Module, byte length, byte
address, byte[] calib);

/*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Param int Module: Select QSFP module
* Param byte length: Length of the data to read
* Param byte address: starting address
* Param byte[] Calib: array of data to be written
* return bool true if success, false if failed
* Allows to write a block of data to the module
* Modules are identified using the module parameter
* Boards connected are identified using the instance parameter
*/

public bool QSFP_Block_Write(int Module, byte length, byte address, byte[] calib)
{
return mlBert_QSFP_Block_Write (instance, Module, length, address, calib);
}
```

mlBert_QSFP_Get_temperature

This API call is used to read QSFP Module internally measured temperature.
Returns true on success, false on failure.

Used for ML ODVT and ML4070-QSFP only.

Example of use:

```
/*
 * mlBert_QSFP_Get_temperature EntryPoint inside the DLL allows reading QSFP Module Temperature
*/

[DllImport(@"DLL/MLBert_API.dll", EntryPoint = " mlBert_QSFP_Get_temperature")]
public static extern bool mlBert_QSFP_Get_temperature (mlbertapi* instance, int Module, ref double
Temp);

/*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Param int Module: Select QSFP module
* Param ref double Current: to store the Temperature's value

* the value is read from the QSFP module and need to be divided by 256 to get the temperature
* Allows reading QSFP Module internally measured temperature

*/

public bool QSFP_Get_Temperature(int Module, ref double Temperature)

{

return mlBert_QSFP_Get_temperature (instance, Module, ref Temperature);

}
```

mlBert_QSFP_Get_VCCRX

This API call is used to read QSFP Voltage measurement of RX power rail.
Returns true on success, false on failure.

Used for ML ODVT and ML4070-QSFP only.

Example of use:

```
/*
 * mlBert_QSFP_Get_VCCRX EntryPoint inside the DLL allows reading QSFP RX Voltage measurement
*/
[DllImport(@"DLL/MLBert_API.dll", EntryPoint = " mlBert_QSFP_Get_VCCRX")]
public static extern bool mlBert_QSFP_Get_VCCRX (mlbertapi* instance, int Module, ref double Data);

/*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Param int Module: Select QSFP module
* Param ref double Data: to store the voltage

* The read value should be divided by 256 to get the RX voltage

*/

public bool QSFP_Get_VCCRX(int Module, ref double Data)

{

 return mlBert_QSFP_Get_VCCRX (instance, Module, ref Data);

}
```

mlBert_QSFP_Get_VCCTX

This API call is used to read QSFP Voltage measurement of TX power rail.
Returns true on success, false on failure.

Used for ML ODVT and ML4070-QSFP only.

Example of use:

```
/*
 * mlBert_QSFP_Get_VCCTX EntryPoint inside the DLL allows reading QSFP TX Voltage measurement
*/
[DllImport(@"DLL/MLBert_API.dll", EntryPoint = " mlBert_QSFP_Get_VCCTX")]
public static extern bool mlBert_QSFP_Get_VCCTX (mlbertapi* instance, int Module, ref double Data);
/*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Param int Module: Select QSFP module
* Param ref double Data: to store the voltage

*/

public bool QSFP_Get_VCCTX(int Module, ref double Data)

{

return mlBert_QSFP_Get_VCCTX (instance, Module, ref Data);

 }
```

mlBert_QSFP_GetVCC

This API call is used to read QSFP Voltage measurement of VCC power rail.
Returns true on success, false on failure.

Used for ML ODVT and ML4070-QSFP only.

Example of use:

```
/*
* mlBert_QSFP_GetVCC EntryPoint inside the DLL allows reading QSFP VCC Voltage measurement
*/
[DllImport(@"DLL/MLBert_API.dll", EntryPoint = " mlBert_QSFP_GetVCC")]
public static extern bool mlBert_QSFP_GetVCC (mlbertapi* instance, int Module, ref double Data);

/*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Param int Module: Select QSFP module
* Param ref double Data: to store the voltage

*/

public bool QSFP_GetVCC(int Module, ref double Data)

{

return mlBert_QSFP_GetVCC (instance, Module, ref Data);

}
```

mlBert_QSFP_IntL

This API call is used to read module interrupt signal.
Returns true on success, false on failure.

Used for ML ODVT and ML4070-QSFP only.

Example of use:

```
/*
 * mlBert_QSFP_IntL EntryPoint inside the DLL allows reading QSFP the module interrupt signal.
*/

[DllImport(@"DLL/MLBert_API.dll", EntryPoint = " mlBert_QSFP_IntL")]

public static extern bool mlBert_QSFP_IntL (mlbertapi* instance, int Module, ref bool status);

/*
* Public Interface to connect with DLL
* Param mlbertapi* instance

* Param int Module: Select QSFP module

* Param ref bool status: represent the status of the interrupt signal

*/

public bool QSFP_IntL(int Module, ref bool status)

{ return mlBert_QSFP_IntL (instance, Module, ref status);

 }
```

mlBert_QSFP_LPMODE

This API call is used to set or release the module from the Low Power mode.
Returns true on success, false on failure.

Used for ML ODVT and ML4070-QSFP only.

Example of use:

```
/*
 * mlBert_QSFP_LPMODE EntryPoint inside the DLL allows setting or releasing the QSFP module from
Low Power mode.
*/
[DllImport(@"DLL/MLBert_API.dll", EntryPoint = " mlBert_QSFP_LPMODE")]
public static extern bool mlBert_QSFP_LPMODE (mlbertapi* instance, int Module, bool asserted);

/*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Param int Module: Select QSFP module
* Param bool asserted: if true set the module to the Low Power mode, else release it from the Low
Power mode.

*/

public bool QSFP_LPMODE(int Module, bool asserted)

{ return mlBert_QSFP_LPMODE (instance, Module, asserted);

}
```

mlBert_QSFP_MODSEL

This API call is used to select the module for the I2C communication.
Returns true on success, false on failure.

Used for ML ODVT and ML4070-QSFP only.

Example of use:

```
/*
 * mlBert_QSFP_MODSEL EntryPoint inside the DLL allows selecting the QSFP module for I2C
communication.
*/

[DllImport(@"DLL/MLBert_API.dll", EntryPoint = " mlBert_QSFP_MODSEL")]

public static extern bool mlBert_QSFP_MODSEL (mlbertapi* instance, int Module, bool asserted);

/*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Param int Module: Select QSFP module
* Param bool asserted: select the module if true, unselect if false.

*/

public bool QSFP_MODSEL(int Module, bool asserted)

{

 return mlBert_QSFP_MODSEL (instance, Module, asserted);

}
```

mlBert_QSFP_MODPRS

This API call is used to check if the module is present.
Returns true on success, false on failure.

Used for ML ODVT and ML4070-QSFP only.

Example of use:

```
/*
 * mlBert_QSFP_MODPRS EntryPoint inside the DLL allows reading the QSFP module present status.
*/

[DllImport(@"DLL/MLBert_API.dll", EntryPoint = " mlBert_QSFP_MODPRS")]
public static extern bool mlBert_QSFP_MODPRS (mlbertapi* instance, int Module, ref bool status);

/*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Param int Module: Select QSFP module
* Param ref bool status: true if the module is present, false otherwise.

*/

public bool QSFP_MODPRS(int Module, ref bool status)

{

 return mlBert_QSFP_MODPRS (instance, Module, ref status);

}
```

mlBert_QSFP_RESET

This API call is used to set/release the QSFP module from reset mode.
Setting the module to the reset mode for longer than the minimum pulse length initiates a complete module reset, returning all user module settings to their default state.
Returns true on success, false on failure.

Used for ML ODVT and ML4070-QSFP only.

Example of use:

```
/*
 * mlBert_QSFP_RESET EntryPoint inside the DLL allows setting/releasing the QSFP reset pin.
*/

[DllImport(@"DLL/MLBert_API.dll", EntryPoint = " mlBert_QSFP_RESET")]

public static extern bool mlBert_QSFP_RESET (mlbertapi* instance, int Module, bool asserted);

/*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Param int Module: Select QSFP module
* Param bool asserted: set if true, clear if false.

*/

public bool QSFP_RESET(int Module, bool asserted)

{

return mlBert_QSFP_RESET (instance, Module, asserted);

}
```

mlBert_QSFP_P3V3_Current_Monitor

This API call is used to read QSFP current measurement passing through VCC power rail.
Returns true on success, false on failure.

Used for ML ODVT and ML4070-QSFP only.

Example of use:

```
/*
 * mlBert_QSFP_P3V3_Current_Monitor EntryPoint inside the DLL allows reading QSFP current
measurement passing through VCC power rail.
*/

[DllImport(@"DLL/MLBert_API.dll", EntryPoint = " mlBert_QSFP_P3V3_Current_Monitor")]

public static extern bool mlBert_QSFP_P3V3_Current_Monitor (mlbertapi* instance, int Module, ref
double data);

/*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Param int Module: Select QSFP module
* Param ref double Data: to store the current

*/

public bool QSFP_P3V3_Current_Monitor(int Module, ref double data)

{

 return mlBert_QSFP_P3V3_Current_Monitor (instance, Module, ref data);

}
```

mlBert_QSFP_GetRegister

This API call is used to get QSFP Register.
Returns true on success, false on failure.

Used for ML ODVT and ML4070-QSFP only.

Example of use:

```
/*
 * mlBert_QSFP_GetRegister EntryPoint inside the DLL allows reading QSFP module register.
*/

[DllImport(@"DLL/MLBert_API.dll", EntryPoint = " mlBert_QSFP_GetRegister")]

public static extern bool mlBert_QSFP_GetRegister (mlbertapi* instance, int Module, UInt16 Register,
ref UInt16 Value);

/*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Param int Module: Select QSFP module
* Param UInt16 Register: register address to read.
* Param ref UInt16 value: store the value read from the QSFP module's register.

*/

public bool QSFP_GetRegister(int Module, UInt16 Register, ref UInt16 Value)

 {

return mlBert_QSFP_GetRegister (instance, Module, Register, ref Value);

}
```

mlBert_QSFP_SetRegister

This API call is used to set QSFP Register.
Returns true on success, false on failure.

Used for ML ODVT and ML4070-QSFP only.

Example of use:

```
/*
 * mlBert_QSFP_SetRegister EntryPoint inside the DLL allows setting a QSFP module register value.
*/

[DllImport(@"DLL/MLBert_API.dll", EntryPoint = " mlBert_QSFP_SetRegister")]

public static extern bool mlBert_QSFP_SetRegister (mlbertapi* instance, int Module, UInt16 Register,
UInt16 Value);

/*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Param int Module: Select QSFP module
* Param UInt16 Register: register address to read.
* Param UInt16 Value: the value of the register.

*/

public bool QSFP_SetRegister(int Module, UInt16 Register, UInt16 Value)

{

return mlBert_QSFP_SetRegister (instance, Module, Register, Value);

}
```

mlBert_SetQsfpI2cControlExternal

This API call is used to switch the QSFP I2C bus and control signal source. It can be controlled internally by the microcontroller or externally.
Returns true on success, false on failure.

Used for ML ODVT and ML4070-QSFP only.

Example of use:

```
/*
 * mlBert_SetQsfpI2cControlExternal EntryPoint inside the DLL allows selecting the QSFP I2C bus and
control signal source.
*/

[DllImport(@"DLL/MLBert_API.dll", EntryPoint = " mlBert_SetQsfpI2cControlExternal")]

public static extern bool mlBert_SetQsfpI2cControlExternal (mlbertapi* instance, bool IsExternal);

/*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Param bool IsExternal: if true the I2C bus and control signal are external.
*/

public bool SetQsfpI2cControlExternal(bool IsExternal)

{

return mlBert_SetQsfpI2cControlExternal (instance, IsExternal);

}
```

mlBert_SFP_Read_SlaveAddress

This API call is used to read SFP Slave Address.
Returns true on success, false on failure.

Used for ML4070SFP.

Example of use:

```
/*
* mlBert_SFP_Read_SlaveAddress EntryPoint inside the DLL allows reading the SFP Slave Address
 */

 [DllImport (@"DLL/MLBert_API.dll", EntryPoint = “mlBert_SFP_Read_SlaveAddress")]
private static extern bool mlBert_SFP_Read_SlaveAddress (mlbertapi* instance, int *SlaveAddress, int
module);

 /*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Param int *SlaveAddress: Pointer to the Slave address
* Param int module: select the module.
* Allows reading the SFP Slave Address
*/
Public bool SFP_ReadSlaveAddress (mlbertapi* instance, int * SlaveAddress, int module)

{
return mlBert_SFP_Read_SlaveAddress (instance, SlaveAddress, module);

}
```

mlBert_SFP_ Set_SlaveAddress

This API call is used to set SFP Slave Address.
Returns true on success, false on failure.

Used for ML4070-SFP.

Example of use:

```
/*
* mlBert_SFP_ Set_SlaveAddress EntryPoint inside the DLL allows setting the SFP Slave Address */

 [DllImport (@"DLL/MLBert_API.dll", EntryPoint = “mlBert_SFP_ Set_SlaveAddress")]
private static extern bool mlBert_SFP_ Set_SlaveAddress (mlbertapi* instance, int SlaveAddress, int
module);

 /*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Param int SlaveAddress: the Slave address
* Param int module: select the module.
* Allows setting the SFP Slave Address
*/
Public bool SFP_SetSlaveAddress (mlbertapi* instance, int SlaveAddress, int module)

{
return mlBert_SFP_ Set_SlaveAddress (instance, SlaveAddress, module);

}
```

mlBert SFP_ Set_Register

This API call is used to set SFP Register.
Returns true on success, false on failure.

Used for ML4070-SFP.

Example of use:

```
/*
* mlBert_SFP_ Set_Register EntryPoint inside the DLL allows setting the SFP Register */

 [DllImport (@"DLL/MLBert_API.dll", EntryPoint = “mlBert_SFP_ Set_Register")]
private static extern bool mlBert_SFP_ Set_Register (mlbertapi* instance, int module, UInt16 Register,
UInt16 Value);

 /*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Param int module: select the module.
* Param UInt16 Register: Register address
* Param UInt16 Value: Value for this register
* Allows setting the SFP Register
*/

Public bool SFP_ Set_Register (mlbertapi* instance, int module, UInt16 Register, UInt16 Value)

{
return mlBert_SFP_ Set_Register (instance, module, Register, Value);

}
```

mlBert_SFP_ Get_Register

This API call is used to get SFP Register.
Returns true on success, false on failure.

Used for ML4070-SFP.

Example of use:

```
/*
* mlBert_SFP_ Get_Register EntryPoint inside the DLL allows getting the SFP Register */

 [DllImport (@"DLL/MLBert_API.dll", EntryPoint = "mlBert_SFP_ Get_Register")]
private static extern bool mlBert_SFP_ Get_Register (mlbertapi* instance, int module, UInt16 Register,
UInt16 *Value);

 /*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Param int module: select the module.
* Param UInt16 Register: Register address
* Param UInt16 *Value: pointer Value for this register
* Allows getting the SFP Register
*/

Public bool SFP_ Get_Register (mlbertapi* instance, int module, UInt16 Register, UInt16 *Value)

{
return mlBert_SFP_ Get_Register (instance, module, Register, *Value);

}
```

mlBert_SFP_TX_FaultStatus

This API call is used to check SFP TX Fault status.
Returns true if a TX fault occurred, false otherwise.

Used for ML4070-SFP.

Example of use:

```
/*
* mlBert_SFP_TX_FaultStatus EntryPoint inside the DLL allows checking SFP TX Fault status */

 [DllImport (@"DLL/MLBert_API.dll", EntryPoint = “mlBert_SFP_TX_FaultStatus")]
private static extern bool mlBert_SFP_TX_FaultStatus (mlbertapi* instance, int module);

 /*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Param int module: select the module.
* Allows checking SFP TX Fault status
*/

Public bool SFP_TX_FaultStatus (mlbertapi* instance, int module)

{
return mlBert_SFP_TX_FaultStatus (instance, module);

}
```

mlBert_SFP_RX_LOSStatus

This API call is used to check SFP RX LOS status.
Returns true if RX Los occurred, false otherwise.

Used for ML4070-SFP.

Example of use:

```
/*
* mlBert_SFP_RX_LOSStatus EntryPoint inside the DLL allows checking SFP RX LOS status */

 [DllImport (@"DLL/MLBert_API.dll", EntryPoint = “mlBert_SFP_RX_LOSStatus")]
private static extern bool mlBert_SFP_RX_LOSStatus (mlbertapi* instance, int module);

 /*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Param int module: select the module.
* Allows checking SFP RX LOS status
*/

Public bool SFP_RX_LOSStatus (mlbertapi* instance, int module)

{
return mlBert_SFP_RX_LOSStatus (instance, module);

}
```

mlBert SFP Module Present

This API call is used to check SFP Module Present status.
Returns true if module is present, false otherwise.

Used for ML4070-SFP.

Example of use:

```
/*
* mlBert_SFP_Module_Present EntryPoint inside the DLL allows checking SFP Module Presentstatus */

 [DllImport (@"DLL/MLBert_API.dll", EntryPoint = "mlBert_SFP_Module_Present")]
private static extern bool mlBert_SFP_Module_Present (mlbertapi* instance, int module);

 /*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Param int module: select the module.
* Allows checking SFP Module Present status
*/

Public bool SFP_Module_Present (mlbertapi* instance, int module)

{
return mlBert_SFP_Module_Present (instance, module);

}
```

mlBert_SFP_TX_Disable

This API call is used to disable TX.
Returns true on success, false on failure.

Used for ML4070-SFP.

Example of use:

```
/*
* mlBert_SFP_TX_Disable EntryPoint inside the DLL allows disabling TX */

 [DllImport (@"DLL/MLBert_API.dll", EntryPoint = “mlBert_SFP_TX_Disable")]
private static extern bool mlBert_SFP_TX_Disable (mlbertapi* instance, int Mode, int module);

 /*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Param int module: select the module.
* Param int Mode: Mode = 0 to enable TX and 1 to Disable TX
* Allows disabling TX

*/

Public bool SFP_TX_Disable (mlbertapi* instance, int Mode, int module)

{
return mlBert_SFP_TX_Disable (instance, Mode, module);

}
```

mlBert_SFP_RS0_RateSelect

This API call is used to select SFP RS0 Mode.
Returns true on success, false on failure.

Used for ML4070-SFP.

Example of use:

```
/*
* mlBert_SFP_RS0_RateSelect EntryPoint inside the DLL allows selecting SFP RS0 Mode */

 [DllImport (@"DLL/MLBert_API.dll", EntryPoint = "mlBert_SFP_RS0_RateSelect")]
private static extern bool mlBert_SFP_RS0_RateSelect (mlbertapi* instance, int mode, int module);

 /*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Param int module: select the module.
* Param int mode: 1 high mode, 0 low mode
* Allows selecting SFP RS0 Mode

*/

Public bool SFP_RS0_RateSelect (mlbertapi* instance, int mode, int module)

{
return mlBert_SFP_RS0_RateSelect (instance, mode, module);

}
```

mlBert_SFP_RS1_RateSelect

This API call is used to select SFP RS1 Mode.
Returns true on success, false on failure.

Used for ML4070-SFP.

Example of use:

```
/*
* mlBert_SFP_RS1_RateSelect EntryPoint inside the DLL allows selecting SFP RS1 Mode */

 [DllImport (@"DLL/MLBert_API.dll", EntryPoint = “mlBert_SFP_RS1_RateSelect")]
private static extern bool mlBert_SFP_RS1_RateSelect (mlbertapi* instance, int mode);

 /*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Param int module: select the module.
* Param int mode: 1 high mode, 0 low mode
* Allows selecting SFP RS1 Mode

*/

Public bool SFP_RS1_RateSelect (mlbertapi* instance, int mode, int module)

{
return mlBert_SFP_RS1_RateSelect (instance, mode, module);

}
```

mlBert_SFP_Get_Current_sens

This API call is used to read SFP Module current level.
Returns true on success, false on failure.

Used for ML4070-SFP.

Example of use:

```
/*
* mlBert_SFP_Get_Current_sens EntryPoint inside the DLL allows reading SFP Module current level */

 [DllImport (@"DLL/MLBert_API.dll", EntryPoint = “mlBert_SFP_Get_Current_sens")]
private static extern bool mlBert_SFP_Get_Current_sens (mlbertapi* instance, double* Current, int
module);

 /*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Param int module: select the module.
* Param double* Current: Temperature Current
* Allows reading SFP Module current level

*/

Public bool SFP_Get_Current _sens (mlbertapi* instance, double* Current, int module)

{
return mlBert_SFP_Get_Current_sens (instance, Current, module);

}
```

mlBert_Power_SetVoltage

This API call is used to set LDO Voltage.
Returns true on success, false on failure.

Used for ML ODVT, ML4070-QSFP and ML4070-SFP.

Example of use:

```
/*
* mlBert_Power_SetVoltage EntryPoint inside the DLL allows setting the voltage level */

 [DllImport (@"DLL/MLBert_API.dll", EntryPoint = “mlBert_Power_SetVoltage ")]
private static extern bool mlBert_Power_SetVoltage (mlbertapi* instance, int module, double
VoltageLevel);

 /*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Param int module: select the module.
* Param double VoltageLevel: voltage level to apply (between 3.1 and 3.6)
* Allows setting the voltage level

*/

Public bool Power_SetVoltage (mlbertapi* instance, int module, double VoltageLevel)

{
return mlBert_Power_SetVoltage (instance, module, VoltageLevel);

}
```

mlBert_Power_EnableNoiseFilteringCapacitor

This API call is used to enable/disable the noise filtering capacitors causing excess noise to be added to the DC out.
Returns true on success, false on failure.

Used for ML ODVT, ML4070-QSFP and ML4070-SFP.

Example of use:

```
/*
* mlBert_Power_EnableNoiseFilteringCapacitor EntryPoint inside the DLL allows disabling the Noise Filtering Capacitor to add noise

*/

 [DllImport (@"DLL/MLBert_API.dll", EntryPoint = “mlBert_Power_EnableNoiseFilteringCapacitor")]
private static extern bool mlBert_Power_EnableNoiseFilteringCapacitor (mlbertapi* instance, bool IsNoiseEnabled);

 /*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Param bool IsNoiseEnabled: if true, LDOs output capacitors will be disconnected from ground causing excess noise to be added to the DC out, else they will be connected to the ground disabling the noise.

*/

Public bool Power_EnableNoiseFilteringCapacitor (mlbertapi* instance, bool IsNoiseEnabled);

{
return mlBert_Power_EnableNoiseFilteringCapacitor (instance, IsNoiseEnabled);

}
```

mlBert_Power_SetNoiseFrequency_Amplitude

This API call is used to set the noise frequency and amplitude.
Returns true on success, false on failure.

Used for ML ODVT, ML4070-QSFP and ML4070-SFP.

Example of use:

```
/*
* mlBert_Power_SetNoiseFrequency_Amplitude EntryPoint inside the DLL allows setting the noise
frequency and amplitude

 */

 [DllImport (@"DLL/MLBert_API.dll", EntryPoint = “mlBert_Power_SetNoiseFrequency_Amplitude ")]
private static extern bool mlBert_Power_SetNoiseFrequency_Amplitude (mlbertapi* instance, double
NoiseFrequency, byte Amplitude);

 /*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Param double Noise frequency: selected noise frequency
* Param byte amplitude: noise amplitude

*/

Public bool Power_SetNoiseFrequency_Amplitude (mlbertapi* instance, double NoiseFrequency, byte
Amplitude)

{
return mlBert_Power_SetNoiseFrequency_Amplitude (instance, NoiseFrequency, Amplitude);

}
```

mlBert_LineRateConfiguration_Polarise

This API call is used to set the Line Rate of the connected Polaris BERT. It returns 1 if successful, 0 if a problem occurs while setting the line Rate. For every Line Rate a File should be provided or generated that will configure the Silab, location of the file will be set by using "APIConfigureApplication". To generate the clock file the user should use a special GUI provided by MultiLane, or a C# API (ClockLibrary.dll) that generate clock files also provided by multilane.

Used for ML4039D and ML4079D.

Example of use:

```
mlBert_LineRateConfiguration_Polarise EntryPoint inside the DLL allows setting the line rate
configuration for Polaris BERT

 [DllImport (@"DLL/MLBert_API.dll", EntryPoint = "mlBert_LineRateConfiguration_Polarise")]
private static extern int mlBert_LineRateConfiguration_Polarise (mlbertapi* instance, double linerate,
EyeMode_ML4004 EyeMode, int GrayMapping, int PRECoding, int VGAtracking, int clockSource, int
ClockType, int divider, int FEC, int FECmode, int IEEEmode, int AFE, int is7taps);
* Public Interface to connect with DLL
* Param mlbertapi* instance

* Param double linerate
* Param EyeMode: (0= Pam4, 1= NRZ)
* Param int GrayMapping: 1 enable, 0 disable
* Param int PRECoding: 1 enable, 0 disable
* Param int VGAtracking: 1 enable, 0 disable
* Param int clockSource: 1 internal, 0 external
* Param int ClockType: 0 monitor clock, 1 external clock, 2 silab clockout
* Param int divider: 1, 4, 8,16,32,64,128
* Param int FEC: 1 enable, 0 disable
* Param int FECmode: 0 for PLR_LM_FEC_50G_KS4 , 1 for PLR_LM_FEC_50G_KR4, 2 for
PLR_LM_FEC_50G_KP4, 3 for PLR_LM_FEC_100G_KR4, 4 for PLR_LM_FEC_100G_KP4, 5 for
PLR_LM_FEC_200G_KP4, 6 for PLR_LM_FEC_400G_KP4, 7 for PLR_LM_FEC_USER_DEF
* Param int IEEEmode: 1 to set IEEE802.3bs DEMAP, 0 to disable it
* Param int AFE: 0 for 4db, 1 for 16db
* Param int is7taps: 1 to enable 7 taps, 0 to disable it
*/
Public int LineRateConfiguration_Polarise (mlbertapi* instance, double linerate, EyeMode_ML4004
EyeMode, int GrayMapping, int PRECoding, int VGAtracking, int clockSource, int ClockType, int divider,
int FEC, int FECmode, int IEEEmode, int AFE, int is7taps) {
return mlBert_LineRateConfiguration_Polarise (instance, linerate, EyeMode, GrayMapping, PRECoding,
VGAtracking, clockSource, ClockType, divider, FEC, FECmode, IEEEmode, AFE , is7taps; }
```

mlBert_SetTxUserPatternPolaris

This API call is used to set a user defined PRBS to be generated by the selected channel.
mlBert_SetPRBSPattern should be set to 14 “user Defined” in order for this API to work.
Each channel should have a separate call.

API returns 1 upon success, 0 if failure.

Used for ML4039D and ML4079D.

Example of Use:

```
/*
*  mlBert_SetTxUserPatternPolaris EntryPoint inside the DLL allows to set user Defined Pattern to be
generated
*/

[DllImport(@"DLL/MLBert_API.dll", EntryPoint = " mlBert_SetTxUserPatternPolaris ")]
private static extern int mlBert_SetTxUserPatternPolaris (mlbertapi* instance, int channel, ulong
UserDefinedPatternLo, ulong UserDefinedPatternMid, int repetition1, int repetition2);

/*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Param int channel
* Param ulong UserDefinedPatternLo: the length must be 16 ex: 0000000000111111
* Param ulong UserDefinedPatternMid: the length must be 16 ex: 0000000000111111
* Param int repetition1: number of repetition for UserDefinedPatternLo (from 0 to 255)
* Param int repetition2: number of repetition for UserDefinedPatternMid (from 0 to 255)
* APISetPRBSPattern should be set to (user Defined = 14) in order for this API to work
* Returns 1 on success, 0 on failure.
* Channels are identified using the channel parameter
* Boards connected are identified using the instance parameter
*/

public int SetTxUserPatternPolaris (mlbertapi* instance, int channel, ulong UserDefinedPatternLo, ulong
UserDefinedPatternMid, int repetition1, int repetition2)
 {
   return mlBert_SetTxUserPatternPolaris (instance, channel, UserDefinedPatternLo,
UserDefinedPatternMid, repetition1, repetition2);
 }
```

mlBert_ErrorInsertionPolarise

This API call is used to insert specific count of errors into a signal generated on a TX channel.

Each channel should have a separate call.

Return 1 on success, 0 on failure.

Used for ML4039D and ML4079D.

Example of use:

```
/*
* mlBert_ErrorInsertionPolarise EntryPoint inside the DLL allows to insert specific count of errors into a
signal generated on a TX channel
*/

[DllImport(@"DLL/MLBert_API.dll", EntryPoint = " mlBert_ErrorInsertionPolarise")]
 static extern int mlBert_ErrorInsertionPolarise (mlbertapi* instance, int channel, int enable, int gap, int
mode, int duration);

/*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Param int channel
* Param int enable: 1 to enable, 0 to disable
* Param int gap: number of bits without error between words
*Param mode: error bit sequence in one word: 0 for 0x0000_0000_0000_0001, 1 for
0x0000_0000_0000_0002, 2 for 0x0000_0000_0000_0003, 3 for 0x5555_5555_5555_5555, 4 for
0xAAAA_AAAA_AAAA_AAAA, 5 for 0xFFFF_FFFF_FFFF_FFFF, 6 for One bit per word.
* Param int duration: Number of 64-bit words to inject errors on. A value of 1 injects errors on one
word. A value of 127 injects errors continuously
* Return 1 on success, 0 on failure.
*/

public int ErrorInsertionPolarise(mlbertapi* instance, int channel, int enable, int gap, int mode, int
duration)
 {
return mlBert_ErrorInsertionPolarise (instance, channel, enable, gap, mode, duration);
 }
```

mlBert_Check4039PolariseRev

This API call is used to check if the board is ML4039D or ML4079D.

Return 1 on success, 0 on failure.

Used for ML4039D and ML4079D.

Example of use:

```
/*
* mlBert_Check4039PolariseRev EntryPoint inside the DLL allows to check if the board is ML4039D or
ML4079D
*/

[DllImport(@"DLL/MLBert_API.dll", EntryPoint = " mlBert_Check4039PolariseRev")]
 static extern int mlBert_Check4039PolariseRev (mlbertapi* instance, int *PolariseRevision);
/*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Param int *PolariseRevision: 0 for ML4039D, 1 for ML4079D
* Return 1 on success, 0 on failure.
*/

public int Check4039PolariseRev(mlbertapi* instance, int *PolariseRevision)
 {
return mlBert_Check4039PolariseRev (instance, *PolariseRevision);

 }
```

mlBert_MainTap

This API call is used to set the main tap value.

Return 1 on success, 0 on failure.

Used for ML4039D and ML4079D.

Example of use:

```
/*
* mlBert_MainTap EntryPoint inside the DLL allows to set the main tap value
*/

[DllImport(@"DLL/MLBert_API.dll", EntryPoint = " mlBert_MainTap")]
 static extern int mlBert_MainTap (mlbertapi* instance, int channel, int value, int mode);
/*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Param int channel
* Param int value: from -1000 to 1000
* Param int mode: 0 for low rate, 1 for high rate NRZ and 2 for high rate PAM
* Return 1 on success, 0 on failure.
*/

public int MainTap (mlbertapi* instance, int channel, int value, int mode)

 {
return mlBert_MainTap (instance, channel, value, mode);

 }
```

mlBert_OutterEyeLevel

This API call is used to set the outter eye value.

Return 1 on success, 0 on failure.

Used for ML4039D and ML4079D.

Example of use:

```
/*
* mlBert_OutterEyeLevel EntryPoint inside the DLL allows to set the outter eye value
*/

[DllImport(@"DLL/MLBert_API.dll", EntryPoint = " mlBert_OutterEyeLevel")]
 static extern int mlBert_OutterEyeLevel (mlbertapi* instance, int channel, int value);
/*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Param int channel
* Param int value: from 1500 to 2500
* Return 1 on success, 0 on failure.
*/

public int OutterEyeLevel (mlbertapi* instance, int channel, int value)

 {
return mlBert_OutterEyeLevel (instance, channel, value);

 }
```

mlBert_VPeakEnable

This API call is used to enable Vpeak.

Return 1 on success, 0 on failure.

Used for ML4039D and ML4079D.

Example of use:

```
/*
* mlBert_VPeakEnableEntryPoint inside the DLL allows to enable Vpeak
*/

[DllImport(@"DLL/MLBert_API.dll", EntryPoint = " mlBert_VPeakEnable")]
 static extern int mlBert_VPeakEnable (mlbertapi* instance, int channel, int status);


/*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Param int channel
* Param int status: 1 for enable, 0 for disable
* Return 1 on success, 0 on failure.
*/

public int VPeakEnable (mlbertapi* instance, int channel, int status)

 {
return mlBert_VPeakEnable (instance, channel, status);

 }
```

mlBert_LineRateConfiguration_Porrima

This API call is used to set the Line Rate of the connected Porrima BERT. It returns 1 if successful, 0 if a problem occurs while setting the line Rate. For every Line Rate a File should be provided or generated that will configure the Silab, location of the file will be set by using “APIConfigureApplication”. To generate the clock file the user should use a special GUI provided by MultiLane, or a C# API (ClockLibrary.dll) that generate clock files also provided by multilane.

Return 1 on success, 0 on failure.

Used for ML4039E.

Example of use:

```
/*
* mlBert_LineRateConfiguration_Porrima EntryPoint inside the DLL allows to enable Vpeak
*/

[DllImport(@"DLL/MLBert_API.dll", EntryPoint = " mlBert_LineRateConfiguration_Porrima ")]
static extern int mlBert_LineRateConfiguration_Porrima (mlbertapi* instance, double linerate,
EyeMode_ML4004 EyeMode, int GrayMapping, int PreCoding, int ChipMode, int clockSource, int
ClockType, int divider,     int FEC, int FECmode, int IEEEmode);
/*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Param double linerate
* Param EyeMode_ML4004 EyeMode
* Param int GrayMapping
* Param int PreCoding
* Param int ChipMode
* Param int clockSource
* Param int ClockType
* Param int divider
* Param int FEC
* Param int FECmode
* Param int IEEEmode
* Return 1 on success, 0 on failure.
*/

public int LineRateConfiguration_Porrima (mlbertapi* instance, double linerate, EyeMode_ML4004
EyeMode, int GrayMapping, int PreCoding, int ChipMode, int clockSource, int ClockType, int divider,
int FEC, int FECmode, int IEEEmode) {
return mlBert_LineRateConfiguration_Porrima (instance, linerate, EyeMode, GrayMapping, PreCoding,
ChipMode, clockSource, ClockType, divider, FEC, FECmode, IEEEmode); }
```

mlBert_QDD_MODRPS

This API call is used to check if the module is present.
Returns true on success, false on failure.

Used for ML4070-QDD only.

Example of use:

```
/*
 * mlBert_QDD_MODRPS EntryPoint inside the DLL allows reading the QDD module present status.
*/

[DllImport(@"DLL/MLBert_API.dll", EntryPoint = " mlBert_QDD_MODRPS")]
public static extern bool mlBert_QDD_MODRPS (mlbertapi* instance, int Module, ref bool status);

/*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Param int Module: Select QDD module
* Param ref bool status: true if the module is present, false otherwise.

*/

public bool QDD_MODPRS(int Module, ref bool status)

{

 return mlBert_QDD_MODRPS (instance, Module, ref status);

}
```

mlBert_QDD_LPMODE

This API call is used to set or release the module from the Low Power mode.
Returns true on success, false on failure.

Used for ML4070-QDD only.

Example of use:

```
/*
 * mlBert_QDD_LPMODE EntryPoint inside the DLL allows setting or releasing the QDD module from
Low Power mode.
*/
[DllImport(@"DLL/MLBert_API.dll", EntryPoint = " mlBert_QDD_LPMODE ")]
public static extern bool mlBert_QDD_LPMODE (mlbertapi* instance, int Module, bool asserted);

/*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Param int Module: Select QDD module
* Param bool asserted: if true set the module to the Low Power mode, else release it from the Low
Power mode.

*/

public bool QDD_LPMODE(int Module, bool asserted)

{

 return mlBert_QDD_LPMODE (instance, Module, asserted);

}
```

mlBert_QDD_RESET

This API call is used to set/release the QDD module from reset mode.
Setting the module to the reset mode for longer than the minimum pulse length initiates a complete module reset, returning all user module settings to their default state.
Returns true on success, false on failure.

Used for ML4070-QDD only.

Example of use:

```
/*
 * mlBert_QDD_RESET EntryPoint inside the DLL allows setting/releasing the QDD reset pin.
*/

[DllImport(@"DLL/MLBert_API.dll", EntryPoint = " mlBert_QDD_RESET ")]

public static extern bool mlBert_QDD_RESET (mlbertapi* instance, int Module, bool asserted);

/*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Param int Module: Select QDD module
* Param bool asserted: set if true, clear if false.

*/

public bool QDD_RESET(int Module, bool asserted)

{

return mlBert_QDD_RESET (instance, Module, asserted);

}
```

mlBert_QDD_MODSEL

This API call is used to select the module for the I2C communication.
Returns true on success, false on failure.

Used for ML4070-QDD only.

Example of use:

```
/*
 * mlBert_QDD_MODSEL EntryPoint inside the DLL allows selecting the QDD module for I2C
communication.
*/

[DllImport(@"DLL/MLBert_API.dll", EntryPoint = "mlBert_QDD_MODSEL ")]

public static extern bool mlBert_QDD_MODSEL (mlbertapi* instance, int Module, bool asserted);

/*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Param int Module: Select QDD module
* Param bool asserted: select the module if true, unselect if false.

*/

public bool QDD_MODSEL(int Module, bool asserted)

{

 return mlBert_QDD_MODSEL (instance, Module, asserted);

}
```

mlBert_QDD_IntL

This API call is used to read module interrupt signal.
Returns true on success, false on failure.

Used for ML4070-QDD only.

Example of use:

```
/*
 * mlBert_QDD_IntL EntryPoint inside the DLL allows reading QDD the module interrupt signal.
*/

[DllImport(@"DLL/MLBert_API.dll", EntryPoint = " mlBert_QDD_IntL ")]

public static extern bool mlBert_QDD_IntL (mlbertapi* instance, int Module, ref bool status);

/*
* Public Interface to connect with DLL
* Param mlbertapi* instance

* Param int Module: Select QDD module

* Param ref bool status: represent the status of the interrupt signal

*/

public bool QDD_IntL(int Module, ref bool status)

{

return mlBert_QDD_IntL (instance, Module, ref status);

 }
```

mlBert_QDD_Get_VCCTX

This API call is used to read QDD Voltage measurement of TX power rail.
Returns true on success, false on failure.

Used for ML4070-QDD only.

Example of use:

```
/*
 * mlBert_QDD_Get_VCCTX EntryPoint inside the DLL allows reading QDD TX Voltage measurement
*/
[DllImport(@"DLL/MLBert_API.dll", EntryPoint = " mlBert_QDD_Get_VCCTX ")]
public static extern bool mlBert_QDD_Get_VCCTX (mlbertapi* instance, int Module, ref double Data);
/*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Param int Module: Select QDD module
* Param ref double Data: to store the voltage

*/

public bool QDD_Get_VCCTX(int Module, ref double Data)

{

return mlBert_QDD_Get_VCCTX (instance, Module, ref Data);

 }
```

mlBert_QDD_Get_VCCRX

This API call is used to read QDD Voltage measurement of RX power rail.
Returns true on success, false on failure.

Used for ML4070-QDD only.

Example of use:

```
/*
 * mlBert_QDD_Get_VCCRX EntryPoint inside the DLL allows reading QDD RX Voltage measurement
*/
[DllImport(@"DLL/MLBert_API.dll", EntryPoint = " mlBert_QDD_Get_VCCRX ")]
public static extern bool mlBert_QDD_Get_VCCRX (mlbertapi* instance, int Module, ref double Data);

/*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Param int Module: Select QDD module
* Param ref double Data: to store the voltage

* The read value should be divided by 256 to get the RX voltage

*/

public bool QDD_Get_VCCRX(int Module, ref double Data)

{

 return mlBert_QDD_Get_VCCRX (instance, Module, ref Data);

}
```

mlBert_QDD_GetVCC

This API call is used to read QDD Voltage measurement of VCC power rail.
Returns true on success, false on failure.

Used for ML4070-QDD only.

Example of use:

```
/*
* mlBert_QDD_GetVCC EntryPoint inside the DLL allows reading QDD VCC Voltage measurement
*/
[DllImport(@"DLL/MLBert_API.dll", EntryPoint = " mlBert_QDD_GetVCC ")]
public static extern bool mlBert_QDD_GetVCC (mlbertapi* instance, int Module, ref double Data);

/*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Param int Module: Select QDD module
* Param ref double Data: to store the voltage

*/

public bool QDD_GetVCC(int Module, ref double Data)

{

return mlBert_QDD_GetVCC (instance, Module, ref Data);

}
```

mlBert_QDD_GetVolt_sens

This API call is used to read QDD Module Voltage level.
Returns true on success, false on failure.

Used for ML4070-QDD only.

Example of use:

```
/*
* mlBert_QDD_GetVolt_sens EntryPoint inside the DLL allows reading QDD Module Voltage level */

 [DllImport (@"DLL/MLBert_API.dll", EntryPoint = “mlBert_QDD_GetVolt_sens")]
private static extern bool mlBert_QDD_GetVolt_sens (mlbertapi* instance, ref double Voltage, int
module);

 /*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Param ref double Voltage: Temperature Voltage
* Param int module
* Allows reading QDD Module Voltage level

*/

Public bool QDD_GetVolt_sens (mlbertapi* instance, ref double Voltage, int module)

{
return mlBert_QDD_GetVolt_sens (instance, ref Voltage, module);

}
```

mlBert_QDD_ReadI2C

This API call is used to read from QDD MSA registers.
Returns true on success, false on failure.

Used for ML4070-QDD only.

Example of use:

```
/*
 * mlBert_QDD_ReadI2C EntryPoint inside the DLL allows reading QDD I2C Data.
*/

[DllImport(@"DLL/MLBert_API.dll", EntryPoint = " mlBert_QDD_ReadI2C ")]

public static extern bool mlBert_QDD_ReadI2C (mlbertapi* instance, int Address, ref double Data);

/*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Param int Address
* Param ref double Data
*/

public bool QDD_ReadI2C(mlbertapi* instance, int Address, ref double Data)

{

return mlBert_QDD_ReadI2C (instance, Address, ref Data);

}
```

mlBert_QDD_WriteI2C

This API call is used to write data into QDD MSA registers
Returns true on success, false on failure.

Used for ML4070-QDD only.

Example of use:

```
/*
 * mlBert_QDD_WriteI2C EntryPoint inside the DLL allows writing QDD I2C Data.
*/

[DllImport(@"DLL/MLBert_API.dll", EntryPoint = " mlBert_QDD_WriteI2C ")]

public static extern bool mlBert_QDD_WriteI2C (mlbertapi* instance, int Address, double Data);

/*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Param int Address
* Param double Data
*/

public bool QDD_WriteI2C(mlbertapi* instance, int Address, double Data)

{

return mlBert_QDD_WriteI2C (instance, Address, Data);

}
```

mlBert_WriteValueToEEPROM

This API call is used to write value into an EEPROM

API returns true upon success, false if failure.

Example of Use:

```
/*
* mlBert_WriteValueToEEPROM EntryPoint inside the DLL allows to write value into an EEPROM */

[DllImport(@"DLL/MLBert_API.dll", EntryPoint = " mlBert_WriteValueToEEPROM ")]
private static extern bool mlBert_WriteValueToEEPROM (mlbertapi* instance, UINT16 Address, UINT16
Data);

/*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Param int Address
* Param int Data

*/

public bool WriteValueToEEPROM (mlbertapi* instance, UINT16 Address, UINT16 Data)
 {
  return mlBert_WriteValueToEEPROM (instance, Address, Data);
 }
```

mlBert_ReadValueFromEEPROM

This API call is used to read value from EEPROM

API returns true upon success, false if failure.

Example of Use:

```
/*
* mlBert_ReadValueFromEEPROM EntryPoint inside the DLL allows to read value from EEPROM */

[DllImport(@"DLL/MLBert_API.dll", EntryPoint = " mlBert_ReadValueFromEEPROM ")]
private static extern bool mlBert_ReadValueFromEEPROM (mlbertapi* instance, UINT16 Address, ref
UINT16 Data);

/*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Param int Address
* Param ref int Data

*/

public bool ReadValueFromEEPROM (mlbertapi* instance, UINT16 Address, ref UINT16 Data)
 {
   return mlBert_ReadValueFromEEPROM (instance, Address, ref Data);
 }
```

mlBert_LineRateConfiguration_Porrima

This API call is used to set the Line Rate of the connected Porrima BERT. It returns 1 if successful, 0 if a problem occurs while setting the line Rate. For every Line Rate a File should be provided or generated that will configure the Silab, location of the file will be set by using "mlBert_ConfigureApplication". To generate the clock file the user should use a special GUI provided by MultiLane, or a C# API (ClockLibrary.dll) that generate clock files also provided by multilane.

Used for ML4039E and AT4039E.

Example of use:

```
mlBert_LineRateConfiguration_Porrima EntryPoint inside the DLL allows setting the line rate
configuration for Porrima BERT

 [DllImport (@"DLL/MLBert_API.dll", EntryPoint = "mlBert_LineRateConfiguration_Porrima")]
private static extern int mlBert_LineRateConfiguration_Porrima (mlbertapi* instance,double linerate,
EyeMode_ML4004 EyeMode, int GrayMapping, int PRECoding, int ChipMode, int clockSource, int
ClockType, int divider, int FEC, int FECmode, int IEEEmode, int alltaps);
* Public Interface to connect with DLL
* Param mlbertapi* instance

* Param double linerate
* Param EyeMode: (0= Pam4, 1= NRZ)
* Param int GrayMapping: 1 enable, 0 disable
* Param int PRECoding: 1 enable, 0 disable
* Param int ChipMode: 1 for 53G, 0 for 26G
* Param int clockSource: 1 internal, 0 external
* Param int ClockType: 0 monitor clock, 1 external clock, 2 silab clock out
* Param int divider: 128
* Param int FEC: 1 enable, 0 disable
* Param int FECmode: 0 for PLR_LM_FEC_50G_KS4 , 1 for PLR_LM_FEC_50G_KR4, 2 for
PLR_LM_FEC_50G_KP4, 3 for PLR_LM_FEC_100G_KR4, 4 for PLR_LM_FEC_100G_KP4, 5 for
PLR_LM_FEC_200G_KP4, 6 for PLR_LM_FEC_400G_KP4, 7 for PLR_LM_FEC_USER_DEF
* Param int IEEEmode: 1 to set IEEE802.3bs DEMAP, 0 to disable it
* Param int alltaps: 1 to set 7 taps, 0 to use specific taps
*/

Public int LineRateConfiguration_Porrima(mlbertapi* instance,double linerate, EyeMode_ML4004
EyeMode, int GrayMapping, int PRECoding, int ChipMode, int clockSource, int ClockType, int divider, int
FEC, int FECmode, int IEEEmode, int alltaps){
return mlBert_LineRateConfiguration_Porrima (instance, linerate, EyeMode, GrayMapping, PRECoding,
ChipMode, clockSource, ClockType, divider, FEC, FECmode, IEEEmode, alltaps; }
```

mlBert_LineRateConfiguration_Vega

This API call is used to set the Line Rate of the connected Vega BERT. It returns 1 if successful, 0 if a problem occurs while setting the line Rate. For every Line Rate a File should be provided or generated that will configure the Silab, location of the file will be set by using "mlBert_ConfigureApplication". To generate the clock file the user should use a special GUI provided by MultiLane, or a C# API (ClockLibrary.dll) that generate clock files also provided by multilane.

Used for ML4039B and ML4054-400.

Example of use:

```
mlBert_LineRateConfiguration_Vega EntryPoint inside the DLL allows setting the line rate configuration
for Vega BERT

 [DllImport (@"DLL/MLBert_API.dll", EntryPoint = "mlBert_LineRateConfiguration_Vega")]
private static extern int mlBert_LineRateConfiguration_Vega (mlbertapi* instance, double linerate, int
clockSource, EyeMode_ML4004 EyeMode, int PRECoding, int GrayMapping, int FECEnable, int FECType,
int side, int Taps);
* Public Interface to connect with DLL
* Param mlbertapi* instance

* Param double linerate
* Param int clockSource: 1 internal, 0 external
* Param EyeMode: (0= Pam4, 1= NRZ)
* Param int GrayMapping: 1 enable, 0 disable
* Param int PRECoding: 1 enable, 0 disable
* Param int FEC: 1 enable, 0 disable
* Param int FECmode: 0 for RS544, 1 for RS528, 2 FIRECODE
* Param int side: 1 to enable host side mode
* Param int Taps: 1 to set 7 taps, 0 to use specific taps
*/

Public int LineRateConfiguration_Vega (mlbertapi* instance, double linerate, int clockSource,
EyeMode_ML4004 EyeMode, int PRECoding, int GrayMapping, int FECEnable, int FECType, int side, int
Taps){
return mlBert_LineRateConfiguration_Vega (instance, linerate, clockSource, EyeMode, PRECoding,
GrayMapping, FECEnable, FECType, side, Taps); }
```

mlBert_SetTxUserPatternVega

This API call is used to set a user defined PRBS to be generated by the selected channel.
mlBert_SetPRBSPattern should be set to 15 “user Defined” in order for this API to work.
Each channel should have a separate call.

API returns 1 upon success, 0 if failure.

Used for ML4039B and ML4054-400.

Example of use:

```
mlBert_SetTxUserPatternVega EntryPoint inside the DLL allows setting the user defined for Vega BERT

 [DllImport (@"DLL/MLBert_API.dll", EntryPoint = “mlBert_SetTxUserPatternVega")]
private static extern int mlBert_SetTxUserPatternVega (mlbertapi* instance, int channel, unsigned long
UserDefinedPatternLo, unsigned long UserDefinedPatternMid);
* Public Interface to connect with DLL
* Param mlbertapi* instance

* Param int channel
* Param long UserDefinedPatternLo
* Param long UserDefinedPatternMid
*/

Public int SetTxUserPatternVega (mlbertapi* instance, int channel, unsigned long UserDefinedPatternLo,
unsigned long UserDefinedPatternMid){
return mlBert_SetTxUserPatternVega (instance, channel, UserDefinedPatternLo,
UserDefinedPatternMid); } GrayMapping, FECEnable, FECType, side, Taps); }
```

mlBert_ReadTXLock

This API call is used to check TX lock (TX is transmitting a signal from the chip). Each channel should have a separate call.

Return true upon success and false if failed.

Used for ML4039B, ML4079D and ML4054-400.

Example of use:

```
/*
* mlBert_ReadTXLock EntryPoint inside the DLL responsible to verify the TX Lock for each channel
*/

 [DllImport(@"DLL/MLBert_API.dll", EntryPoint = " mlBert_ReadTXLock")]
 static extern int mlBert_ReadTXLock (mlbertapi* instance, int channel, ref bool status);

/*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Param int channel
* Param ref status indicates the status of TX
* return bool
* Channels are identified using the channel parameter
* Boards connected are identified using the instance parameter
*/

public int ReadTXLock(mlbertapi* instance, int channel, ref bool status)
 {
   return mlBert_ReadTXLock (instance, channel, ref status);
 }
```

mlBert_ClockOut3b

This API call is used to specify the clock out for ML4003B.

Returns 1 on success, 0 on failure.

Used for ML4003B.

Example of use:

```
/*
* mlBert_ClockOut3b EntryPoint inside the DLL allows to specify the clock out for ML4003B
*/

 [DllImport(@"DLL/MLBert_API.dll", EntryPoint = " mlBert_ClockOut3b")]
private static extern int mlBert_ClockOut3b (mlbertapi* instance, int clockIndex, int
ReferenceClockOutRate);

/*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Param int clockIndex: clock index
* Param int ReferenceClockOutRate:
* Returns 1 on success, 0 on failure
*/

Public int ClockOut3b (mlbertapi* instance, int clockIndex, int ReferenceClockOutRate)
{
return mlBert_ClockOut3b (instance, clockIndex, ReferenceClockOutRate);
}
```

mlBert_ClockOut4054

This API call is used to specify clock out for ML4054-400

Returns 1 on success, 0 on failure.

Used for ML4054-400

Example of use:

```
/*
* mlBert_ClockOut4054 EntryPoint inside the DLL allows to specify the clock out for ML4054-400
*/

 [DllImport(@"DLL/MLBert_API.dll", EntryPoint = " mlBert_ClockOut4054")]
private static extern int mlBert_ClockOut4054 (mlbertapi* instance, int TriggerOut, int MonitorSetting,
int CDRChannel, int CDRRate);

/*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Param int TriggerOut:
            0 = refernce clock QSFP1
            1 = refernce clock QSFP2
            2 = Monitor 1
            3 = Monitor 2
* Param int MonitorSetting:
            0 = LPCDR A
            1 = TX PLL
            2= CDR
* Param int CDRChannel: CDR channel or TX clock Pll channel from 0 to 7
* Param int CDRRate:
            0 = Datarate/8
            1 = Datarate/16
            2 = Datarate/32
            3 = Datarate/64
            4 = Datarate/128
            5 = Datarate/256
* Returns 1 on success, 0 on failure
*/
Public int ClockOut4054 (mlbertapi* instance, int TriggerOut, int MonitorSetting, int CDRChannel, int
CDRRate)
{
return mlBert_ClockOut4054 (instance, TriggerOut, MonitorSetting, CDRChannel, CDRRate);
}
```

mlBert_ClockOut4644

This API call is used to specify clock out for ML4039B VEGA.

Returns 1 on success, 0 on failure.

Used for ML4039B VEGA

Example of use:

```
/*
* mlBert_ClockOut4644 EntryPoint inside the DLL allows to specify the clock out for ML4039B VEGA
*/

 [DllImport(@"DLL/MLBert_API.dll", EntryPoint = " mlBert_ClockOut4644")]
private static extern int mlBert_ClockOut4644 (mlbertapi* instance, int TriggerOut, int CDRRate);

/*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Param int TriggerOut: 0 -> monitor clk, 1-> external clk, 2-> reference clk
* Param int CDRRate: 32, 64,128,256,512,1024,2048,4096
* Returns 1 on success, 0 on failure
*/

Public int ClockOut4644 (mlbertapi* instance, int TriggerOut, int CDRRate)
{
return mlBert_ClockOut4644 (instance, TriggerOut, CDRRate);
}
```

mlBert_ClockOut4070

This API call is used to specify clock out for ML4070.

Returns 1 on success, 0 on failure.

Used for ML4070

Example of use:

```
/*
* mlBert_ClockOut4070 EntryPoint inside the DLL allows to specify the clock out for ML4070
*/

 [DllImport(@"DLL/MLBert_API.dll", EntryPoint = " mlBert_ClockOut4070")]
private static extern int mlBert_ClockOut4070 (mlbertapi* instance, int TriggerOut, int CDRChannel, int
Rate);

/*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Param int TriggerOut: 0 = refernce clock QSFPA
                        1 = refernce clock QSFPC
                        2 = Monitor GB2 A
                        3 = Monitor GB2 C
* Param int CDRChannel:
* Param int Rate:
* Returns 1 on success, 0 on failure
*/

Public int ClockOut4070 (mlbertapi* instance, int TriggerOut, int CDRChannel, int Rate)
{
return mlBert_ClockOut4070 (instance, TriggerOut, CDRChannel, Rate);
}
```

mlBert_ReadAvogoDividerCalibration

This API call is used to read the divider which configure the Avogo Chip for a specified Line Rate and Channel.

Returns the divider value

Used for ML4439 and ML 4839

Example of use:

```
/*
* mlBert_ReadAvogoDividerCalibration EntryPoint inside the DLL allows to read the divider that
configures the Avogo Chip for a specified line rate and channel
*/

 [DllImport(@"DLL/MLBert_API.dll", EntryPoint = " mlBert_ReadAvogoDividerCalibration")]
private static extern int mlBert_ReadAvogoDividerCalibration (mlbertapi* instance, int channel, double
LineRate);

/*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Param int channel: channel number
* Param double LineRate: line rate
* Returns the divider value
*/

Public int ReadAvogoDividerCalibration (mlbertapi* instance, int channel, double LineRate)
{
return mlBert_ReadAvogoDividerCalibration (instance, channel, LineRate);
}
```

mlBert_LineRateConfigurationBert3B

This API call is used to configure board line rate.

Returns 1 on success, 0 on failure.

Used for ML4003B

Example of use:

```
/*
* mlBert_LineRateConfigurationBert3B EntryPoint inside the DLL allows to configure the board line rate
*/

 [DllImport(@"DLL/MLBert_API.dll", EntryPoint = " mlBert_LineRateConfigurationBert3B")]
private static extern int mlBert_LineRateConfigurationBert3B (mlbertapi* instance, double LineRate, int
clockSource);

/*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Param double LineRate: line rate required in Gbps ranging from 8.5-15 and from 21-28.9
            - The clock file should be included in the specified directory in
                    ConfigureApplication:saveConfig parameter (default: clk)
            - configuration will be lost after applying line rate.
                    Configuration can be restored automatically using RestoreAllConfig() function
* Param int clockSource: 1 for internal and 0 for external and the default value is 0
* Returns 1 on success, 0 on failure
*/

Public int LineRateConfigurationBert3B (mlbertapi* instance, double LineRate, int clockSource)
{
return mlBert_LineRateConfigurationBert3B (instance, LineRate, clockSource);
}
```

mlBert_TDALineRateConfigurationBert3B

This API call is used to configure board line rate

Returns 1 on success, 0 on failure.

Used for ML4003B and ML4003BX

Example of use:

```
/*
* mlBert_TDALineRateConfigurationBert3B EntryPoint inside the DLL allows to configure board line rate
*/

 [DllImport(@"DLL/MLBert_API.dll", EntryPoint = " mlBert_TDALineRateConfigurationBert3B")]
private static extern int mlBert_TDALineRateConfigurationBert3B (mlbertapi* instance, double LineRate,
int clockSource, double *Fin_Bert, double *Fout_Bert, double *Fin_DSO, double *Fout_DSO);

/*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Param double LineRate: line rate
* Param int clockSource: 1 for internal and 0 for external clock
* Param *Fin_Bert: input frequency for BERT
* Param *Fout_Bert: output frequency for BERT
* Param *Fin_DSO: input frequency for DSO
* Param *Fout_DSO: output frequency for DSO
* Returns 1 on success, 0 on failure
*/

Public int TDALineRateConfigurationBert3B (mlbertapi* instance, double LineRate, int clockSource,
double *Fin_Bert, double *Fout_Bert, double *Fin_DSO, double *Fout_DSO)
{
return mlBert_TDALineRateConfigurationBert3B (instance, LineRate, clockSource, ref Fin_Bert, ref
Fout_Bert, ref Fin_DSO, ref Fout_DSO);
}
```

mlBert_LineRateConfiguration_4054

This API call is used to configure board line rate for ML4054-400

Returns 1 on success, 0 on failure

Used for ML4054-400

Example of use:

```
/*
* mlBert_LineRateConfiguration_4054 EntryPoint inside the DLL allows to configure board line rate for
ML4054
*/

 [DllImport(@"DLL/MLBert_API.dll", EntryPoint = " mlBert_LineRateConfiguration_4054")]
private static extern int mlBert_LineRateConfiguration_4054 (mlbertapi* instance, double LineRate1,
double LineRate2);

/*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Param double LineRate1:
        Contains the line rate(for channel 0 to 3)required in Gbps ranging from 1-30.2
        - The clock file should be included in the specified directory in
           ConfigureApplication:saveConfig parameter (default:clk)
        - Configuration will be lost after applying Linerate
* Param double LineRate2:
        Contains the line rate(for channel 4 to 7) required in Gbps ranging from 1-30.2
        - The clock file should be included in the specified directory in
          ConfigureApplication:saveConfig parameter (default:clk)
        - Configuration will be lost after applying Linerate
* Returns on 1 success, 0 on failure
*/

Public int LineRateConfiguration_4054 (mlbertapi* instance, double LineRate1, double LineRate2)
{
return mlBert_LineRateConfiguration_4054 (instance, LineRate1, LineRate2);
}
```

mlBert_SetTxUserPatternPAM

This API call is used to set user defined pattern for PAM.

Returns 1 on success, 0 on failure

Used for ML4004-PAM

Example of use:

```
/*
* mlBert_SetTxUserPatternPAM EntryPoint inside the DLL allows to set user defined pattern for PAM.
*/

 [DllImport(@"DLL/MLBert_API.dll", EntryPoint = " mlBert_SetTxUserPatternPAM")]
private static extern int mlBert_SetTxUserPatternPAM (mlbertapi* instance, int channel, unsigned long
long UserDefinedPatternLo, unsigned long long UserDefinedPatternMid, unsigned long long
UserDefinedPatternHi);

/*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Param int channel:
* Param unsigned long long UserDefinedPatternLo:
* Param unsigned long long UserDefinedPatternMid:
* Param unsigned long long UserDefinedPatternHi:
* Returns 1 on success, 0 on failure
*/

Public int SetTxUserPatternPAM (mlbertapi* instance, int channel, unsigned long long
UserDefinedPatternLo, unsigned long long UserDefinedPatternMid, unsigned long long
UserDefinedPatternHi)
{
return mlBert_SetTxUserPatternPAM (instance, channel, UserDefinedPatternLo,
UserDefinedPatternMid, UserDefinedPatternHi);
}
```

mlBert_ML403bHVAmpltiude

This API call is used to change the output level of the signal in HV channel ML4003B-HV.

Returns 1 on success, 0 on failure

Used for ML4003B-HV

Example of use:

```
/*
* mlBert_ML403bHVAmpltiude EntryPoint inside the DLL allows to change the output level of the signal
in HV channel ML403b
 */

 [DllImport(@"DLL/MLBert_API.dll", EntryPoint = " mlBert_ML403bHVAmpltiude")]
private static extern int mlBert_ML403bHVAmpltiude (mlbertapi* instance, double AmplitudeValue,
double RiseTimeAmplitude, double FallTimeAmplitude);

/*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Param double AmplitudeValue: output level of the signal in steps
* Param double RiseTimeAmplitude: output level of the RiseTime
* Param double FallTimeAmplitude: output level of the FallTime
* Returns 1 on success, 0 on failure
*/

Public int ML403bHVAmpltiude (mlbertapi* instance, double AmplitudeValue, double
RiseTimeAmplitude, double FallTimeAmplitude)
{
return mlBert_ML403bHVAmpltiude (instance, AmplitudeValue, RiseTimeAmplitude,
FallTimeAmplitude);
}
```

mlBert_InnerEyeLevel

This API call is used to change the output level of the Inner Eye in PAM boards.

Returns 1 on success, 0 on failure

Used for ML4004-PAM, ML4079D, ML4039E, ML4039D, ML4039B and ML4039PAM-ATE

Example of use:

```
/*
* mlBert_InnerEyeLevel EntryPoint inside the DLL allows to change the output level of the Inner Eye in
PAM boards
*/

 [DllImport(@"DLL/MLBert_API.dll", EntryPoint = " mlBert_InnerEyeLevel")]
private static extern int mlBert_InnerEyeLevel (mlbertapi* instance, int channel, int value);

/*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Param int channel: channel selected
*Para int value: amplitude for the inner eye, this value is from 0 to 15
* Returns 1 on success, 0 on failure
*/

Public int InnerEyeLevel (mlbertapi* instance, int channel, int value)
{
return mlBert_InnerEyeLevel (instance, channel, value);
}
```

mlBert_ReadErrorCorrectionPolaris

This API call is used to read corrected errors.

Returns 1 on success and 0 on failure

Used for ML4039D and ML4079D

Example of use:

```
/*
* mlBert_ReadErrorCorrectionPolaris EntryPoint inside the DLL allows to read corrected errors
*/

 [DllImport(@"DLL/MLBert_API.dll", EntryPoint = " mlBert_ReadErrorCorrectionPolaris")]
private static extern int mlBert_ReadErrorCorrectionPolaris (mlbertapi* instance, int channelIdx, UInt64
errorCorrected[], UInt64 BlockCount[], UInt64 SaturatedBlockCount[]);

/*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Param channelIdx:
* Param UInt64 errorCorrected[]: BER corrected errors
* Param UInt64 BlockCount[]: BER Block count
* Param UInt64 SaturatedBlockCount[]: BER saturated block count
* Returns 1 on success and 0 on failure
*/

Public int ReadErrorCorrectionPolaris (mlbertapi* instance, int channelIdx, UInt64 errorCorrected[],
UInt64 BlockCount[], UInt64 SaturatedBlockCount[])
{
return mlBert_ReadErrorCorrectionPolaris (instance, channelIdx, errorCorrected[], BlockCount[],
SaturatedBlockCount[]);
}
```

mlBert_GetPam4Histogram

This API call is used to read the PAM4 histogram for a specific channel

Returns 1 on success, 0 on failure

Used for all PAM BERTs

Example of use:

```
/*
* mlBert_GetPam4Histogram EntryPoint inside the DLL allows to read the PAM4 histogram for a specific
channel
*/

 [DllImport(@"DLL/MLBert_API.dll", EntryPoint = " mlBert_GetPam4Histogram")]
private static extern int mlBert_GetPam4Histogram (mlbertapi* instance, int channel, double *xValues,
double *yValues);

/*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Param int channel: Channel number
* Param double *xValues: return the XValues in an array of 160 values
* Param double *yValues: return the YValues of a specific xValue in an array of 160 values
* Returns 1 on success, 0 on failure
*/

Public int GetPam4Histogram (mlbertapi* instance, int channel, double *xValues, double *yValues)
{
return mlBert_GetPam4Histogram (instance, channel, ref xValues, ref yValues);
}
```

mlBert_GetPam4SignalToNoiseRatio

This API call is used to read the PAM4 signal to noise ratio for a specific channel

Returns 1 on success, 0 on failure

Used for all PAM BERTs

Example of use:

```
/*
* mlBert_GetPam4SignalToNoiseRatio EntryPoint inside the DLL allows to read the PAM4 signal to noise
ratio for a specific channel
*/

 [DllImport(@"DLL/MLBert_API.dll", EntryPoint = " mlBert_GetPam4SignalToNoiseRatio")]
private static extern int mlBert_GetPam4SignalToNoiseRatio (mlbertapi* instance, int channel, double
*Values);

/*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Param int channel: channel number
* Param double Values: return the ratio
* Returns 1 on success, 0 on failure
*/

Public int GetPam4SignalToNoiseRatio (mlbertapi* instance, int channel, double *Values)
{
return mlBert_GetPam4SignalToNoiseRatio (instance, channel, ref Values);
}
```

mlBert_Tap0

This API call is used to enable TX7 taps in ML4039E, ML4039D and ML4079D

Returns 1 on success and 0 on failure

Used for ML4039E, ML4039D and ML4079D

Example of use:

```
/*
* mlBert_Tap0 EntryPoint inside the DLL allows to enable TX7 taps in ML4039E, ML4039D and ML4079D
*/
 [DllImport(@"DLL/MLBert_API.dll", EntryPoint = " mlBert_Tap0")]
private static extern int mlBert_Tap0 (mlbertapi* instance, int channel, int value);

/*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Param int channel:
*Para int value:
* Returns 1 on success and 0 on failure
*/

Public int Tap0 (mlbertapi* instance, int channel, int value)
{
return mlBert_Tap0 (instance, channel, value);
}
```

mlBert_Tap1

This API call is used to enable TX7 taps in ML4039E, ML4039D and ML4079D

Returns 1 on success and 0 on failure

Used for ML4039E, ML4039D and ML4079D

Example of use:

```
/*
* mlBert_Tap1 EntryPoint inside the DLL allows to enable TX7 taps in ML4039E, ML4039D and ML4079D
*/

 [DllImport(@"DLL/MLBert_API.dll", EntryPoint = " mlBert_Tap1")]
private static extern int mlBert_Tap1 (mlbertapi* instance, int channel, int value);

/*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Param int channel:
*Para int value:
* Returns 1 on success and 0 on failure
*/

Public int Tap1 (mlbertapi* instance, int channel, int value)
{
return mlBert_Tap1 (instance, channel, value);
}
```

mlBert_Tap2

This API call is used to enable TX7 taps in ML4039E, ML4039D and ML4079D

Returns 1 on success and 0 on failure

Used for ML4039E, ML4039D and ML4079D

Example of use:

```
/*
* mlBert_Tap2 EntryPoint inside the DLL allows to enable TX7 taps in ML4039E, ML4039D and ML4079D
*/

 [DllImport(@"DLL/MLBert_API.dll", EntryPoint = " mlBert_Tap2")]
private static extern int mlBert_Tap2 (mlbertapi* instance, int channel, int value);

/*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Param int channel:
*Para int value:
* Returns 1 on success and 0 on failure
*/

Public int Tap2 (mlbertapi* instance, int channel, int value)
{
return mlBert_Tap2 (instance, channel, value);
}
```

mlBert_Tap3

This API call is used to enable TX7 taps in ML4039E, ML4039D and ML4079D

Returns 1 on success and 0 on failure

Used for ML4039E, ML4039D and ML4079D

Example of use:

```
/*
* mlBert_Tap3 EntryPoint inside the DLL allows to enable TX7 taps in ML4039E, ML4039D and ML4079D
*/

 [DllImport(@"DLL/MLBert_API.dll", EntryPoint = " mlBert_Tap3")]
private static extern int mlBert_Tap3 (mlbertapi* instance, int channel, int value);

/*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Param int channel:
*Para int value:
* Returns 1 on success and 0 on failure
*/

Public int Tap3 (mlbertapi* instance, int channel, int value)
{
return mlBert_Tap3 (instance, channel, value);
}
```

mlBert_Tap4

This API call is used to enable TX7 taps in ML4039E, ML4039D and ML4079D

Returns 1 on success and 0 on failure

Used for ML4039E, ML4039D and ML4079D

Example of use:

```
/*
* mlBert_Tap4 EntryPoint inside the DLL allows to enable TX7 taps in ML4039E, ML4039D and ML4079D
*/

 [DllImport(@"DLL/MLBert_API.dll", EntryPoint = " mlBert_Tap4")]
private static extern int mlBert_Tap4 (mlbertapi* instance, int channel, int value);

/*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Param int channel:
*Para int value:
* Returns 1 on success and 0 on failure
*/

Public int Tap4 (mlbertapi* instance, int channel, int value)
{
return mlBert_Tap4 (instance, channel, value);
}
```

mlBert_Tap5

This API call is used to enable TX7 taps in ML4039E, ML4039D and ML4079D

Returns 1 on success and 0 on failure

Used for ML4039E, ML4039D and ML4079D

Example of use:

```
/*
* mlBert_Tap5 EntryPoint inside the DLL allows to enable TX7 taps in ML4039E, ML4039D and ML4079D
*/

 [DllImport(@"DLL/MLBert_API.dll", EntryPoint = " mlBert_Tap5")]
private static extern int mlBert_Tap5 (mlbertapi* instance, int channel, int value);

/*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Param int channel:
*Para int value:
* Returns 1 on success and 0 on failure
*/

Public int Tap5 (mlbertapi* instance, int channel, int value)
{
return mlBert_Tap5 (instance, channel, value);
}
```

mlBert_Tap6

This API call is used to enable TX7 taps in ML4039E, ML4039D and ML4079D

Returns 1 on success and 0 on failure

Used for ML4039E, ML4039D and ML4079D

Example of use:

```
/*
* mlBert_Tap6 EntryPoint inside the DLL allows to enable TX7 taps in ML4039E, ML4039D and ML4079D
*/

 [DllImport(@"DLL/MLBert_API.dll", EntryPoint = " mlBert_Tap6")]
private static extern int mlBert_Tap6 (mlbertapi* instance, int channel, int value);

/*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Param int channel:
*Para int value:
* Returns 1 on success and 0 on failure
*/

Public int Tap6 (mlbertapi* instance, int channel, int value)
{
return mlBert_Tap6 (instance, channel, value);
}
```

mlBert_EnableSJRJBUJ

This API call is used to enable and disable the SJ, RJ and BUJ

Returns 1 on success, 0 on failure

Used for ML4039B-JIT

Example of use:

```
/*
* mlBert_EnableSJRJBUJEntryPoint inside the DLL allows to enable and disable the SJ, RJ and BUJ
*/

 [DllImport(@"DLL/MLBert_API.dll", EntryPoint = " mlBert_EnableSJRJBUJ")]
private static extern int mlBert_EnableSJRJBUJ (mlbertapi* instance, int channel, int status);

/*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Param int channel: channel number
*Para int status: 0 to disable, 1 to enable
* Returns 1 on success, 0 on failure
*/

Public int EnableSJRJBUJ (mlbertapi* instance, int channel, int status)
{
return mlBert_EnableSJRJBUJ (instance, channel, status);
}
```

mlBert_UnOptimizedPMRJ

This API call is used to disable the PM and RJ calibration

Returns 1 on success, 0 on failure

Used for ML4039B-JIT

Example of use:

```
/*
* mlBert_UnOptimizedPMRJ EntryPoint inside the DLL allows to disable the PM and RJ calibration
*/

 [DllImport(@"DLL/MLBert_API.dll", EntryPoint = " mlBert_UnOptimizedPMRJ")]
private static extern int mlBert_UnOptimizedPMRJ (mlbertapi* instance, int channel);

/*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Param int channel: channel index
* Returns 1 on success, 0 on failure
*/

Public int UnOptimizedPMRJ (mlbertapi* instance, int channel)
{
return mlBert_UnOptimizedPMRJ (instance, channel);
}
```

mlBert_LoadCalibrationValue

This API call is used to load calibration values

Returns 1 on success and 0 on failure

Used for all BERTs

Example of use:

```
/*
* mlBert_LoadCalibrationValue EntryPoint inside the DLL allows to load calibration values
*/

 [DllImport(@"DLL/MLBert_API.dll", EntryPoint = " mlBert_LoadCalibrationValue")]
private static extern int mlBert_LoadCalibrationValue (mlbertapi* instance);

/*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Returns 1 on success and 0 on failure
*/

Public int LoadCalibrationValue (mlbertapi* instance)
{
return mlBert_LoadCalibrationValue (instance);
}
```

mlBert_OptimizedPMRJ

This API call is used to enable the PM and RJ optimization for a specific channel

Returns the calibration number and 0 if not calibrated

Used for all JIT Boards

Example of use:

```
/*
* mlBert_OptimizedPMRJ EntryPoint inside the DLL allows to enable the PM and RJ optimization for a
specific channel
*/

 [DllImport(@"DLL/MLBert_API.dll", EntryPoint = " mlBert_OptimizedPMRJ")]
private static extern int mlBert_OptimizedPMRJ (mlbertapi* instance, int cahnnel, double* MInMax);

/*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Param int cahnnel: cahnnel index
* Param double* MInMax: array that contains the min and max for each SJ frequency
        {5Mhz, 10Mhz, 20Mhz, 40Mhz, 80Mhz}
*  Returns the calibration number and 0 if not calibrated
*/

Public int OptimizedPMRJ (mlbertapi* instance, int cahnnel, double* MInMax)
{
return mlBert_OptimizedPMRJ (instance, channe, , ref MInMax);
}
```

mlBert_QSFP_SetPower

This API call is used to Set QSFP Power between 3.15V, 3.33V and 3.45V

Returns true on success, false on failure

Used for ML4054-QSFP

Example of use:

```
/*
* mlBert_QSFP_SetPower EntryPoint inside the DLL allows Set QSFP Power between 3.15V , 3.33 V and
3.45V
*/

 [DllImport(@"DLL/MLBert_API.dll", EntryPoint = " mlBert_QSFP_SetPower")]
private static extern bool mlBert_QSFP_SetPower (mlbertapi* instance, int Module, int Rate);

/*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Param int Module: 0 or 1 to select the QSFP Module
* Param int Rate: Contains the Voltage index 0 = 3.15, 1 = 3.33, 2 =3.45
* Returns true on success, false on failure
*/

Public bool QSFP_SetPower (mlbertapi* instance, int Module, int Rate)
{
return mlBert_QSFP_SetPower (instance, Module, Rate);
}
```

mlBert_SFP_TX_DisableStatus

This API call is used to disable TX

Returns true if TX is disabled and false if it's not

Used for ML4003B

Example of use:

```
/*
* mlBert_SFP_TX_DisableStatus EntryPoint inside the DLL allows to disable TX
*/

 [DllImport(@"DLL/MLBert_API.dll", EntryPoint = " mlBert_SFP_TX_DisableStatus")]
private static extern bool mlBert_SFP_TX_DisableStatus (mlbertapi* instance, int Module);

/*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Param int Module:
* Returns true if TX is disabled and false if it's not
*/

Public bool SFP_TX_DisableStatus (mlbertapi* instance, int Module)
{
return mlBert_SFP_TX_DisableStatus (instance, Module);
}
```

mlBert_SFP_RS0_RateStatus

This API call is used to select SFP and RS0 Mode

Returns true on success, false on failure

Used for ML4003B

Example of use:

```
/*
* mlBert_SFP_RS0_RateStatus EntryPoint inside the DLL allows to select SFP and RS0 Mode
*/

 [DllImport(@"DLL/MLBert_API.dll", EntryPoint = " mlBert_SFP_RS0_RateStatus")]
private static extern bool mlBert_SFP_RS0_RateStatus (mlbertapi* instance, int module);

/*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Param int Module:
* Returns true on success, false on failure
*/

Public bool SFP_RS0_RateStatus (mlbertapi* instance, int module)
{
return mlBert_SFP_RS0_RateStatus (instance, module);
}
```

mlBert_SFP_RS1_RateStatus

This API call is used to select select SFP and RS1 Mode

Returns true on success, false on failure

Used for ML4003B

Example of use:

```
/*
* mlBert_SFP_RS1_RateStatus EntryPoint inside the DLL allows to select SFP and RS1 Mode
*/

 [DllImport(@"DLL/MLBert_API.dll", EntryPoint = mlBert_SFP_RS1_RateStatus")]
private static extern bool mlBert_SFP_RS1_RateStatus (mlbertapi* instance, int module);

/*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Param int Module:
* Returns true on success, false on failure
*/

Public bool SFP_RS1_RateStatus (mlbertapi* instance, int module)
{
return mlBert_SFP_RS1_RateStatus (instance, module);
}
```

mlBert_GetFFE

This API call is used to read RX 15 taps for ML4039E

Returns 1 on success and 0 on failure

Used for ML4039E

Example of use:

```
/*
* mlBert_GetFFE EntryPoint inside the DLL allows to read RX 15 taps for ML4039E
*/

 [DllImport(@"DLL/MLBert_API.dll", EntryPoint = " mlBert_GetFFE")]
private static extern int mlBert_GetFFE (mlbertapi* instance, int Channel, double *taps);

/*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Param int Channel:
* Param double *taps:
* Returns 1 on success and 0 on failure
*/

Public int GetFFE (mlbertapi* instance, int Channel, double *taps)
{
return mlBert_GetFFE (instance, Channel, ref taps);
}
```

mlBert_ReadBerMSBLSB

This API call is used to to read BER MSB and BER LSB for ML4039E, ML4079D and ML4039D

Returns 1 on success and 0 on failure

Used for ML4039E, ML4079D and ML4039D

Example of use:

```
/*
* mlBert_ReadBerMSBLSB EntryPoint inside the DLL allows to read BER MSB and BER LSB for ML4039E,
ML4079D and ML4039D
*/

 [DllImport(@"DLL/MLBert_API.dll", EntryPoint = " mlBert_ReadBerMSBLSB")]
private static extern int mlBert_ReadBerMSBLSB (mlbertapi* instance, int channel, double* BERMSB,
double* BERLSB);

/*
* Public Interface to connect with DLL
* Param mlbertapi* instance
* Param int Channel:
* Param double * BERMSB:
* Param double* BERLSB:
* Returns 1 on success and 0 on failure
*/

Public int APIReadBerMSBLSB (mlbertapi* instance, int channel, double* BERMSB, double* BERLSB)
{
return mlBert_ReadBerMSBLSB (instance, channel, ref BERMSB, ref BERLSB);
}
```