# ML4054-400
# API documentation

Rev 1.4 (API v1.6)

Jully 10th, 2019

# Table of Contents

# I. Connection:

## APIConnect

This API call is responsible to establish a connection between the client application and the hardware. It 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.

```
int_stdcall APIConnect(byte instance, char* ip);
```

Arguments:
instance: API instance.
ip: pointer to the char array containing the IPv4 of the hardware.
Return value: 1 if success and 0 if failed.

C# Example:

```
public int Connect(byte instance, string ipAddress)
{
return APIConnect(instance, ipAddress);
}
```

## APIDisconnect

This API call is used to disconnect the client application from the hardware.

```
int_stdcall APIDisconnect(byte instance);
```

Arguments:
instance: API instance.
Return value: 1 if success and 0 if failed.

C# Example:

```
public int Disconnect(byte instance)
{
return APIDisconnect(instance);
}
```

## APIreadIPMask

This API call is used to read the Ethernet mask of the board.

```
bool_stdcall APIreadIPMask(byte instance, char* IPMask);
```

Arguments:
instance: API instance.
IPMask: Output argument, pointer to the char array containing the Ethernet mask of the hardware.
Return value: true if success and false if failed.

C# Example:

```
public int ReadIPMask(byte instance, ref string IPMask)
{
return APIreadIPMask(instance, ref IPMask);
}
```

## APIreadGateway

This API call is used to read the Ethernet gateway of the board.

```
bool_stdcall APIreadGateway(byte instance, char* IPGateway);
```

Arguments:
instance: API instance.
IPGateway: Output argument, pointer to the char array containing the Ethernet gateway of the hardware.
Return value: true if success and false if failed.

C# Example:

```
public int ReadGateway(byte instance, ref string IPGateway)
{
return APIReadGateway(instance, ref IPGateway);
}
```

## APIReadBoardID

This API call is used to read the board ID from the Firmware.

```
int_stdcall APIReadBoardID(byte instance);
```

Arguments:
instance: API instance.
Return value: 32 bits integer representing the board's ID.

C# Example:

```
public int ReadBoardID(byte instance)
{
return APIReadBoardID(instance);
```

```
}
```

## APIReadSerialNumber

This API call is used to read the board serial number from the Firmware.

```
int_stdcall APIReadSerialNumber(byte instance, UINT64 *SN, int
*SerialNumberVersion);
```

Arguments:
instance: API instance.
SN: Output argument, pointer to a 64 bits unsinged integer representing the board's serial number.
SerialNumberVersion: Output argument, pointer to a 32 bits integer representing the board's serial number version (0 if old and 1 if new).
Return value: 1 if success and 0 if failed.

C# Example:

```
public int ReadSerialNumber(byte instance, ref UINT64 SerialNumber, ref int SerialNumberVersion)
{
return APIReadSerialNumber(instance, ref SerialNumber, ref SerialNumberVersion);
}
```

## APIConfigureApplication

This API call is used to configure the clock file location.

```
int_stdcall APIConfigureApplication(byte instance, char* saveConfig, char*
saveBathtub, char* saveEye, int saveBathtubEnable, int saveEyeEnable);
```

Arguments:
instance: API instance.
saveConfig: char array containing the path of the clock file (.clk) used for the line rate configuration.
saveBathtub: char array containing the path of the location to save the bathtub (not used for ML4054-400).
saveEye: char array containing the path of the location to save the eye diagram (not used for ML4054-400).
saveBathtubEnable: integer used to enable saving bathtub points (0 disable, 1 enable. Not used for ML4054-400).
saveEyeEnable: integer used to enable saving Eye diagram points (0 disable, 1 enable. Not used for ML4054-400).
Return value: 1 if success and 0 if failed.

C# Example:

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

## APIReadFirmwareRevision

This API call is used to read the firmware revision from the firmware.

```
double_stdcall APIReadFirmwareRevision(byte instance);
```

Arguments:
instance: API instance.
Return value: double representing the board's firmware revision.

C# Example:

```
public double ReadFirmwareRevision(byte instance)
{
return APIReadFirmwareRrevision(instance);
}
```

## APIreadIPConfiguration

This API call is used to read the IP of the board.

```
bool_stdcall APIreadIPConfiguration(byte instance, char* ip);
```

Arguments:
instance: API instance.
ip: Output argument, pointer to the char array containing the IPv4 of the hardware.
Return value: true if success and false if failed.

C# Example:

```
public int ReadIPConfiguration(byte instance, ref string ipAddress)
{
return APIReadIPConfiguration(instance, ref ipAddress);
}
```

## APIReadValueFromEEPROM

This API call is used to read a 16 bits value from the EEPROM of the hardware.
⚠ Please double check with ML support to get EEPROM address to read from.

```
bool _stdcall APIReadValueFromEEPROM(byte instance, UINT16 Address, UINT16 *
Data);
```

Arguments:
instance: API instance.
Address: EEPROM address.
Data: 16 bits data read from the EEPROM.
Return value: true if success and false if failed.

C# Example:

```
public bool ReadValueFromEEPROM(byte instance, ushort address, ref ushort Data)
{
return APIReadValueFromEEPROM(instance, address, ref Data);
}
```

## APIReadEEPROMAddressTest

This API call is used to read a double value from the EEPROM of the hardware (starting from 0x2000).
⚠ Please double check with ML support to get EEPROM address to read from.

```
int _stdcall APIReadEEPROMAddressTest(byte instance, int channel, UINT16
Address, double * Data);
```

Arguments:
instance: API instance.
channel: channel index (0x300 EEPROM register for every channel).
Address: EEPROM address.
Data: double data read from the EEPROM.
Return value: 1 if success and 0 if failed.

C# Example:

```
public int ReadEEPROMAddressTest(byte instance, int channel, uint address, ref double Data)
{
return APIReadEEPROMAddressTest(instance, channel, address, ref Data);
}
```

## APIReadCalibrationValues

This API call is used to read the calibration values from the EEPROM.
⚠ Please double check with ML support to get calibration values distribution on the EEPROM.

```
int_stdcall APIReadCalibrationValues(byte instance,double* values);
```

Arguments:
instance: API instance.

Values: Output parameter, pointer to an array of double containing the calibration values read from the EEPROM. Each channel has 14 values (7 for low rate and 7 for high rate).

Return value: 1 if success and 0 if failed.

C# Example:

```
public int ReadCalibrationValues(byte instance, double[] values)
{
return APIReadCalibrationValues(instance, ref Data);
}
```

## APISaveOptimalSettings

This API call is used to save the optimal settings (PreEmphasis, MainTap, PostEmphasis, Inner Eye and Outer Eye) for each channel.

⚠ Using this function will delete ML optimal settings saved on the board.

```
bool _stdcall APISaveOptimalSettings(byte instance, int Channel, int mode,
double *Data);
```

Arguments:

instance: API instance.

channel: index of the channel.

mode: 0 for low rate and 1 for high rate.

Data: pointer to an array containing the five optimal settings values.

Return value: true if success and false if failed.

C# Example:

```
public bool SaveOptimalSettings(byte instance, int channel, double[] values)
{
return APISaveOptimalSettings(instance, channel, Data);
}
```

## APIReadOptimalSettings

This API call is used to load the optimal settings (PreEmphasis, MainTap, PostEmphasis, Inner Eye and Outer Eye) for each channel.

```
bool _stdcall APIReadOptimalSettings(byte instance, int Channel, int mode,
double *Data);
```

Arguments:

instance: API instance.

channel: index of the channel.

mode: 0 for low rate and 1 for high rate.
Data: Output parameter, pointer to an array containing the five optimal settings values read from the EEPROM.
Return value: true if success and false if failed.

C# Example:

```
public bool ReadOptimalSettings(byte instance, int channel, double[] values)
{
return APIReadOptimalSettings(instance, channel, Data);
}
```

## APILineRateConfiguration_4044B

This API call is used to set the line rate and its configuration on the board.

```
int __stdcall APILineRateConfiguration_4044B(byte instance, double lineRate,
int clockSource, EyeMode_ML4004 eyeMode, int PreCoding, int GrayMapping, int
FECEnable, int FECType, int Taps);
```

Arguments:
instance: API instance.
linerate: line rate to be applied to the board.
Clocksource: 0 for external and 1 for internal silab.
eyeMode: 0 for PAM4 and 1 for NRZ.
PreCoding: 0 to disable pre coding and 1 to enable it.
GrayMapping: 0 to disable gray mapping and 1 to enable it.
FECEnable: 0 to disable FEC and 1 to enable it.
FECType: 0 for RS544, 1 for RS528 and 2 for Fire code.
Side: 0 for 3 Taps (Pre/Main/Post), 1 to enable 7 Taps (FW rev >= 1.8)
Return value: true if success and false if failed.

C# Example:

```
Public int LineRateConfiguration_4044B(double linerate, int clocksource, int Eyemode, int precoding, int
graymapping, int FECEnable, int FECType, int Taps)
{
return APILineRateConfiguration_4044B(linerate, clocksource, Eyemode, precoding, graymapping,
FECEnable, FECType, Taps)
}
```

# APIClockOut4044B

This API call is used to set the clock out of the board. The clock out can be either the reference clock or the monitor clock divided by a rate.

```
int __stdcall APIClockOut4044B(byte instance, int TriggerOut,  int Rate);
```

Arguments:

instance: API instance.

TriggerOut: 0 for monitor clock, 1 for reference clock and 3 for TXPLL Clock ( to get TXPLL clock, a hardware rework should be done to the board)

Rate: monitor clock divider (0 for 32, 1 for 64, 2 for 128…. 7 for 4096) or TXPLL clock divider ( 2/4/8/16/32) This feature is available in FW rev 1.8 and above and a HW rework should be done to the board.

Return value: 1 if success and 0 if failed.

C# Example:

```
public int ClockOut4044B(byte instance, int TriggerOut, int rate)
{
return APIClockOut4044B(instance, TriggerOut, rate);
}
```

# II. Monitoring:

## APITempRead

This API call is used to read the temperature from different sensors on the board.

```
double_stdcall APITempRead(byte instance, int index);
```

Arguments:
instance: API instance.
index: index of the temperature sensor.
Return value: double precision variable containing the temperature read from the specific sensor.
C# Example:

```
public double TempRead(byte instance, int index)
{
return APITempRead(instance, index);
}
```

## APIReadAdapterType_4054

This API call is used to read the type of the adapter plugged in the ML4054-400.

```
bool _stdcall APIReadAdapterType_4054(byte instance, UINT16 *Data);
```

Arguments:
instance: API instance.
Data: Output argument, pointer to a 16 bits containing the adapter type (0 for unknown, 1 for QDD and 2 for OSFP)
Return value: true if success and false if failed.

C# Example:

```
public double ReadAdapterType_4054(byte instance, ref ushort type)
{
return APIReadAdapterType_4054 (instance, ref type);
}
```

## APIPRBSVerification

This API call is used to check the RX lock for each channel.

```
int_stdcall APIPRBSVerfication(byte instance, int channel);
```

Arguments:

instance: API instance.
Channel: channel index.
Return value: 0 if RX is not locked and 1 if RX is locked.

C# Example:

```
public int PRBSVerification(byte instance, int channel)
{
return APIPRBSVerification(instance, channel);
}
```

## APIReadTXLock

This API call is used to read the TX lock for each channel.

```
bool _stdcall APIReadTXLock(byte instance,int channel, bool *status);
```

Arguments:
instance: API instance.
Channel: channel index.
status: Output argument, pointer to a bool containing status of the TX Lock.
Return value: true if success and false if failed.

C# Example:

```
public double ReadTXLock(byte instance, int channel, ref bool status)
{
return APIReadTXLock(instance, channel, ref type);
}
```

## APIMonitor

This API call is used to read latest applied values from EEPROM for each channel.

```
int _stdcall APIMonitor(byte instance,int channel, byte StatusID, double
*Data);
```

Arguments:
instance: API instance.
Channel: channel index.
StatusID: Enumeration value of the configuration to be read.
Data: Output argument, pointer to a double containing the value read from the EEPROM.
Return value: 1 if success and 0 if failed.

C# Example:

```
public int Monitor(byte instance, int channel, byte StatusID, ref double data)
{
return APIMonitor(instance, channel, StatusID, ref data);
}
```

ML Monitor Enumeration:

- The values marked in red are used for the ML4054-400
- APImonitor reads the latest applied value from the EEPROM (values must be applied using API functions)

```
enum E_Monitor {

        M_ClockIn = 0,
        M_LineRate = 1, // Line Rate value
        M_ClockOut = 2,

        M_TX_enable = 3,
        M_RX_enable = 4,

        M_TXpattern = 5, // TX Pattern value
        M_CustomPattern = 6,
        M_TXPattern_length = 7,
        M_TXinvert = 8,
        M_OutputLevel = 9, // Amplitude value
        M_RXpattern = 10, // RX Pattern value
        M_RXinvert = 11, // RX Invert value
        M_RXPatternLength = 12,
        M_PatternLock = 13,

        M_PreEmphasis = 14, // PreEmaphasis value
        M_PostEmphasis = 15, // PostEmphasis value

        M_ErrorIns = 16,
        M_RXDFEValue = 17, // CTLE value

        M_BERTimer = 18,
        M_BERPhase = 19,
        M_BERVerticalOffset = 20,

        M_PhaseSkew = 21,
        M_PMFreq = 22,
        M_PMAmp = 23,
        M_PMRJ = 24,
        M_AMRJ = 25,
        M_HWTXInvert = 26,
        M_HWRXInvert = 27,
```

```
        M_CustomPatternMSB = 28,
        M_InternalLoopBackStatus = 29,
        M_ReferenceClockoutValue = 30,
        M_InnerAmplitude = 31, // Inner Eye Amplitude value

        M_PRE_Coding = 32, // Pre Coding (Enabled = 1, Disabled = 0)
        M_GrayMapping = 33, // Gray Mapping (Enabled = 1, Disabled = 0)
        M_FEC = 34, // FEC (Enabled = 1, Disabled = 0)
        M_TXMode = 35, // Eye Mode (PAM4 = 0, NRZ = 1)
        M_Amplitude = 36,
        M_FECOperationMode = 37,

        M_FM_Amplitude = 38,
        M_FM_Frequency = 39,
        M_CTLELowRateCalibration = 40,
        M_CTLEHightRateCalibration = 41,
        M_BUJAmplitude = 42,
        M_BUJFrequency = 43,
        M_HostSide = 44,
        M_HostLine = 45,
        M_HostLineDebug = 46,
        M_VGAtracking = 47,
        M_ClockType = 48,
        M_MainTap = 49, // Main Tap value
        M_OutterEye = 50, // Outer Eye Amplitude value
        M_VPEAK = 51,
        M_IEEE = 52,
        M_Alltaps = 53,
        M_Tap0 = 54, // Tap 1 value
        M_Tap1 = 55, // Tap 2 value
        M_Tap2 = 56, // Tap 3 value
        M_Tap3 = 57, // Tap 4 value
        M_Tap4 = 58, // Tap 5 value
        M_Tap5 = 59, // Tap 6 value
        M_Tap6 = 60, // Tap 7 value
        M_is70Calibrated = 61,
        M_is80Calibrated = 62,
        M_is90Calibrated = 63,
        M_is100Calibrated = 64,
        M_is110Calibrated = 65,
        M_is120Calibrated = 66,
        M_is60Calibrated = 67,
        M_DSPMode = 68, // DSP mode value
        M_isLowVoltageCalibrated = 69,
        M_isHighVoltageCalibrated = 70
};
```

# III. TX Configurations:

## APISetPRBSPattern

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.

```
int_stdcall APISetPRBSPattern(byte instance, int channel, int txPattern,
int rxPattern, int txInvert, int rxInvert);
```

Arguments:
instance: API instance.
Channel: channel index.
txPattern:

- 0 for PN7
- 1 for PN9
- 2 for PN11
- 3 for PN15
- 4 for PN23
- 5 for PN31
- 9 for PN13
- 10 for PN9_4.

rxPattern:

- 0 for PN7
- 1 for PN9
- 2 for PN11
- 3 for PN15
- 4 for PN23
- 5 for PN31
- 9 for PN13
- 10 for PN9_4.

txInvert: 1 if TX is inverted and 0 if not.
rxInvert: 1 if RX is inverted and 0 if not.
Return value: true if success and false if failed.

C# Example:

```
public int SetPRBSPattern(byteinstance, int channel, int txPattern, int rxPattern, int txInvert, int
rxInvert)
{
return APISetPRBSPattern(instance, channel, txPattern, rxPattern, txInvert, rxInvert);
}
```

## APIOutputLevel

This API call is used to set the output amplitude for each channel.

```
int_stdcall APIOutputLevel(byte instance, int channel, int value);
```

Arguments:
instance: API instance.
Channel: channel index.
Value: an integer containing a percentage controlling the output level (60%, 70%, 80%, 90%, 100%, 110%, 120%).
Return value: 1 if success and 0 if failed.

C# Example:

```
public int OutputLevel(byte instance, int channel, int value)
{
return APIOutputLevel(instance, channel, value);
}
```

## APIPreEmphasis

This API call is used to set the Pre Emphasis tap for each channel.

```
int_stdcall APIPreEmphasis(byte instance, int channel, int value);
```

Arguments:
instance: API instance.
Channel: channel index.
Value: an integer containing the value to be set (-1000 to 1000)
Return value: 1 if success and 0 if failed.

C# Example:

```
public int PreEmphasis(byte instance, int channel, int value)
{
return APIPreEmphasis(instance, channel, value);
}
```

## APIPostEmphasis

This API call is used to set the Post Emphasis tap for each channel.

```
int_stdcall APIPostEmphasis(byte instance, int channel, int value);
```

Arguments:
instance: API instance.

Channel: channel index.
Value: an integer containing the value to be set (-1000 to 1000)
Return value: 1 if success and 0 if failed.

C# Example:

```
public int PostEmphasis(byte instance, int channel, int value)
{
return APIPostEmphasis(instance, channel, value);
}
```

## APIMainTap

This API call is used to set the Main tap for each channel.

```
int_stdcall APIMainTap(byte instance, int channel, int value);
```

Arguments:
instance: API instance.
Channel: channel index.
Value: an integer containing the value to be set (-1000 to 1000)
Return value: 1 if success and 0 if failed.

C# Example:

```
public int MainTap(byte instance, int channel, int value)
{
return APIMainTap(instance, channel, value);
}
```

## APIInnerEyeLevel

This API call is used to set the PAM4 inner eye amplitude for each channel.

```
int_stdcall APIInnerEyeLevel(byte instance, int channel, int value);
```

Arguments:
instance: API instance.
Channel: channel index.
Value: an integer containing the value to be set. The inner eye amplitude decreases while increasing the value from 500 to 1500.
Return value: 1 if success and 0 if failed.

C# Example:

```
public int InnerEyeLevel(byte instance, int channel, int value)
{
```

```
return APIInnerEyeLevel(instance, channel, value);
}
```

## APIOuterEyeLevel

This API call is used to set the two PAM4 outer eyes amplitude for each channel.

```
int_stdcall APIOuterEyeLevel(byte instance, int channel, int value);
```

Arguments:
instance: API instance.
Channel: channel index.
Value: an integer containing the value to be set. The outer eyes amplitude decreases while increasing the value from 1500 to 2500.
Return value: 1 if success and 0 if failed.

C# Example:

```
public int OuterEyeLevel(byte instance, int channel, int value)
{
return APIOuterEyeLevel(instance, channel, value);
}
```

## APIErrorInsertionPolarise

This API call is used to enable and insert errors into each channel.

```
int _stdcall APIErrorInsertionPolarise(byte instance, int channel, int
enable, int gap, int mode, int duration);
```

Arguments:
instance: API instance.
Channel: channel index.
Enable: 1 to enable error insertion and 0 to disable it.
Gap: Number of 64-bit words without errors to insert between words with errors.
Mode:

- 0 for one bit error in the MSB.
- 1 for one bit error in the LSB.
- 2 for 1 error in the PAM4 symbol.
- 3 for errors on all the bits of the MSB.
- 4 for errors on all the bits of the LSB.
- 5 for errors on all bits of the LSB and MSB.
- 6 for error on one bit per word and the position shifts right each time.
- 7 for error on one PAM4 symbol per word and the position shifts right each time.

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. The spacing of the words with errors is determined by the gap configuration.
Return value: 1 if success and 0 if failed.

C# Example:

```
public int ErrorInsertionPolarise(byte instance, int channel, int enable, int gap, int mode , int duration)
{
return APIErrorInsertionPolarise(instance, channel, enable, gap, mode, duration);
}
```

# APITap0

This API call is used to set the first tap for each channel. 7 Taps should be enabled using APILineRateConfiguration_4044B.
This feature is available in FW rev 1.8 and above.

```
int_stdcall APITap0(byte instance, int channel, int value);
```

Arguments:
instance: API instance.
Channel: channel index.
Value: a integer containing the value to be set (-1000 to 1000)
Return value: 1 if success and 0 if failed.

C# Example:

```
public int Tap0(byte instance, int channel, int value)
{
return APITap0(instance, channel, value);
}
```

# APITap1

This API call is used to set the second tap for each channel. 7 Taps should be enabled using APILineRateConfiguration_4044B.
This feature is available in FW rev 1.8 and above.

```
int_stdcall APITap1(byte instance, int channel, int value);
```

Arguments:
instance: API instance.
Channel: channel index.
Value: a integer containing the value to be set (-1000 to 1000)

Return value: 1 if success and 0 if failed.

C# Example:

```
public int Tap1(byte instance, int channel, int value)
{
return APITap1(instance, channel, value);
}
```

## APITap2

This API call is used to set the third tap for each channel. 7 Taps should be enabled using APILineRateConfiguration_4044B.
This feature is available in FW rev 1.8 and above.

```
int_stdcall APITap2(byte instance, int channel, int value);
```

Arguments:
instance: API instance.
Channel: channel index.
Value: a integer containing the value to be set (-1000 to 1000)
Return value: 1 if success and 0 if failed.

C# Example:

```
public int Tap2(byte instance, int channel, int value)
{
return APITap2(instance, channel, value);
}
```

## APITap3

This API call is used to set the fourth tap for each channel. 7 Taps should be enabled using APILineRateConfiguration_4044B.
This feature is available in FW rev 1.8 and above.

```
int_stdcall APITap3(byte instance, int channel, int value);
```

Arguments:
instance: API instance.
Channel: channel index.
Value: a integer containing the value to be set (-1000 to 1000)
Return value: 1 if success and 0 if failed.

C# Example:

```
public int Tap3(byte instance, int channel, int value)
{
return APITap3(instance, channel, value);
}
```

## APITap4

This API call is used to set the fifth tap for each channel. 7 Taps should be enabled using APILineRateConfiguration_4044B.
This feature is available in FW rev 1.8 and above.

```
int_stdcall APITap4(byte instance, int channel, int value);
```

Arguments:
instance: API instance.
Channel: channel index.
Value: a integer containing the value to be set (-1000 to 1000)
Return value: 1 if success and 0 if failed.

C# Example:

```
public int Tap4(byte instance, int channel, int value)
{
return APITap4(instance, channel, value);
}
```

## APITap5

This API call is used to set the sixth tap for each channel. 7 Taps should be enabled using APILineRateConfiguration_4044B.
This feature is available in FW rev 1.8 and above.

```
int_stdcall APITap5(byte instance, int channel, int value);
```

Arguments:
instance: API instance.
Channel: channel index.
Value: a integer containing the value to be set (-1000 to 1000)
Return value: 1 if success and 0 if failed.

C# Example:

```
public int Tap5(byte instance, int channel, int value)
{
return APITap5(instance, channel, value);
```

}

## APITap6

This API call is used to set the seventh tap for each channel. 7 Taps should be enabled using APILineRateConfiguration_4044B.
This feature is available in FW rev 1.8 and above.

```
int_stdcall APITap6(byte instance, int channel, int value);
```

Arguments:
instance: API instance.
Channel: channel index.
Value: a integer containing the value to be set (-1000 to 1000)
Return value: 1 if success and 0 if failed.

C# Example:

```
public int Tap6(byte instance, int channel, int value)
{
return APITap6(instance, channel, value);
}
```

# IV. RX Configurations:

## APIDSPMode

This API call is used to set the DSP mode.

```
int __stdcall APIDSPMode(byte instance, int Channel, int Status);
```

Arguments:
instance: API instance.
Channel: channel index.
Status: 0 for PAM4 slicer used for short non-strenuous links and 2 for PAM4 slicer with reflection canceller.
Return value: 1 if success and 0 if failed.

C# Example:

```
public int DSPMode(byte instance, int channel, int value)
{
return APIDSPMode(instance, channel, value);
}
```

## APICTLESetValue

This API call is used to set the value of the CTLE on the chip.

```
int __stdcall APICTLESetValue(byte instance, int Channel, int Value);
```

Arguments:
instance: API instance.
Channel: channel index.
Value: integer containing the CTLE value to be set (0-63).
Return value: 1 if success and 0 if failed.

C# Example:

```
public int CTLESetValue(byte instance, int channel, int value)
{
return APICTLESetvalue(instance, channel, value);
}
```

## APIRealTimeBEREnable

This API call is used to start/stop real time BER.

```
int __stdcall APIRealTimeBEREnable(byte instance, int enable);
```

Arguments:
instance: API instance.
Channel: channel index.
Enable: 1 to enable real time BER and 2 to disable it.
Return value: 1 if success and 0 if failed.

C# Example:

```
public int RealTimeBEREnable(byte instance, int enable)
{
return APIRealTimeBEREnable(instance, enable);
}
```

## APIReadRealTimeBER

This API call is used to read the error counts and time values.

```
int __stdcall APIReadRealTimeBER (byte instance, UInt64 *errorCount,
UInt32 * TimeSpendIms);
```

Arguments:
instance: API instance.
Channel: channel index.
errorCount: output argument, pointer to an array containing the error counts for each index.
TimeSpendInms: output argument, pointer to an array containing the duration from the start of the BER for each channel.
Return value: 1 if success and 0 if failed.

C# Example:

```
public int ReadRealTimeBER(byte instance, ref UINT64 errorCount, ref UINT32 TimeSpentInms)
{
return APIReadRealTimeBER(instance, ref errorCount, ref TimeSpentInms);
}
```

## APIGetPam4SignalToNoiseRatio

This API call is used to read the signal to noise ratio for each channel.

```
int __stdcall APIGetPam4SignalToNoiseRatio(byte instance,int channel,
double *Values);
```

Arguments:
instance: API instance.

Channel: channel index.
Values: Output arguments, pointer to the SNR value read from the FW.
Return value: 1 if success and 0 if failed.

C# Example:

```
public int GetPam4SignalToNoiseRatio(byte instance, ref double value)
{
return APIGetPam4SignalToNoiseRatio(instance, ref value);
}
```

## APIGetPam4Histogram

This API call is used to read histogram X and Y values for PAM4 and NRZ.

```
int __stdcall APIGetPam4Histogram(byte instance,int channel, double
*xValues, double *yValues);
```

Arguments:
instance: API instance.
Channel: channel index.
xValues: Output arguments, pointer to an array containing the Histogram X values.
yValues: Output arguments, pointer to an array containing the Histogram Y values.
Return value: 1 if success and 0 if failed.

C# Example:

```
public int GetPam4SHistogram(byte instance, int channel, ref double xValue, ref double yValue)
{
return APIGetPam4Histogram(instance, channel, ref xValue, ref yValue);
}
```

## APIReSyncRX

This API call is used to reset the RX for every channel.
It is available for firmware >= 2.1

```
int __stdcall APIReSyncRX(byte instance,int channel);
```

Arguments:
instance: API instance.
Channel: channel index.
Return value: 1 if success and 0 if failed.

C# Example:

```
public int ReSyncRX(byte instance, int channel)
{
return APIReSyncRX(instance, channel);
}
```

# V. QDD:

## APIQDD_4054_MODPRS

This API call checks if a module is present in the QDD adapter plugged in the board.

```
bool _stdcall APIQDD_4054_MODPRS(byte instance, UINT16 index, bool *status);
```

Arguments:
instance: API instance.
index: index of the QDD adapter.
status: Output argument, true if a module is present and false otherwise.
Return value: true if success and false if failed.

C# Example:

```
public bool QDD_4054_MODPRS(byte instance, ushort index, ref bool status)
{
return APIQDD_4054_MODPRS (instance, index, ref status);
}
```

## APIQDD_4054_INT

This API call checks if an interrupt is present in the QDD plugged in the adapter.

```
bool _stdcall APIQDD_4054_INT(byte instance, UINT16 index, bool *status);
```

Arguments:
instance: API instance.
index: index of the QDD adapter.
status: Output argument, true if a interrupt is present and false otherwise.
Return value: true if success and false if failed.

C# Example:

```
public bool QDD_4054_INT(byte instance, ushort index, ref bool status)
{
return APIQDD_4054_INT (instance, index, ref status);
}
```

## APIQDD_4054_MODINIT

This API call is used to read/change the status of the MODINIT pin of the QDD.

```
bool _stdcall APIQDD_4054_MODINIT(byte instance, int Read_Write, UINT16
index, bool *status);
```

Arguments:
instance: API instance.
Read_Write: integer indicating the functionality of this function (0 = Read and 1 = Write)
index: index of the QDD adapter.
status: Output/Input argument, represent the status of the MODINIT pin of the QDD.
Return value: true if success and false if failed.

C# Example:

```
public bool QDD_4054_MODINIT(byte instance, int Read_Write, ushort index, ref bool status)
{
return APIQDD_4054_MODINIT(instance, Read_Write, index, ref status);
}
```

## APIQDD_4054_MODSEL

This API call is used to read/change the status of the MODSEL pin of the QDD.

```
bool _stdcall APIQDD_4054_MODSEL(byte instance, int Read_Write, UINT16 index,
bool *status);
```

Arguments:
instance: API instance.
Read_Write: integer indicating the functionality of this function (0 = Read and 1 = Write)
index: index of the QDD adapter.
status: Output/Input argument, represent the status of the MODSEL pin of the QDD.
Return value: true if success and false if failed.

C# Example:

```
public bool QDD_4054_MODSEL(byte instance, int Read_Write, ushort index, ref bool status)
{
return APIQDD_4054_MODSEL(instance, Read_Write, index, ref status);
}
```

## APIQDD_4054_MODRST

This API call is used to read/change the status of the MODRST pin of the QDD.

```
bool _stdcall APIQDD_4054_MODRST(byte instance, int Read_Write, UINT16 index,
bool *status);
```

Arguments:
instance: API instance.
Read_Write: integer indicating the functionality of this function (0 = Read and 1 = Write)
index: index of the QDD adapter.

status: Output/Input arguments, represent the status of the MODRST pin of the QDD.
Return value: true if success and false if failed.

C# Example:

```
public bool QDD_4054_MODRST(byte instance, int Read_Write, ushort index, ref bool status)
{
return APIQDD_4054_MODRST(instance, Read_Write, index, ref status);
}
```

## APIQDD_4054_Voltage

This API call reads the status of the voltage from the QDD adapter.

```
bool _stdcall APIQDD_4054_Voltage(byte instance, UINT16 index, double
*voltage);
```

Arguments:
instance: API instance.
index: index of the QDD adapter.
voltage: Output argument, pointer to a double containing the voltage.
Return value: true if success and false if failed.

C# Example:

```
public bool QDD_4054_Voltage(byte instance, ushort index, ref double voltage)
{
return APIQDD_4054_Voltage(instance, index, ref voltage);
}
```

## APIQDD_4054_VCC_Current

This API call reads the status of the VCC current from the QDD adapter.

```
bool _stdcall APIQDD_4054_VCC_Current(byte instance, UINT16 index, double
*current);
```

Arguments:
instance: API instance.
index: index of the QDD adapter.
current: Output argument, pointer to a double containing the current.
Return value: true if success and false if failed.

C# Example:

```
public bool QDD_4054_VCC_Current(byte instance, ushort index, ref double current)
```

```
{
return APIQDD_4054_VCC_Current(instance, index, ref current);
}
```

## APIQDD_4054_VCC1_Current

This API call reads the status of the VCC1 current from the QDD adapter.

```
bool _stdcall APIQDD_4054_VCC1_Current(byte instance, UINT16 index, double
*current);
```

Arguments:
instance: API instance.
index: index of the QDD adapter.
current: Output argument, pointer to a double containing the current.
Return value: true if success and false if failed.

C# Example:

```
public bool QDD_4054_VCC1_Current(byte instance, ushort index, ref double current)
{
return APIQDD_4054_VCC1_Current(instance, index, ref current);
}
```

## APIQDD_4054_VCCTX_Current

This API call reads the status of the VCCTX current from the QDD adapter.

```
bool _stdcall APIQDD_4054_VCCTX_Current(byte instance, UINT16 index, double
*current);
```

Arguments:
instance: API instance.
index: index of the QDD adapter.
current: Output argument, pointer to a double containing the current.
Return value: true if success and false if failed.

C# Example:

```
public bool QDD_4054_VCCTX_Current(byte instance, ushort index, ref double current)
{
return APIQDD_4054_VCCTX_Current(instance, index, ref current);
}
```

## APIQDD_4054_VCCRX_Current

This API call reads the status of the VCC current from the QDD adapter.

```
bool _stdcall APIQDD_4054_VCCRX_Current(byte instance, UINT16 index, double
*current);
```

Arguments:
instance: API instance.
index: index of the QDD adapter.
current: Output argument, pointer to a double containing the current.
Return value: true if success and false if failed.

C# Example:

```
public bool QDD_4054_VCCRX_Current(byte instance, ushort index, ref double current)
{
return APIQDD_4054_VCCRX_Current(instance, index, ref current);
}
```

## APIQDD_ReadI2C

This API call reads the value of a QDD register using I2C communication.

```
bool _stdcall APIQDD_ReadI2C(byte instance, int address, double* Data);
```

Arguments:
instance: API instance.
address: address of the register from the QDD MSA.
Data: Output argument, pointer to a double containing value of the register read.
Return value: true if success and false if failed.

C# Example:

```
public bool QDD_ReadI2C(byte instance, int address, ref double Data)
{
return APIQDD_ReadI2C(instance, address, ref Data);
}
```

## APIQDD_WriteI2C

This API call sets the value of a QDD register using I2C communication.

```
bool _stdcall APIQDD_WriteI2C(byte instance, int address, double Data);
```

Arguments:

instance: API instance.

address: address of the register from the QDD MSA.

Data: Output argument, double containing value of the register to be set.

Return value: true if success and false if failed.

C# Example:

```
public bool QDD_WriteI2C(byte instance, int address, double Data)
{
return APIQDD_WriteI2C(instance, address, Data);
}
```

# VI. OSFP:

## APIOSFP_4054_MODPRS

This API call checks if a module is present in the OSFP adapter plugged in the board.

```
bool _stdcall APIOSFP_4054_MODPRS(byte instance, UINT16 index, bool *status);
```

Arguments:
instance: API instance.
index: index of the OSFP adapter.
status: Output argument, true if a module is present and false otherwise.
Return value: true if success and false if failed.

C# Example:

```
public bool OSFP _4054_MODPRS(byte instance, ushort index, ref bool status)
{
return API OSFP _4054_MODPRS (instance, index, ref status);
}
```

## API OSFP_4054_INT

This API call checks if an interrupt is present in the OSFP plugged in the adapter.

```
bool _stdcall APIOSFP_4054_INT(byte instance, UINT16 index, bool *status);
```

Arguments:
instance: API instance.
index: index of the OSFP adapter.
status: Output argument, true if a interrupt is present and false otherwise.
Return value: true if success and false if failed.

C# Example:

```
public bool OSFP_4054_INT(byte instance, ushort index, ref bool status)
{
return API OSFP_4054_INT (instance, index, ref status);
}
```

## APIOSFP_4054_MODLP

This API call is used to read/change the status of the MODLP pin of the OSFP.

```
bool _stdcall APIOSFP_4054_MODLP(byte instance, int Read_Write, UINT16 index,
bool *status);
```

Arguments:
instance: API instance.
Read_Write: integer indicating the functionality of this function (0 = Read and 1 = Write)
index: index of the OSFP adapter.
status: Output/Input argument, represent the status of the MODLP pin of the OSFP.
Return value: true if success and false if failed.

C# Example:

```
public bool OSFP_4054_MODLP(byte instance, int Read_Write, ushort index, ref bool status)
{
return APIOSFP_4054_MODLP(instance, Read_Write, index, ref status);
}
```

## APIOSFP_4054_MODRST

This API call is used to read/change the status of the MODRST pin of the OSFP.

```
bool _stdcall APIOSFP_4054_MODRST(byte instance, int Read_Write, UINT16
index, bool *status);
```

Arguments:
instance: API instance.
Read_Write: integer indicating the functionality of this function (0 = Read and 1 = Write)
index: index of the OSFP adapter.
status: Output/Input arguments, represent the status of the MODRST pin of the OSFP.
Return value: true if success and false if failed.

C# Example:

```
public bool OSFP_4054_MODRST(byte instance, int Read_Write, ushort index, ref bool status)
{
return APIOSFP_4054_MODRST(instance, Read_Write, index, ref status);
}
```

## APIOSFP_4054_Voltage

This API call reads the status of the voltage from the OSFP adapter.

```
bool _stdcall APIOSFP_4054_Voltage(byte instance, UINT16 index, double
*voltage);
```

Arguments:
instance: API instance.
index: index of the OSFP adapter.

voltage: Output argument, pointer to a double containing the voltage.
Return value: true if success and false if failed.

C# Example:

```
public bool OSFP_4054_Voltage(byte instance, ushort index, ref double voltage)
{
return APIOSFP_4054_Voltage(instance, index, ref voltage);
}
```

## APIOSFP_4054_VCC_Current

This API call reads the status of the VCC current from the OSFP adapter.

```
bool _stdcall APIOSFP_4054_VCC_Current(byte instance, UINT16 index, double
*current);
```

Arguments:
instance: API instance.
index: index of the OSFP adapter.
current: Output argument, pointer to a double containing the current.
Return value: true if success and false if failed.

C# Example:

```
public bool OSFP_4054_VCC_Current(byte instance, ushort index, ref double current)
{
return APIOSFP_4054_VCC_Current(instance, index, ref current);
}
```

## APIOSFP_4054_ReadI2C

This API call reads the value of an OSFP register using I2C communication.

```
bool _stdcall APIOSFP_4054_ReadI2C(byte instance, int address, double* Data);
```

Arguments:
instance: API instance.
address: address of the register from the OSFP MSA.
Data: Output argument, pointer to a double containing value of the register read.
Return value: true if success and false if failed.

C# Example:

```
public bool OSFP_4054_ReadI2C(byte instance, int address, ref double Data)
{
```

```
return APIOSFP_4054_ReadI2C(instance, address, ref Data);
}
```

## APIOSFP_4054_WriteI2C

This API call sets the value of an OSFP register using I2C communication.

```
bool _stdcall APIOSFP_4054_WriteI2C(byte instance, int address, double Data);
```

Arguments:

instance: API instance.

address: address of the register from the OSFP MSA.

Data: Output argument, double containing value of the register to be set.

Return value: true if success and false if failed.

C# Example:

```
public bool OSFP_4054_WriteI2C(byte instance, int address, double Data)
{
return APIOSFP_4054_WriteI2C(instance, address, Data);
}
```