ML-TDR-API 1.0 API Manual

Contents

Introduction

In this document we explain the different functions of the TDR API.

Most functionality is demonstrated in details in the sample code release in the same package with the API library.

For any questions please email support@multilaneinc.com

List of Supported Measurements

a) Impedance profile
b) Insertion loss, near end crosstalk, and far end crosstalk.

MLTDR-API Functions and Parameters

In this section we describe the various API functions and their parameters

Connect (required)

```
bool Connect(string IP, bool isCoherent)
```

Routine Description:

Connect to

Board Arguments:

1. IP: The IP of the board to be connected.
2. isCoherent: specifies whether the board is coherent (parallel data capture) or not.

Return Value:

Returns true if the connection was successfully made

ConfigureTDR (required)

```
bool ConfigureTDR(int pulseMode,  vector<int> channelsConfiguration, int
averagingType, int numberOfAverages, bool averagePosistiveNegativeTDR,
double lpfCutOffFreq, double voltageAmplitude, double &timeBase, int
&timeResolution, double &frequencyBase, int &frequencyResolution)
```

Routine Description:

This function configures the TDR for execution.

Arguments:

1. pulseMode:	TDR mode: 0, TDT mode: 1
2. channelsConfiguration:	Array indexes are equivalent to TDR channel indexes. In TDR mode: Set channel index to 1 if active and to 0 if inactive. In TDT mode: Set Tx channel indexes to 1 and Rx channel indexes to -1. (0 if inactive).
3. averagingType	Hardware averaging :0, Software averaging : 1.
4. Number of averages	Set the number of the desired averages.(default 1)
5. aveargePosistiveNegativeTDR	Enable : 1, disable:0.
6. lpfCutOffFreq	Set low pass filter cut off frequency in GHz. (default 0).
7. voltageAmplitude	Set voltage amplitude from 0 to 255. If 0, TDR is used as a scope.
8. timeBase	returns the value of the time base used by the device.
9. timeResolution	Number of samples in time domain
10. frequencyBase	returns the value of the frequency base used by the device.
11. frequencyResolution	Number of points in frequency domain.

GetMeasurementData(required)

```
bool GetMeasurementData()
```

Routine Description:

This function acquires data from all channels.

Return Value:

Returns true if GetMeasurementData was successfully executed

ReadTDRValues (optional)

```
int ReadTDRValues(vector<double> *y)
```

Routine Description:

Return the data captured on all channels.

Arguments:

y: Returned TDR values captured on all channels. The indexes of the main array represent the indexes of the instrument channels.

Return Value:

Returns the number of samples in the data capture.

ReadTDRChannel (optional)

```
int ReadTDRChannel(std::vector<double>& y, int channelNumber)
```

Routine Description:

Return the data captured on a desired channel.

Arguments:

y: Returned TDR values captured on the desired channel.
channelNumber: Specify which channel need to be measured.

Return Value:

Returns the number of samples in the data capture.

SetAsReference (optional)

```
bool SetAsReference()
```

Routine Description:

This function captures the reference circuit used in calculating the insertion loss.

Return Value:

Returns true if reference was captured successfully.

ReadS21Values (optional)

```
int ReadS21Values(vector<double>& frequencyValues, vector<double>& y0, vector<double>& y1, vector<double>& y2, vector<double>& y3, vector<double>& y4, vector<double> & y5, vector<double>& y6, vector<double>& y7)
```

Routine Description:

Return the insertion loss captured on Rx channel. If there's no reference circuit. Data will be empty. (Reference circuit is captured using setAsReference function).

Arguments:

1. frequencyValues: Returned frequency values.
2. y0: returned insertion loss magnitude captured on channel 0.
   y1: returned insertion loss magnitude captured on channel 1.
   y2: returned insertion loss magnitude captured on channel 2.
   y3: returned insertion loss magnitude captured on channel 3.
   y4: returned insertion loss magnitude captured on channel 4.
   y5: returned insertion loss magnitude captured on channel 5.
   y6: returned insertion loss magnitude captured on channel 6.
   y7: returned insertion loss magnitude captured on channel 7.

Return Value:

Returns the number of data captured.

ReadS21MagValues (optional)

```
int ReadS21MagValues(vector<double>& frequencyValues, vector<double> * yMag)
```

Routine Description:

Return the magnitude of the insertion loss captured on Rx channel. If there's no reference circuit. Data will be empty. (Reference circuit is captured using setAsReference function).

Arguments:

1. frequencyValues: Returned frequency values.
2. yMag: returned insertion loss magnitude captured on all channels. The indexes of the main array represent the indexes of the instrument channels.

Return Value:

Returns the number of data captured.

ReadSerialNumber (optional)

```
UInt64 ReadSerialNumber(UInt64 * SN)
```

Routine Description:

This function returns the serial number of the board.

Arguments:

SN: Returned serial number

Return Value:

Returns 1 if ReadSerialNumber was successfully executed.

GetTimeBase (optional)

```
bool GetTimeBase(double &timeBase)
```

Routine Description:

This function returns the value of the time base used by the device.

Arguments:

timeBase: Returned time base.

Return Value:

Returns true if GetTimeBase was successfully executed.

Special Purpose Functions (Don't use unless instructed by MultiLane)

```
bool AccessBoardRegister(UInt16 Read_Write, UInt16 Index, UInt16 Reg, UInt16 *Data)
```

```
bool ReadCalibration(int chIndex, double lineRate, double &A, double &B, double & T, double &Tx)
```

```
bool ChangeIP(char*IPToChange)
```

Usual Flow of Required Functions

In this section we describe the usual flow of the required functions:

1- Instantiate a TDRWrapper object where applicable (C#, C++, C), otherwise an ML_TDR_API object. Keep reusing this instance for the same device to avoid the need to reconfigure things every time.
2- Connect
3- ConfigureTDR

Loop as much as needed:

4- GetMeasurementData
   - Read Data : (ReadTDRValues/ Reads21Values)

Sample Code

Sample code is usually provided with the API as well and may show more features