# QSFP API Documentation

## Table of Contents

1. Introduction ..... 2
1.1. Acronyms and abbreviations ..... 2
2. APIs ..... 2
2.1. General Functions ..... 2
2.1.1. USB Connection ..... 2
- ConnectToHost ..... 2
- Disconnect ..... 2
- GetDeviceCount ..... 2

2.1.2. Monitoring ..... 3
- P3V3_Current_Monitor ..... 3
- GetVCC ..... 3
- Get_VCCTX ..... 3
- Get_VCCRX ..... 3

2.2. QSFP MSA functions ..... 4
2.2.1. I2C access ..... 4
- I2CRead ..... 4
- I2CWrite ..... 4

2.2.2. Alarms and controls signals ..... 4
- MODPRS ..... 4
- LPMODE ..... 5
- RESET ..... 5
- MODSEL ..... 5
- IntL ..... 5

# QSFP API Documentation

## 1. Introduction

This document describes the various Application Programming Interface (API) functions for the Multilane QSFP host boards (ML4021 and ML4041). Each function is described with its parameters and return values.

### 1.1. Acronyms and abbreviations

| | |
|---|---|
| **API** | Application Programming Interface |
| **DLL** | Dynamic Link Library (.dll file) |
| **USB** | Universal Serial Bus |
| **I2C** | Inter-Integrated Circuit |

## 2. APIs

### 2.1. General Functions

#### 2.1.1. USB Connection

- **ConnectToHost**

| | |
|---|---|
| **Description** | Opens a USB connection to QSFP Host |
| **Call** | **bool __stdcall ConnectToHost(UInt16 Instance)** |
| **Parameters** | UInt16 Instance: USB instance of plugged host |
| **Returns** | True or False |

- **Disconnect**

| | |
|---|---|
| **Description** | Disconnects from CFP2 Host and close open USB connection |
| **Call** | **bool __stdcall Disconnect(UInt16 Instance)** |
| **Parameters** | UInt16 Instance: USB instance of plugged host |
| **Returns** | True or False |

- **GetDeviceCount**

| | |
|---|---|
| **Description** | Gets the number of devices attached |
| **Call** | **UInt32 __stdcall GetDeviceCount(void)** |
| **Parameters** | None |
| **Returns** | Number of connected devices |

### 2.1.2. Monitoring

- **P3V3_Current_Monitor**

| | |
|---|---|
| **Description** | Measures current value on the 3.3V line |
| **Call** | **bool __stdcall P3V3_Current_Monitor(UInt16 Instance, double* data)** |
| **Parameters** | UInt16 Instance: USB instance<br>double* data: Current value in mA |
| **Returns** | True or False |

- **GetVCC**

| | |
|---|---|
| **Description** | Measures voltage on the VCC1 line |
| **Call** | **bool __stdcall GetVCC(int Instance, double* Data)** |
| **Parameters** | UInt16 Instance: USB instance<br>double* data: VCC1 value in V |
| **Returns** | True or False |

- **Get_VCCTX**

| | |
|---|---|
| **Description** | Measures voltage on the VCCTX line |
| **Call** | **bool __stdcall Get_VCCTX(int Instance, double* Data)** |
| **Parameters** | UInt16 Instance: USB instance<br>double* data: VCCTX value in V |
| **Returns** | True or False |

- **Get_VCCRX**

| | |
|---|---|
| **Description** | Measures voltage on the VCCRX line |
| **Call** | **bool __stdcall Get_VCCRX(int Instance, double* Data)** |
| **Parameters** | UInt16 Instance: USB instance<br>double* data: VCCRX value in V |
| **Returns** | True or False |

# QSFP API Documentation

## 2.2. QSFP MSA functions

### 2.2.1. I2C access

- **I2CRead**

| | |
|---|---|
| **Description** | Reads I2C |
| **Call** | **bool __stdcall I2CRead(int Instance, BYTE SlaveAddress, BYTE registerAddress, BYTE* ReadBuff, BYTE Length);** |
| **Parameters** | int instance: USB instance<br>BYTE SlaveAddress: Slave Address which is 0xA0<br>BYTE registerAddress: Register to read data from<br>BYTE* ReadBuff: Pointer to the data that is read<br>BYTE Length: Number of registers read sequentially (max=32) |
| **Returns** | True or False |

- **I2CWrite**

| | |
|---|---|
| **Description** | Writes I2C |
| **Call** | **bool __stdcall I2CWrite(int Instance, BYTE SlaveAddress, BYTE Register , BYTE Value);** |
| **Parameters** | int instance: USB instance<br>BYTE SlaveAddress: Slave Address which is 0xA0<br>BYTE Register: Register where the data will be written<br>BYTE Value: The value to write on the Register |
| **Returns** | True or False |

### 2.2.2. Alarms and controls signals

- **MODPRS**

| | |
|---|---|
| **Description** | Reads the opposite of MODPRS_L QSFP pin to check if the QSFP module is inserted in the Host. |
| **Call** | **bool __stdcall MODPRS(UInt16 Instance, bool* status)** |
| **Parameters** | UInt16 Instance: USB instance<br>bool* status: True if HW pin is 0 : module present<br>False if HW pin is 1 : module absent |
| **Returns** | True or False |

# QSFP API Documentation

## LPMODE

| | |
|---|---|
| **Description** | Asserts/Deasserts LPMODE |
| **Call** | **bool __stdcall LPMODE(UInt16 Instance, bool asserted)** |
| **Parameters** | UInt16 Instance: USB instance<br>bool asserted: True to assert LPMODE<br>False to deassert LPMODE |
| **Returns** | True or False |

## RESET

| | |
|---|---|
| **Description** | Asserts/Deasserts RESET (inverse of RESET_L pin) |
| **Call** | **bool __stdcall RESET (UInt16 Instance, bool asserted)** |
| **Parameters** | UInt16 Instance: USB instance<br>bool asserted: True to assert RESET (RESET_L pin: 0)<br>False to deassert RESET (RESET_L pin: 1) |
| **Returns** | True or False |

## MODSEL

| | |
|---|---|
| **Description** | Asserts/Deasserts MODSEL (inverse of MODSEL_L pin) |
| **Call** | **bool __stdcall MODSEL(UInt16 Instance, bool asserted)** |
| **Parameters** | UInt16 Instance: USB instance<br>bool asserted: True to assert MODSEL (MODSEL_L pin: 0)<br>False to deassert MODSEL (MODSEL_L pin: 1) |
| **Returns** | True or False |

## IntL

| | |
|---|---|
| **Description** | Reads the opposite INT_L QSFP pin |
| **Call** | **bool __stdcall IntL(int Instance, bool* status)** |
| **Parameters** | int Instance: USB instance<br>bool* status: True if HW pin is 1<br>False if HW pin is 0 |
| **Returns** | True or False |