



PL23B3/23C3/23D3 HID to UART/I2C/SPI Windows SDK Programming Guide

Document Revision: 0.7
Document Release: Feb 24, 2020

Prolific Technology Inc.

7F, No. 48, Sec. 3, Nan Kang Rd.
Nan Kang, Taipei 115, Taiwan, R.O.C.
Telephone: +886-2-2654-6363
Fax: +886-2-2654-6161
E-mail: sales@prolific.com.tw
Website: <http://www.prolific.com.tw>

Table of Contents

Revision History	3
1. Introduction.....	4
2. SDK Library File.....	5
3. SDK API Function	5
3.1 Enum Device API Function and Description	5
3.1.1 GetSDKVersion	5
3.1.2 EnumDeviceByVid.....	5
3.1.3 EnumDeviceByVidPid	5
3.1.4 GetVidPidByIndex	5
3.1.5 GetVidPidSerialNumberByIndex	6
3.1.6 OpenDeviceHandle	6
3.1.7 CloseDeviceHandle.....	6
3.2 UART API Function and Description	6
3.2.1 GetUartConfig	6
3.2.2 SetUartConfig.....	6
3.2.3 UartRead	7
3.2.4 UartWrite	7
3.2.5 SetXonXoffSymbol	7
3.2.6 UartReset	7
3.3 I2C API Function and Description	7
3.3.1 SetI2CDeviceAddress	7
3.3.2 SetI2CFrequency	8
3.3.3 I2CReset.....	8
3.3.4 I2CRead	8
3.3.5 I2CWrite.....	8
3.3.6 I2CWriteRead	8
3.4 SPI API Function and Description.....	9
3.4.1 SetSPIFrequency	9
3.4.2 SPIReset	9
3.4.3 SPIRead	9
3.4.4 SPIWrite	9
3.4.5 SPIWriteRead.....	9
4. I2C API Function Call Flow Diagram	10
5. SPI API Function Call Flow Diagram	11
6. UART API Function Call Flow Diagram.....	12

Revision History

Revision	Description	Date
0.4	➤ Modify Hid Device SDK Function.	May 15, 2019
0.5	➤ Modify I2C Function.	July 15, 2019
0.6	➤ Add GetVidPidSerialNumberByIndex Function.	Dec 04, 2019
0.7	➤ Modify UartRead Function.	Feb 24, 2020

1. Introduction

This HID to UART/I2C/SPI SDK API Programming Guide provides simple API (application programming interface) for developers to communicate with the PL23B3/23C3/23D3 HID to UART/I2C/SPI device. It provides how to enumerate HID devices and read/write HID Report Data.

Note that when using the library functions simultaneously in multiple threads, the developer must implement thread safety and call the library functions from within a critical section such that only one single function is call at any given time.

2. SDK Library File

PL23B3/23C3/23D3 SDK supported operating system and file description:

- Operating System : Windows XP and above
- Library : HidDeviceSdk.dll & HidDeviceSdk.lib(x86 & x64)
- Include File : HidDeviceSdkApi.h

3. SDK API Function

3.1 Enum Device API Function and Description

Function	Description
GetSDKVersion	Get SDK Version
EnumDeviceByVid	Get the number of Device Hid by using VID
EnumDeviceByVidPid	Get the number of Device Hid by using VID and PID
GetVidPidByIndex	Get VID and PID by using Hid Device index
OpenDeviceHandle	Open HID device
CloseDeviceHandle	Close HID device

3.1.1 GetSDKVersion

- Description: Get SDK Version, If Version is V1.0.0.4, it will return 0x01000004
- Syntax: int32_t WINAPI GetSDKVersion(uint32_t* SDKVersion)
- Parameters:
Output: SDKVersion, return SDK Version
- Return Value: 0 ➔ successfully

3.1.2 EnumDeviceByVid

- Description: Get the number of Device Hid by using VID
- Syntax: int32_t EnumDeviceByVid (uint32_t* HidDeviceCount, uint16_t VID)
- Parameters:
Output: HidDeviceCount return the number of Hid devices.
Input: VID (ex: 0x067B)
- Return Value: 0 ➔ successfully

3.1.3 EnumDeviceByVidPid

- Description: Get the number of Device Hid by using VID and PID
- Syntax: int32_t EnumDeviceByVidPid (uint32_t* HidDeviceCount, uint16_t VID, uint16_t PID)
- Parameters:
Output: HidDeviceCount, return the number of Hid devices
Input: VID
Input: PID
- Return Value: 0 ➔ successfully

3.1.4 GetVidPidByIndex

- Description: Get VID and PID by using Hid Device index.
- Syntax: int32_t GetVidPidByIndex(uint32_t DeviceIndex, uint16_t* VID, uint16_t* PID)
- Parameters:
Input: DeviceIndex input Hid device index (0~n)
Output: VID
Output: PID
- Return Value: 0 ➔ successfully

3.1.5 GetVidPidSerialNumberByIndex

- Description: Get VID/PID/Serial Number by using Hid Device index.
- Syntax: int32_t GetVidPidSerialNumberByIndex (uint32_t DeviceIndex, uint16_t* VID, uint16_t* PID, wchar_t* SerialNumber, uint8_t Length)
- Parameters:
 - Input: DeviceIndex input Hid device index (0~n)
 - Output: VID
 - Output: PID
 - Output: Serial Number String
 - Input: Buffer Length(bytes)
- Return Value: 0 ➔ successfully

3.1.6 OpenDeviceHandle

- Description: Open HID device
- Syntax: int32_t OpenDeviceHandle(uint32_t DeviceIndex, HANDLE* hDeviceHandle)
- Parameters:
 - Input: DeviceIndex input Hid device index (0~n)
 - Output: hDeviceHandle
- Return Value: 0 ➔ successfully

3.1.7 CloseDeviceHandle

- Description: Close HID device.
- Syntax: int32_t CloseDeviceHandle(HANDLE hDeviceHandle)
- Parameters:
 - Input: hDeviceHandle
- Return Value: 0 ➔ successfully

3.2 UART API Function and Description

Function	Description
GetUartConfig	Get Uart config
SetUartConfig	Set Uart config
UartRead	Read Uart data
UartWrite	Write Uart data
SetXonXoffSymbol	Set Software Flow Control byte (Xon/Xoff) Symbol
UartReset	Reset Uart Interface

3.2.1 GetUartConfig

- Description: Get Uart config
- Syntax: int32_t GetUartConfig(HANDLE hDeviceHandle, uint32_t* BaudRate, UART_STOP_BIT* StopBit, UART_PARITY_TYPE* ParityType, UART_DATA_BIT* Databit, UART_FLOW_CONTROL* FlowControl)
- Parameters:
 - Input: hDeviceHandle
 - Output: BaudRate. Baud Rate, unit in bps
 - Output: StopBit, Stop Bit (1/1.5/2)
 - Output: ParityType, Parity Type (None/Odd/Even/Mark/Space)
 - Output: Databit, Data Bits (5/6/7/8)
 - Output: FlowControl, FlowControl
- Return Value: 0 ➔ successfully

3.2.2 SetUartConfig

- Description: Set Uart config include BaudRate, StopBit, Parity, DataBits and FlowControl
- Syntax: int32_t SetUartConfig(HANDLE hDeviceHandle, uint32_t BaudRate, UART_STOP_BIT StopBit, UART_PARITY_TYPE ParityType, UART_DATA_BIT Databit, UART_FLOW_CONTROL FlowControl)
- Parameters:
 - Input: hDeviceHandle, enter OpenDeviceHandle get hDeviceHandle
 - Input: BaudRate. BaudRate, unit in bps, maximum 12Mbps
 - Input: StopBit, Stop Bit(1/1.5/2)
 - Input: ParityType, Parity Type(None/Odd/Even/Mark/Space)

Input: Databit, Data Bits(5/6/7/8)

Input: FlowControl, FlowControl Mode

- Return Value: 0 ➔ successfully

3.2.3 UartRead

- Description: Read Uart data
- Syntax: int32_t UartRead (HANDLE hDeviceHandle, uint8_t * Buffer, uint32_t NumberOfBytesToRead, uint32_t * NumberOfBytesRead, uint32_t TimeOutms)
- Parameters:
 - Input: hDeviceHandle
 - Output: Buffer, read data buffer
 - Input: NumberOfBytesToWrite, number of data to read
 - Output: NumberOfBytesWritten, actual data read
 - Input: TimeOutms, timeout, unit in millisecond
- Return Value: 0 ➔ successfully

3.2.4 UartWrite

- Description: Write Uart data
- Syntax: int32_t UartWrite(HANDLE hDeviceHandle, uint8_t * Buffer, uint32_t NumberOfBytesToWrite, uint32_t * NumberOfBytesWritten, uint32_t TimeOutms)
- Parameters:
 - Input: hDeviceHandle
 - Input: Buffer, write data buffer
 - Input: NumberOfBytesToWrite, number of data to be written
 - Output: NumberOfBytesWritten, actual data written
 - Input: TimeOutms, timeout, unit in millisecond
- Return Value: 0 ➔ successfully

3.2.5 SetXonXoffSymbol

- Description: Set Software Flow Control byte (Xon/Xoff Symbol)
- Syntax: int32_t SetXonXoffSymbol(HANDLE hDeviceHandle, uint8_t Xon, uint8_t Xoff)
- Parameters:
 - Input: hDeviceHandle
 - Input: Xon
 - Input: Xoff
- Return Value: 0 ➔ successfully

3.2.6 UartReset

- Description: Reset Uart Interface
- Syntax: int32_t UartReset(HANDLE hDeviceHandle)
- Parameters:
 - Input: hDeviceHandle
- Return Value: 0 ➔ successfully

3.3 I2C API Function and Description

Function	Description
SetI2CDeviceAddress	Set I2C Device Address
SetI2CFrequency	Set I2C frequency
I2CReset	Reset I2C Master Interface
I2CRead	Read I2C data
I2CWrite	Write I2C data
I2CWriteRead	Write and read I2C data

3.3.1 SetI2CDeviceAddress

- Description: Set I2C Device Address
- Syntax: int32_t SetI2CDeviceAddress(HANDLE hDeviceHandle, uint8_t DeviceAddress)
- Parameters:
 - Input: hDeviceHandle
 - Input: DeviceAddress I2C Device Address, (ex: input 0xA0)

- Return Value: 0 ➔ successfully

3.3.2 SetI2CFrequency

- Description: Set I2C frequency
- Syntax: int32_t SetI2CFrequency(HANDLE hDeviceHandle, uint8_t FreqDiv)
- Parameters:
 - Input: hDeviceHandle
 - Input: FreqDiv, I2C frequency divisor
 - I2C Frequency (KHz) = 2400k/FreqDiv, FreqDiv need more or equal to 4
- Return Value: 0 ➔ successfully

3.3.3 I2CReset

- Description: Reset I2C Master Interface
- Syntax: int32_t (HANDLE hDeviceHandle)
- Parameters:
 - Input: hDeviceHandle
- Return Value: 0 ➔ successfully

3.3.4 I2CRead

- Description: Read I2C data
- Syntax: int32_t I2CRead (HANDLE hDeviceHandle, uint8_t* Buffer, uint16_t NumberOfBytesToRead, uint16_t* NumberOfBytesRead, uint32_t TimeOutms)
- Parameters:
 - Input: hDeviceHandle
 - Input: DeviceAddr I2C Device Address, (ex: Input 0xA0)
 - Output: Buffer, read data buffer
 - Input: NumberOfBytesToRead, number of data to read
 - Output: NumberOfBytesRead, actual data read
 - Input: TimeOutms, timeout, unit in millisecond
- Return Value: 0 ➔ successfully

3.3.5 I2CWrite

- Description: Write I2C data
- Syntax: int32_t I2CWrite (HANDLE hDeviceHandle, uint8_t* Buffer, uint16_t NumberOfBytesToWrite, uint16_t* NumberOfBytesWritten, uint32_t TimeOutms)
- Parameters:
 - Input: hDeviceHandle
 - Input: DeviceAddr I2C Device Address
 - Input: Buffer, write data buffer
 - Input: NumberOfBytesToWrite, number of data to be written
 - Output: NumberOfBytesWritten, actual data written
 - Input: TimeOutms, timeout, unit in millisecond
- Return Value: 0 ➔ successfully

3.3.6 I2CWriteRead

- Description: Write and read I2C data
- Syntax: int32_t I2CWriteRead(HANDLE hDeviceHandle, uint8_t* WriteBuffer, uint16_t NumberOfBytesToWrite, BYTE* byReadBuffer, WORD NumberOfBytesToRead, WORD* nNumberOfBytesUse, uint32_t TimeOutms)
- Parameters:
 - Input: hDeviceHandle
 - Input: DeviceAddr I2C Device Address
 - Input: WriteBuffer, write data buffer
 - Input: NumberOfBytesToWrite, number of data to be written
 - Output: byReadBuffer, read data buffer
 - Input: NumberOfBytesToRead, number of data to read
 - Output: nNumberOfBytesUse, actual data use
 - Input: TimeOutms, timeout, unit in millisecond
- Return Value: 0 ➔ successfully

3.4 SPI API Function and Description

Function	Description
SetSPIFrequency	Set SPI frequency
SPIReset	Reset SPI Interface
SPIRead	Read SPI data
SPIWrite	Write SPI data
SPIWriteRead	Write and read SPI data

3.4.1 SetSPIFrequency

- Description: Set SPI frequency
- Syntax: `int32_t SetI2CFrequency(HANDLE hDeviceHandle, BYTE FreqDiv, SPI_MODE spiMode)`
- Parameters:
 - Input: `hDeviceHandle`
 - Input: `FreqDiv`, SPI frequency divisor
 - SPI Frequency (KHz) = $24000k / (FreqDiv + 1)$, `FreqDiv` need more or equal to 3
 - Return Value: 0 → successfully
 - Input: `spiMode` → `SPI_MODE0`, `SPI_MODE1`, `SPI_MODE2`, `SPI_MODE3`

3.4.2 SPIReset

- Description: Reset SPI Interface
- Syntax: `int32_t SPIReset (HANDLE hDeviceHandle)`
- Parameters:
 - Input: `hDeviceHandle`
- Return Value: 0 → successfully

3.4.3 SPIRead

- Description: Read SPI data
- Syntax: `int32_t SPIRead SPIRead(HANDLE hDeviceHandle, SPI_SELECT SPISelect, uint8_t* Buffer, uint16_t NumberOfBytesToRead, uint16_t* NumberOfBytesRead, uint32_t TimeOutms)`
- Parameters:
 - Input: `hDeviceHandle`
 - Input: `SPISelect` → `CS0` or `CS1`
 - Output: `Buffer`, read data buffer
 - Input: `NumberOfBytesToRead`, number of data to read
 - Output: `NumberOfBytesRead`, actual data read
 - Input: `TimeOutms`, timeout, unit in millisecond
- Return Value: 0 → successfully

3.4.4 SPIWrite

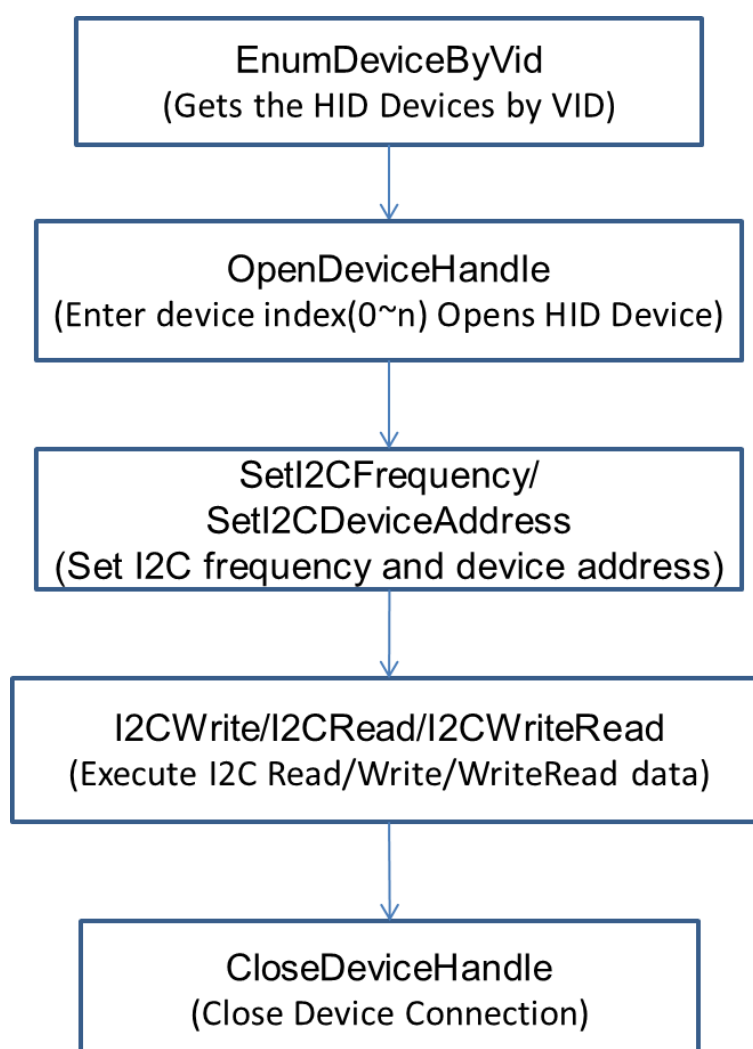
- Description: Write SPI data
- Syntax: `int32_t SPIWrite (HANDLE hDeviceHandle, SPI_SELECT SPISelect, uint8_t* Buffer, uint16_t NumberOfBytesToWrite, uint16_t* NumberOfBytesWritten, uint32_t TimeOutms)`
- Parameters:
 - Input: `hDeviceHandle`
 - Input: `SPISelect` → `CS0` or `CS1`
 - Input: `Buffer`, write data buffer
 - Input: `NumberOfBytesToWrite`, number of data to be written
 - Output: `NumberOfBytesWritten`, actual data written
 - Input: `TimeOutms`, timeout, unit in millisecond
- Return Value: 0 → successfully

3.4.5 SPIWriteRead

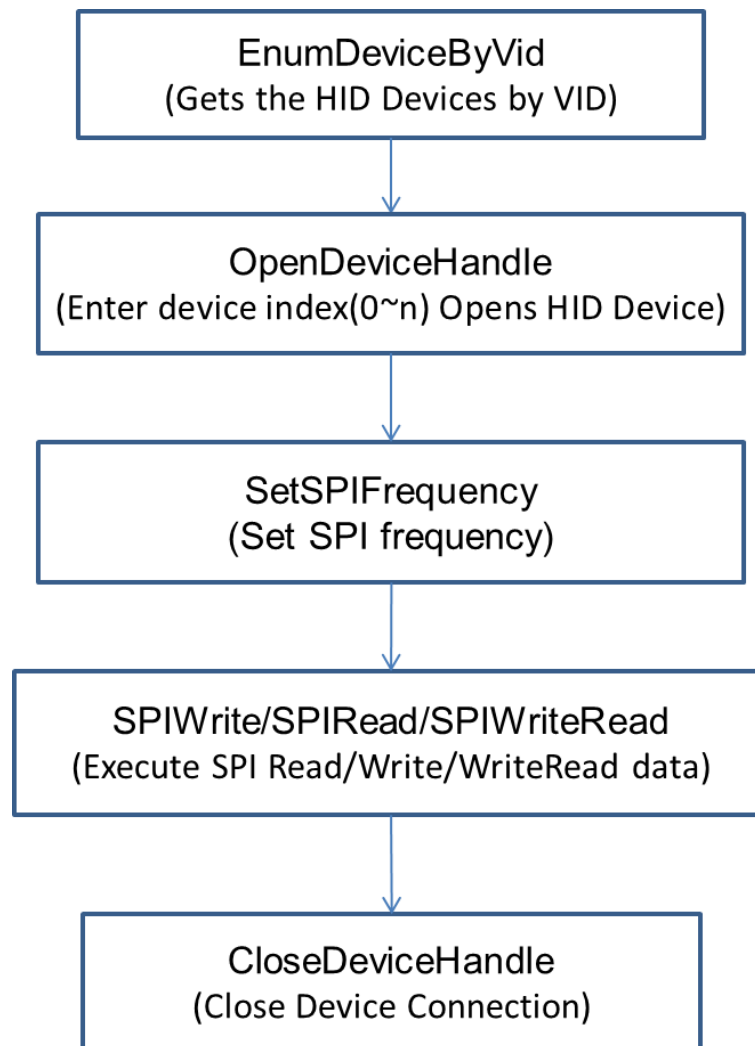
- Description: Write and read SPI data
- Syntax: `int32_t SPIWriteRead(HANDLE hDeviceHandle, SPI_SELECT nSelectSPI, uint8_t* WriteBuffer, uint16_t NumberOfBytesToWrite, BYTE* byReadBuffer, WORD NumberOfBytesToRead, WORD* nNumberOfBytesUse, uint32_t TimeOutms)`
- Parameters:
 - Input: `hDeviceHandle`
 - Input: `nSelectSPI` → `CS0` or `CS1`

Input: WriteBuffer, write data buffer
 Input: NumberOfBytesToWrite, number of data to be written
 Output: byReadBuffer, read data buffer
 Input: NumberOfBytesToRead, number of data to read
 Output: nNumberOfBytesUse, actual data use
 Input: TimeOutms, timeout, unit in millisecond
 ➤ Return Value: 0 → successfully

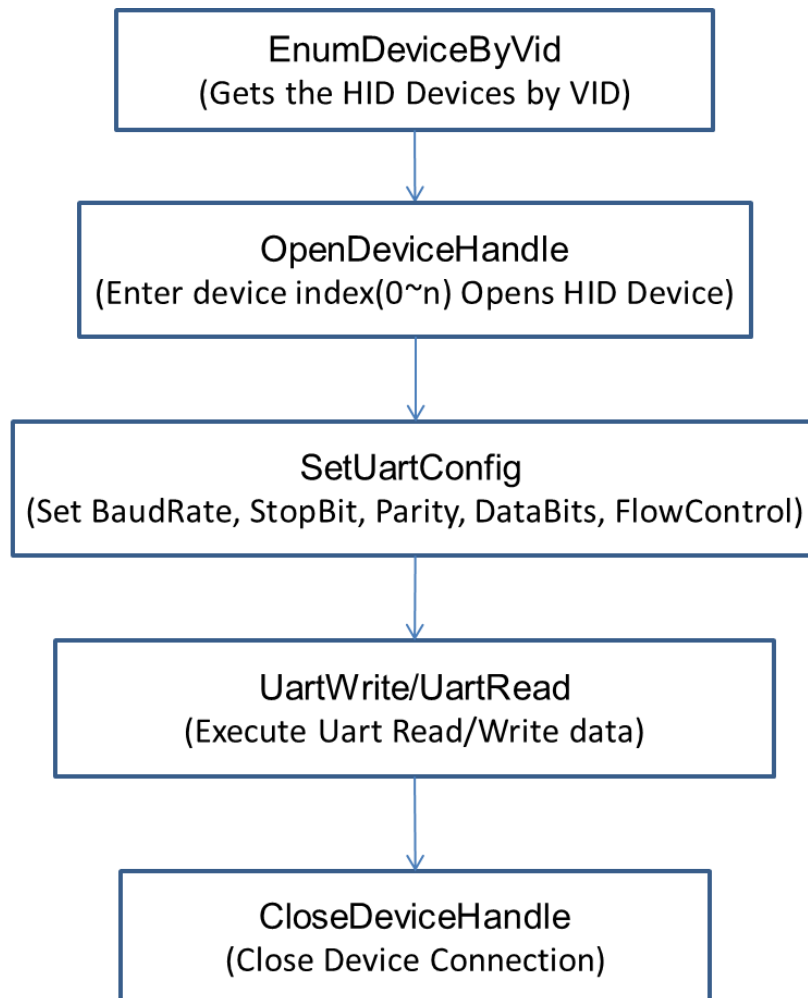
4. I2C API Function Call Flow Diagram



5. SPI API Function Call Flow Diagram



6. UART API Function Call Flow Diagram



Disclaimer

All the information in this document is subject to change without prior notice. Prolific Technology Inc. does not make any representations or any warranties (implied or otherwise) regarding the accuracy and completeness of this document and shall in no event be liable for any loss of profit or any other commercial damage, including but not limited to special, incidental, consequential, or other damages.

Trademarks

The Prolific logo is a registered trademark of Prolific Technology Inc. All brand names and product names used in this document are trademarks or registered trademarks of their respective holders.

Copyrights

Copyright © 2016 Prolific Technology Inc. All rights reserved.

No part of this document may be reproduced or transmitted in any form by any means without the express written permission of Prolific Technology Inc.