



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

Document Revision: 1.0
Document Release: July 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

1. Introduction	3
2. SDK Library File	4
3. SDK API Function	4
3.1 Enumerate Device API Function and Description	4
3.1.1 GetSDKVersion	4
3.1.2 InitDevice	4
3.1.3 ExitDevice	4
3.1.4 EnumDeviceByVid	5
3.1.5 EnumDeviceByVidPid	5
3.1.6 GetVidPidByIndex	5
3.1.7 GetVidPidByHandle	5
3.1.8 GetVidPidSerialNumberByIndex	6
3.1.9 OpenDeviceHandle	6
3.1.10 CloseDeviceHandle	6
3.2 UART API Function and Description	7
3.2.1 GetUartConfig	7
3.2.2 SetUartConfig	7
3.2.3 UartRead	8
3.2.4 UartWrite	8
3.2.5 SetXonXoffSymbol	8
3.2.6 UartReset	8
3.3 I2C API Function and Description	9
3.3.1 SetI2CDeviceAddress	9
3.3.2 SetI2CFrequency	9
3.3.3 I2CReset	9
3.3.4 I2CRead	10
3.3.5 I2CWrite	10
3.3.6 I2CWriteRead	10
3.4 SPI API Function and Description	11
3.4.1 SetSPIFrequency	11
3.4.2 SPIRead	11
3.4.3 SPIWrite	12
3.4.4 SPIWriteRead	12
3.4.5 SPIReset	12
4. I2C API Function Call Flow Diagram	13
5. SPI API Function Call Flow Diagram	14
6. UART API Function Call Flow Diagram	15

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/PL23C3/PL23D3 SDK supported operating system and file description:

- Operation System : Linux related operation system
- Library : libHidDeviceSdk.so
- Header file : HidDeviceSdkApi.h

3. SDK API Function

3.1 Enumerate Device API Function and Description

Function	Description
GetSDKVersion	Get SDK Version
InitDevice	Initial Hid Device
ExitDevice	Exit Hid Device
EnumDeviceByVid	Get the number of Hid Device by using VID
EnumDeviceByVidPid	Get the number of Hid Device by using VID and PID
GetVidPidByIndex	Get VID and PID by using Hid Device index
GetVidPidByHandle	Get VID and PID by using the handle of Hid Device
GetVidPidSerialNumberByIndex	Get serial number string by using Hid Device index
OpenDeviceHandle	Open the handle of Hid Device
CloseDeviceHandle	Close the handle of Hid Device

3.1.1 GetSDKVersion

- Description: Get SDK Version, If Version is V1.0.0.1, it will return 0x01000001
- Syntax: void GetSDKVersion(uint32_t* SdkVersion)
- Parameters:
Output: SDKVersion

3.1.2 InitDevice

- Description: Initial Hid Device
- Syntax: int32_t InitDevice(void)
- Parameters:
- Return Value: 0 ➔ successfully

3.1.3 ExitDevice

- Description: Exit Hid Device
- Syntax: int32_t ExitDevice(void)
- Parameters:
- Return Value: 0 ➔ successfully

3.1.4 EnumDeviceByVid

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

3.1.5 EnumDeviceByVidPid

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

3.1.6 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, Hid Device index (0~n)
 - Output: VID
 - Output: PID
- Return Value: 0 ➔ successfully

3.1.7 GetVidPidByHandle

- Description: Get VID and PID by using the handle of Hid Device
- Syntax: int32_t GetVidPidByHandle(HANDLE hDeviceHandle, uint16_t *VID, uint16_t *PID)
- Parameters:
 - Input: hDeviceHandle, the handle of Hid Device
 - Output: VID
 - Output: PID
- Return Value: 0 ➔ successfully

3.1.8 GetVidPidSerialNumberByIndex

- Description: Get serial number string by using Hid Device index
- Syntax: `int32_t GetVidPidSerialNumberByIndex(uint32_t DeviceIndex, uint16_t *VID, uint16_t *PID, wchar_t* SerialNumber, uint16_t length)`
- Parameters:
 - Input: DeviceIndex, Hid Device index (0~n)
 - Output: VID
 - Output: PID
 - Output: SerialNumber
 - Input: length, the length of SerialNumber that have to smaller than 256
- Return Value: 0 ➔ successfully

3.1.9 OpenDeviceHandle

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

3.1.10 CloseDeviceHandle

- Description: Close the handle of Hid Device
- Syntax: `int32_t CloseDeviceHandle(HANDLE hDeviceHandle)`
- Parameters:
 - Input: hDeviceHandle
- Return Value: 0 ➔ successfully

3.2 UART API Function and Description

Function call	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, uint8_t *StopBit, uint8_t *ParityType, uint8_t *DataBit, uint8_t *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 Mode
- Return Value: 0 ➔ successfully

3.2.2 SetUartConfig

- Description: Set Uart config
- Syntax: `int32_t SetUartConfig(HANDLE hDeviceHandle, uint32_t BaudRate, enum UART_STOP_BIT StopBit, enum UART_PARITY_TYPE ParityType, enum UART_DATA_BIT DataBit, enum UART_FLOW_CONTROL FlowControl)`
- Parameters:
 - Input: 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, uint16_t NumberOfBytesToRead, uint16_t *NumberOfBytesRead, uint32_t TimeOutms)`
- Parameters:
 - Input: `hDeviceHandle`
 - 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

3.2.4 UartWrite

- Description: Write Uart data
- Syntax: `int32_t UartWrite(HANDLE hDeviceHandle, uint8_t *Buffer, uint16_t NumberOfBytesToWrite, uint16_t *NumberOfBytesWritten, uint32_t TimeOutms)`
- Parameters:
 - Input: `hDeviceHandle`
 - Input: `Buffer`, write data buffer
 - Input: `NumberOfBytesToWrite`, number of data to be write
 - 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: Rest UART
- Syntax: `int32_t ResetUART(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: DeviceAddr I2C Device Address, for example 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) = 24000k/FreqDiv, FreqDiv need more or equal to 4
- Return Value: 0 ➔ successfully

3.3.3 I2CReset

- Description: Reset I2C Master Interface
- Syntax: `int32_t I2CReset(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`
 - 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: `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, uint8_t *ReadBuffer, uint16_t NumberOfBytesToRead, uint16_t *NumberOfBytesRead, uint32_t TimeOutms)`
- Parameters:
 - Input: `hDeviceHandle`
 - Input: `WriteBuffer`, write data buffer
 - Input: `NumberOfBytesToWrite`, number of data to be written
 - Input: `ReadBuffer`, 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 SPI API Function and Description

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

3.4.1 SetSPIFrequency

- Description: Set SPI frequency
- Syntax: int32_t SetSPIFrequency(HANDLE hDeviceHandle, uint8_t FreqDiv, enum SPI_MODE spiMode)
- Parameters:
 - Input: hDeviceHandle
 - Input: FreqDiv, SPI frequency divisor
 - SPI Frequency (KHz) = 24000k/ (FreqDiv+1), FreqDiv need more or equal to 4
 - Input: spiMode, SPI_MODE0, SPI_MODE1, SPI_MODE2, SPI_MODE3
- Return Value: 0 ➔ successfully

3.4.2 SPIRead

- Description: Read SPI data
- Syntax: int32_t SPIRead(HANDLE hDeviceHandle, enum SPI_SELECT SelectSPI, uint8_t *Buffer, uint16_t NumberOfBytesToRead, uint16_t *NumberOfBytesRead, uint32_t TimeOutms)
- Parameters:
 - Input: hDeviceHandle
 - Input: SelectSPI ➔ 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

3.4.3 SPIWrite

- Description: Write SPI data
- Syntax: int32_t SPIWrite(HANDLE hDeviceHandle, enum SPI_SELECT SelectSPI, uint8_t *Buffer, uint16_t NumberOfBytesToWrite, uint16_t *NumberOfBytesWritten, uint32_t TimeOutms)
- Parameters:
 - Input: hDeviceHandle
 - Input: SelectSPI → 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

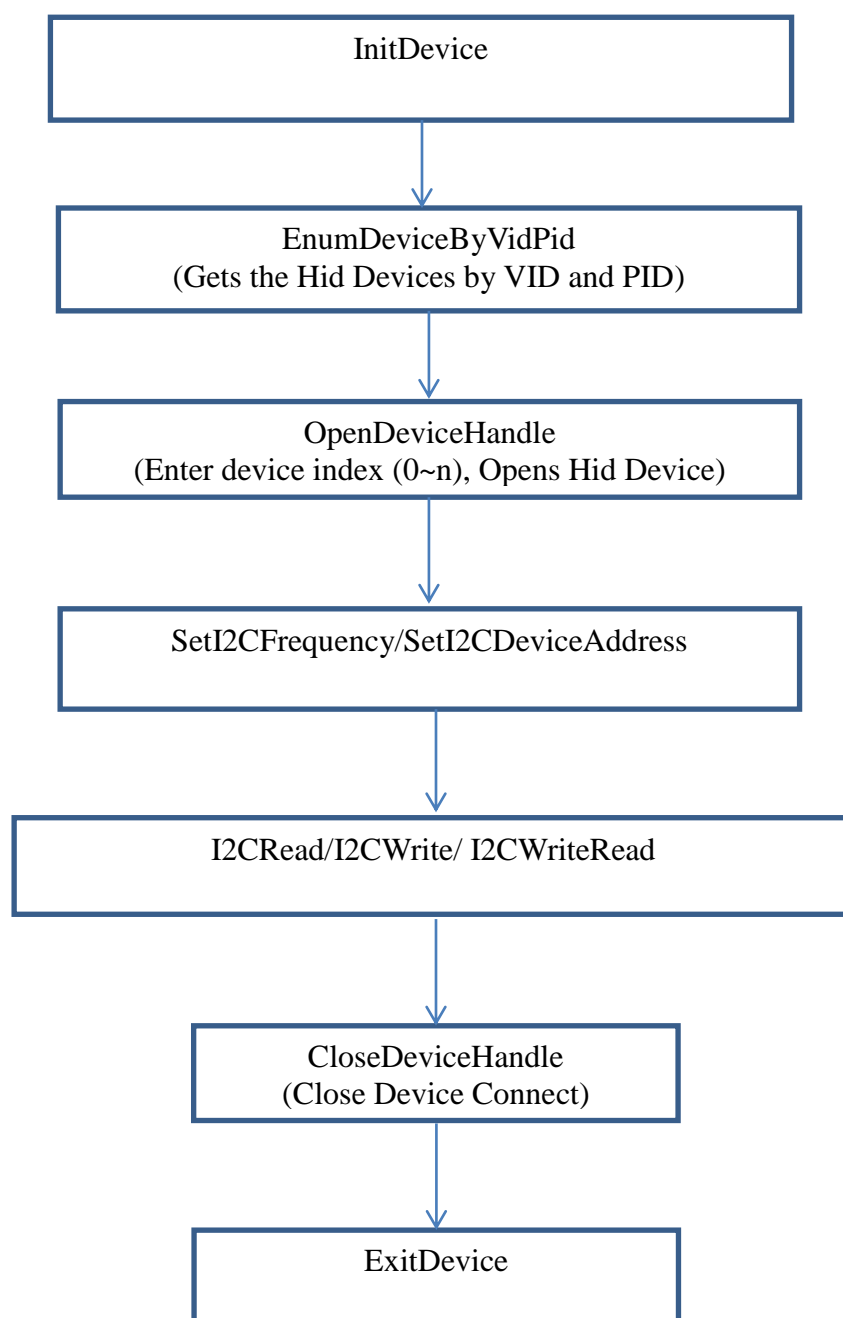
3.4.4 SPIWriteRead

- Description: Write and read SPI data
- Syntax: int32_t SPIWriteRead(HANDLE hDeviceHandle, enum SPI_SELECT SelectSPI, uint8_t *WriteBuffer, uint16_t NumberOfBytesToWrite, uint8_t *ReadBuffer, uint16_t NumberOfBytesToRead, uint16_t *NumberOfBytesRead, uint32_t TimeOutms)
- Parameters:
 - Input: hDeviceHandle
 - Input: SelectSPI → CS0 or CS1
 - Input: WriteBuffer, write data buffer
 - Input: NumberOfBytesToWrite, number of data to be written
 - Input: ReadBuffer, read data buffer
 - Input: NumberOfBytesToRead, number of data to read
 - Output: NumberOfBytesRead, actual data use
 - Input: TimeOutms, timeout, unit in millisecond
- Return Value: 0 → successfully

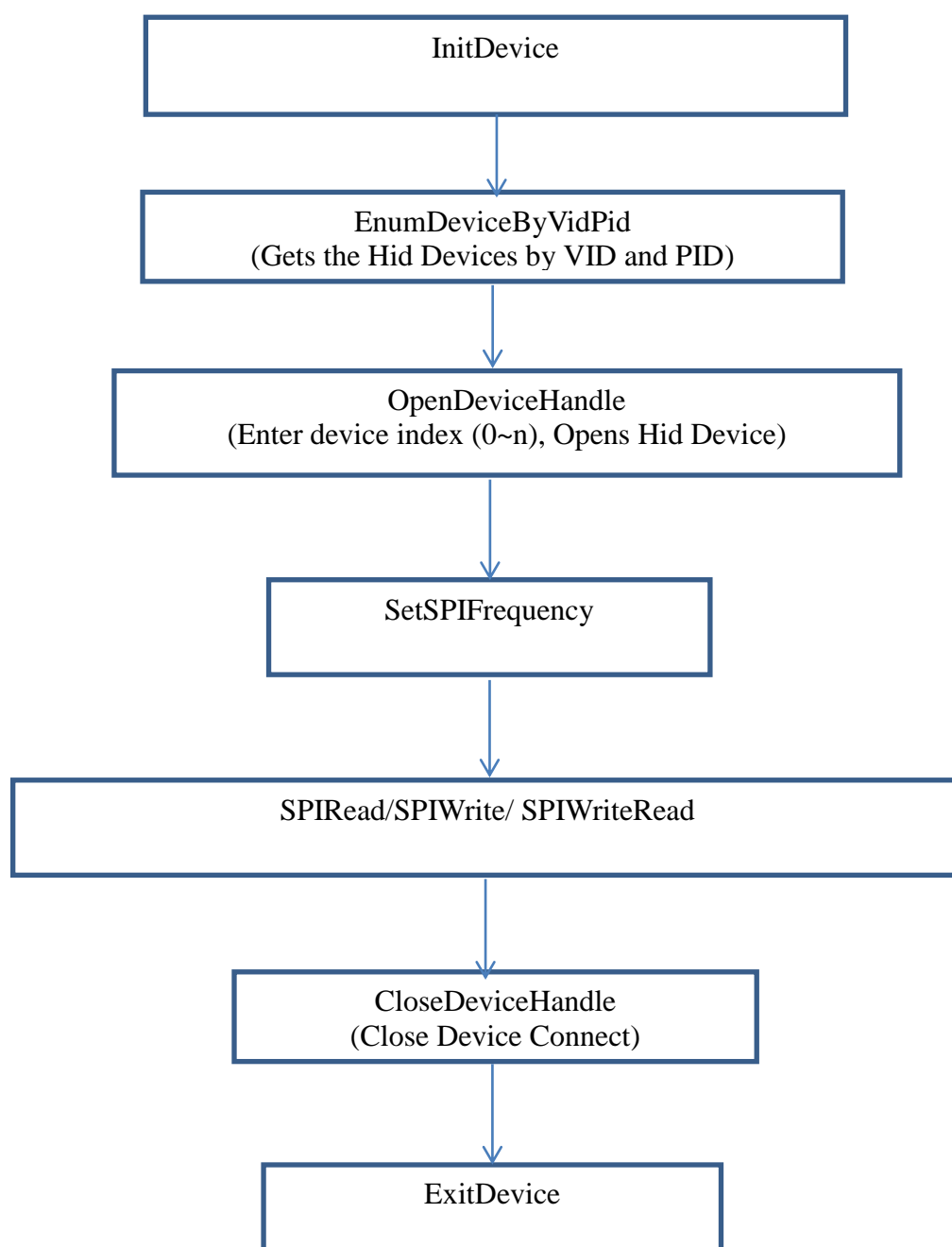
3.4.5 SPIReset

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

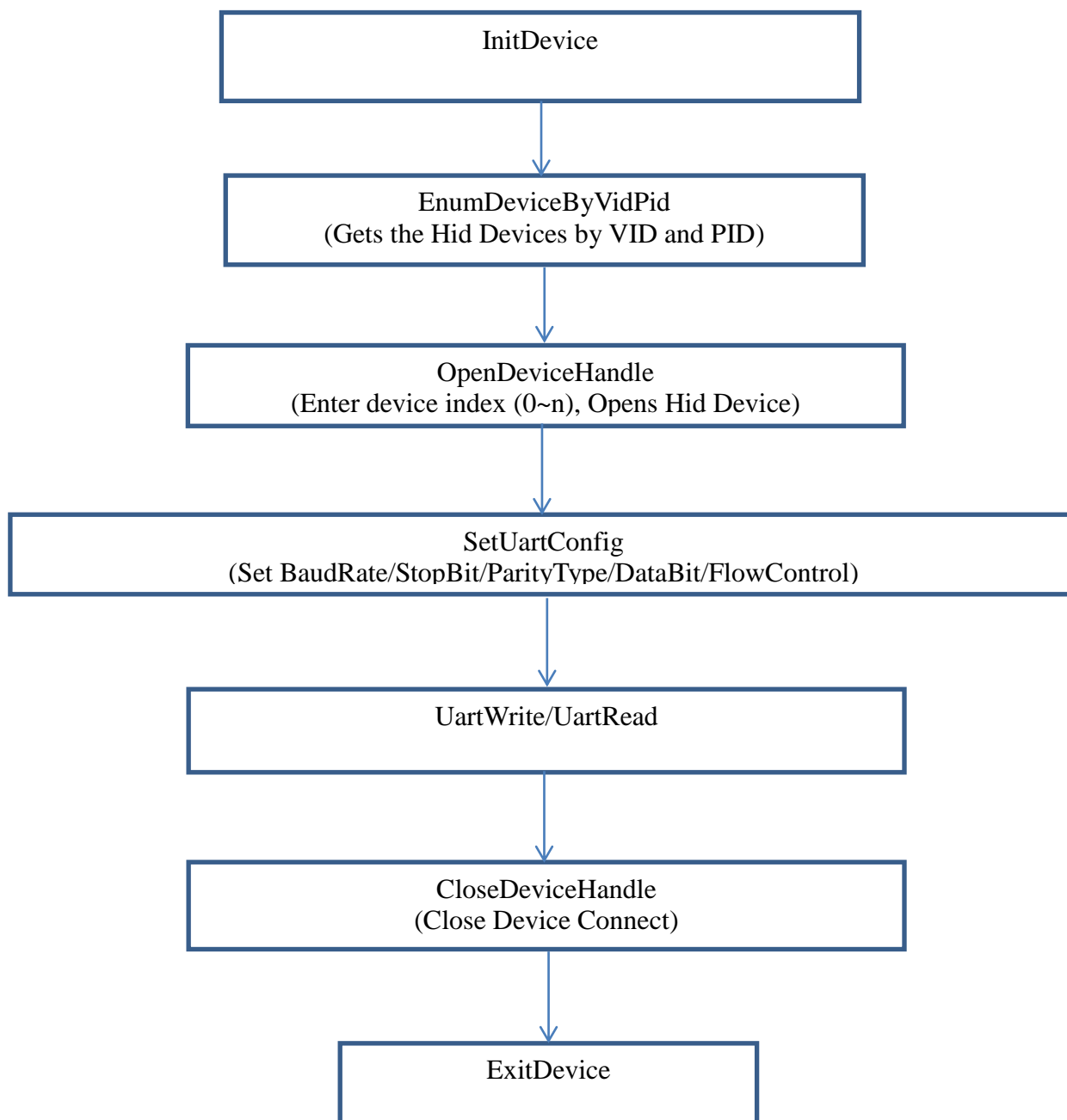
4. I2C API Function Call Flow Diagram



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.