



PL23B3/23C3/23D3 HID to UART/I2C/SPI Windows SDK 使用手冊

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>

目 錄

Revision History	3
1. 簡介	4
2. SDK 檔案	5
3. SDK 函數	5
3.1 列舉裝置相關函數名稱及說明	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 相關函數名稱及說明	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 相關函數名稱及說明	7
3.3.1 SetI2CDeviceAddress	8
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 相關函數名稱及說明	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	10
4. I2C API 呼叫流程	10
5. SPI API 呼叫流程	11
6. UART API 呼叫流程	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. 簡介

此 HID to UART/I2C/SPI SDK 提供相關應用程式介面用來與 PL23B3/23C3/23D3 HID Mode 溝通。主要提供列舉 HID 裝置與讀寫 HID Report Data。若要用在多執行緒(Multi-Thread)，會同時呼叫相關函式，則請加上 Critical Section 來呼叫，用來實現執行緒安全(Thread Safety)。

2. SDK 檔案

PL23B3/23C3/23D3 SDK 支援作業系統及檔案說明如下：

- 作業系統：Windows XP(含)以上
- 程式庫：HidDeviceSdk.dll & HidDeviceSdk.lib(x86 & x64)
- 標題檔：HidDeviceSdkApi.h

3. SDK 函數

3.1 列舉裝置相關函數名稱及說明

函數名稱	說明
GetSDKVersion	取得 SDK 版本
EnumDeviceByVid	利用 VID 取得 Device Hid 的數量
EnumDeviceByVidPid	利用 VID 及 PID 取得 Device Hid 的數量
GetVidPidByIndex	取得 Hid Device 某個索引的 VID PID
OpenDeviceHandle	連接 HID 裝置
CloseDeviceHandle	關閉 HID 裝置

3.1.1 GetSDKVersion

- 敘述:取得 SDK 版本, 如版號為 V1.0.0.4, 會回傳 0x01000004
- 語法: int32_t WINAPI GetSDKVersion(uint32_t* SDKVersion)
- 參數: 輸出 SDKVersion
- 回傳值: 0 ➔ 成功

SDK 版本也可由檔案的內容取得

3.1.2 EnumDeviceByVid

- 敘述: 利用 VID 取得 Device Hid 的數量
- 語法: int32_t EnumDeviceByVid (uint32_t* HidDeviceCount, uint16_t VID)
- 參數:
輸出: HidDeviceCount 傳回 Hid 裝置數量
輸入: VID 如 0x067B
- 回傳值: 0 ➔ 成功

3.1.3 EnumDeviceByVidPid

- 敘述: 利用 VID 及 PID 取得 Device Hid 的數量
- 語法: int32_t EnumDeviceByVidPid (uint32_t* HidDeviceCount, uint16_t VID, uint16_t PID)
- 參數:
輸出: HidDeviceCount 傳回 Hid 裝置數量
輸入: VID
輸入: PID
- 回傳值: 0 ➔ 成功

3.1.4 GetVidPidByIndex

- 敘述: 取得 Hid Device 某個索引的 VID PID
- 語法: int32_t GetVidPidByIndex(uint32_t DeviceIndex, uint16_t* VID, uint16_t* PID)
- 參數:
輸入: DeviceIndex 傳入 Hid 裝置索引(0~n)
輸出: VID
輸出: PID
- 回傳值: 0 ➔ 成功

3.1.5 GetVidPidSerialNumberByIndex

- 敘述: 取得 Hid Device 某個索引的 VID/PID/Serial Number String
- 語法: `int32_t GetVidPidSerialNumberByIndex(uint32_t DeviceIndex, uint16_t* VID, uint16_t* PID, wchar_t* SerialNumber, uint8_t Length)`
- 參數:
 - 輸入: DeviceIndex 傳入 Hid 裝置索引(0~n)
 - 輸出: VID
 - 輸出: PID
 - 輸出: Serial Number String
 - 輸入: Buffer Length(Bytes)
- 回傳值: 0 ➔ 成功

3.1.6 OpenDeviceHandle

- 敘述: 連接 HID 裝置
- 語法: `int32_t OpenDeviceHandle(uint32_t DeviceIndex, HANDLE* hDeviceHandle)`
- 參數:
 - 輸入: DeviceIndex 傳入 Hid 裝置索引(0~n)
 - 輸出: hDeviceHandle
- 回傳值: 0 ➔ 成功

3.1.7 CloseDeviceHandle

- 敘述: 關閉 HID 裝置
- 語法: `int32_t CloseDeviceHandle(HANDLE hDeviceHandle)`
- 參數:
 - 輸入: hDeviceHandle
- 回傳值: 0 ➔ 成功

3.2 UART 相關函數名稱及說明

函數名稱	說明
GetUartConfig	取得 Uart 設定值
SetUartConfig	設定 Uart 設定值
UartRead	讀取 Uart data
UartWrite	寫入 Uart data
SetXonXoffSymbol	設定 Software Flow Control byte (Xon/Xoff) Symbol
UartReset	重置 Uart Interface

3.2.1 GetUartConfig

- 敘述: 取得 SDK 版本
- 語法: `int32_t GetUartConfig(HANDLE hDeviceHandle, uint32_t* BaudRate, UART_STOP_BIT* StopBit, UART_PARITY_TYPE* ParityType, UART_DATA_BIT* Databit, UART_FLOW_CONTROL* FlowControl)`
- 參數:
 - 輸入: hDeviceHandle
 - 輸出: BaudRate. Baud Rate, 單位為 bps
 - 輸出: StopBit, Stop Bit(1/1.5/2)
 - 輸出: ParityType, Parity Type(None/Odd/Even/Mark/Space)
 - 輸出: Databit, Data Bits(5/6/7/8)
 - 輸出: FlowControl, FlowControl
- 回傳值: 0 ➔ 成功

3.2.2 SetUartConfig

- 敘述: 設定 Uart 設定值, 包含 BaudRate, StopBit, Parity, DataBits, FlowControl
- 語法: `int32_t SetUartConfig(HANDLE hDeviceHandle, uint32_t BaudRate, UART_STOP_BIT StopBit, UART_PARITY_TYPE ParityType, UART_DATA_BIT Databit, UART_FLOW_CONTROL FlowControl)`
- 參數:

輸入: hDeviceHandle 輸入 OpenDeviceHandle 所得到的 hDeviceHandle
輸入: BaudRate. Baud Rate, 單位為 bps, 最快 12,000,000bps
輸入: StopBit, Stop Bit(1/1.5/2)
輸入: ParityType, Parity Type(None/Odd/Even/Mark/Space)
輸入: Databit, Data Bits(5/6/7/8)
輸入: FlowControl, FlowControl Mode

➤ 回傳值: 0 ➔ 成功

3.2.3 UartRead

- 敘述: 讀取 Uart data
- 語法: int32_t UartRead (HANDLE hDeviceHandle, uint8_t * Buffer, uint32_t NumberOfBytesToRead, uint32_t * NumberOfBytesRead, uint32_t TimeOutms)
- 參數:
 - 輸入: hDeviceHandle
 - 輸出: Buffer 讀取資料緩衝區
 - 輸入: NumberOfBytesToWrite 讀取資料的位元數
 - 輸出: NumberOfBytesWritten 實際讀取資料的位元數
 - 輸入: TimeOutms 逾時時間, 單位為 milliSecond
- 回傳值: 0 ➔ 成功

3.2.4 UartWrite

- 敘述: 寫入 Uart data
- 語法: int32_t UartWrite(HANDLE hDeviceHandle, uint8_t * Buffer, uint32_t NumberOfBytesToWrite, uint32_t * NumberOfBytesWritten, uint32_t TimeOutms)
- 參數:
 - 輸入: hDeviceHandle
 - 輸入: Buffer 寫入的資料
 - 輸入: NumberOfBytesToWrite 寫入資料的位元數
 - 輸出: NumberOfBytesWritten 實際寫入資料的位元數
 - 輸入: TimeOutms 逾時時間, 單位為 millisecond
- 回傳值: 0 ➔ 成功

3.2.5 SetXonXoffSymbol

- 敘述: 設定 Software Flow Control byte (Xon/Xoff Symbol)
- 語法: int32_t SetXonXoffSymbol(HANDLE hDeviceHandle, uint8_t Xon, uint8_t Xoff)
- 參數:
 - 輸入: hDeviceHandle
 - 輸入: Xon
 - 輸入: Xoff
- 回傳值: 0 ➔ 成功

3.2.6 UartReset

- 敘述: 重置 Uart Interface
- 語法: int32_t UartReset(HANDLE hDeviceHandle)
- 參數:
 - 輸入: hDeviceHandle
- 回傳值: 0 ➔ 成功

3.3 I2C 相關函數名稱及說明

函數名稱	說明
SetI2CDeviceAddress	設定 I2C Device Address
SetI2CFrequency	設定 I2C 頻率
I2CReset	重置 I2C Master Interface
I2CRead	讀取 I2C data

I2CWrite	寫入 I2C data
I2CWriteRead	寫入並讀取 I2C data

3.3.1 SetI2CDeviceAddress

- 敘述: 設定 I2C Device Address
- 語法: `int32_t SetI2CDeviceAddress(HANDLE hDeviceHandle, uint8_t DeviceAddress)`
- 參數:
 - 輸入: `hDeviceHandle`
 - 輸入: `DeviceAddress` I2C Device Address, 如輸入 `0xA0`
- 回傳值: 0 ➔ 成功

3.3.2 SetI2CFrequency

- 敘述: 設定 I2C 頻率
- 語法: `int32_t SetI2CFrequency(HANDLE hDeviceHandle, uint8_t FreqDiv)`
- 參數:
 - 輸入: `hDeviceHandle`
 - 輸入: `FreqDiv` I2C 頻率除數
 - $I2C\ Frequency(KHz) = 24000 / FreqDiv$, `FreqDiv` 需大於等於 4
- 回傳值: 0 ➔ 成功

3.3.3 I2CReset

- 敘述: 重置 I2C Master Interface
- 語法: `int32_t (HANDLE hDeviceHandle)`
- 參數:
 - 輸入: `hDeviceHandle`
- 回傳值: 0 ➔ 成功

3.3.4 I2CRead

- 敘述: 讀取 I2C data
- 語法: `int32_t I2CRead (HANDLE hDeviceHandle, uint8_t DeviceAddr, uint8_t* Buffer, uint16_t NumberOfBytesToRead, uint16_t* NumberOfBytesRead, uint32_t TimeOutms)`
- 參數:
 - 輸入: `hDeviceHandle`
 - 輸入: `DeviceAddr` I2C Device Address, 如輸入 `0xA0`
 - 輸出: `Buffer` 讀取資料緩衝區
 - 輸入: `NumberOfBytesToRead` 讀取資料的位元數
 - 輸出: `NumberOfBytesRead` 實際讀取資料的位元數
 - 輸入: `TimeOutms` 逾時時間, 單位為 `milliSecond`
- 回傳值: 0 ➔ 成功

3.3.5 I2CWrite

- 敘述: 寫入 I2C data
- 語法: `int32_t I2CWrite (HANDLE hDeviceHandle, uint8_t DeviceAddr, uint8_t* Buffer, uint16_t NumberOfBytesToWrite, uint16_t* NumberOfBytesWritten, uint32_t TimeOutms)`
- 參數:
 - 輸入: `hDeviceHandle`
 - 輸入: `DeviceAddr` I2C Device Address
 - 輸入: `Buffer` 寫入的資料
 - 輸入: `NumberOfBytesToWrite` 寫入資料的位元數
 - 輸出: `NumberOfBytesWritten` 實際寫入資料的位元數
 - 輸入: `TimeOutms` 逾時時間, 單位為 `millisecond`
- 回傳值: 0 ➔ 成功

3.3.6 I2CWriteRead

- 敘述: 寫入並讀取 I2C data

- 語法: `int32_t I2CWriteRead(HANDLE hDeviceHandle, uint8_t DeviceAddr, uint8_t* WriteBuffer, uint16_t NumberOfBytesToWrite, BYTE* byReadBuffer, WORD NumberOfBytesToRead, WORD* nNumberOfBytesUse, uint32_t TimeOutms)`
- 參數:
 - 輸入: `hDeviceHandle`
 - 輸入: `DeviceAddr` I2C Device Address
 - 輸入: `WriteBuffer` 寫入的資料
 - 輸入: `NumberOfBytesToWrite` 寫入資料的位元數
 - 輸出: `byReadBuffer` 讀取資料緩衝區
 - 輸入: `NumberOfBytesToRead` 讀取資料的位元數
 - 輸出: `nNumberOfBytesUse` 實際讀取資料的位元數
 - 輸入: `TimeOutms` 逾時時間, 單位為 `milliSecond`
- 回傳值: 0 ➔ 成功

3.4 SPI 相關函數名稱及說明

函數名稱	說明
SetSPIFrequency	設定 SPI 頻率
SPIReset	重置 SPI Interface
SPIRead	讀取 SPI data
SPIWrite	寫入 SPI data
SPIWriteRead	寫入並讀取 SPI data

3.4.1 SetSPIFrequency

- 敘述: 設定 I2C 頻率
- 語法: `int32_t SetI2CFrequency(HANDLE hDeviceHandle, BYTE FreqDiv, SPI_MODE spiMode)`
- 參數:
 - 輸入: `hDeviceHandle`
 - 輸入: `FreqDiv` SPI 頻率除數
 - $\text{SPI Frequency(KHz)} = 24000 / (\text{FreqDiv} + 1)$, `nFreqDiv` 需大於等於 3, 頻率最快為 6000 KHz
 - 輸入: `spiMode` ➔ `SPI_MODE0`, `SPI_MODE1`, `SPI_MODE2`, `SPI_MODE3`
- 回傳值: 0 ➔ 成功

3.4.2 SPIReset

- 敘述: 重置 SPI Interface
- 語法: `int32_t SPIReset (HANDLE hDeviceHandle)`
- 參數:
 - 輸入: `hDeviceHandle`
- 回傳值: 0 ➔ 成功

3.4.3 SPIRead

- 敘述: 讀取 SPI data
- 語法: `int32_t SPIRead SPIRead(HANDLE hDeviceHandle, SPI_SELECT SPISelect, uint8_t* Buffer, uint16_t NumberOfBytesToRead, uint16_t* NumberOfBytesRead, uint32_t TimeOutms)`
- 參數:
 - 輸入: `hDeviceHandle`
 - 輸入: `SPISelect` ➔ `CS0` or `CS1`
 - 輸出: `Buffer` 讀取資料緩衝區
 - 輸入: `NumberOfBytesToRead` 讀取資料的位元數
 - 輸出: `NumberOfBytesRead` 實際讀取資料的位元數
 - 輸入: `TimeOutms` 逾時時間, 單位為 `milliSecond`
- 回傳值: 0 ➔ 成功

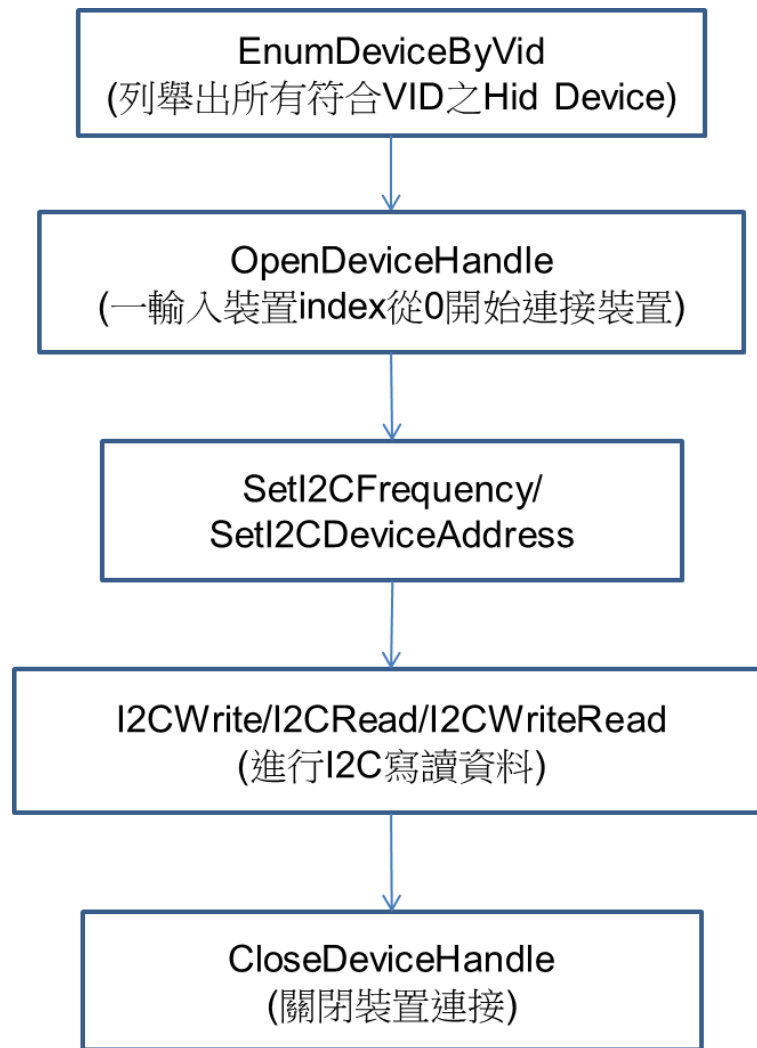
3.4.4 SPIWrite

- 敘述: 寫入 SPI data
- 語法: `int32_t SPIWrite (HANDLE hDeviceHandle, SPI_SELECT SPISelect, uint8_t* Buffer, uint16_t NumberOfBytesToWrite, uint16_t* NumberOfBytesWritten, uint32_t TimeOutms)`
- 參數:
 - 輸入: `hDeviceHandle`
 - 輸入: `SPISelect` → CS0 or CS1
 - 輸入: `Buffer` 寫入的資料
 - 輸入: `NumberOfBytesToWrite` 寫入資料的位元數
 - 輸出: `NumberOfBytesWritten` 實際寫入資料的位元數
 - 輸入: `TimeOutms` 逾時時間, 單位為 `millisecond`
- 回傳值: 0 → 成功

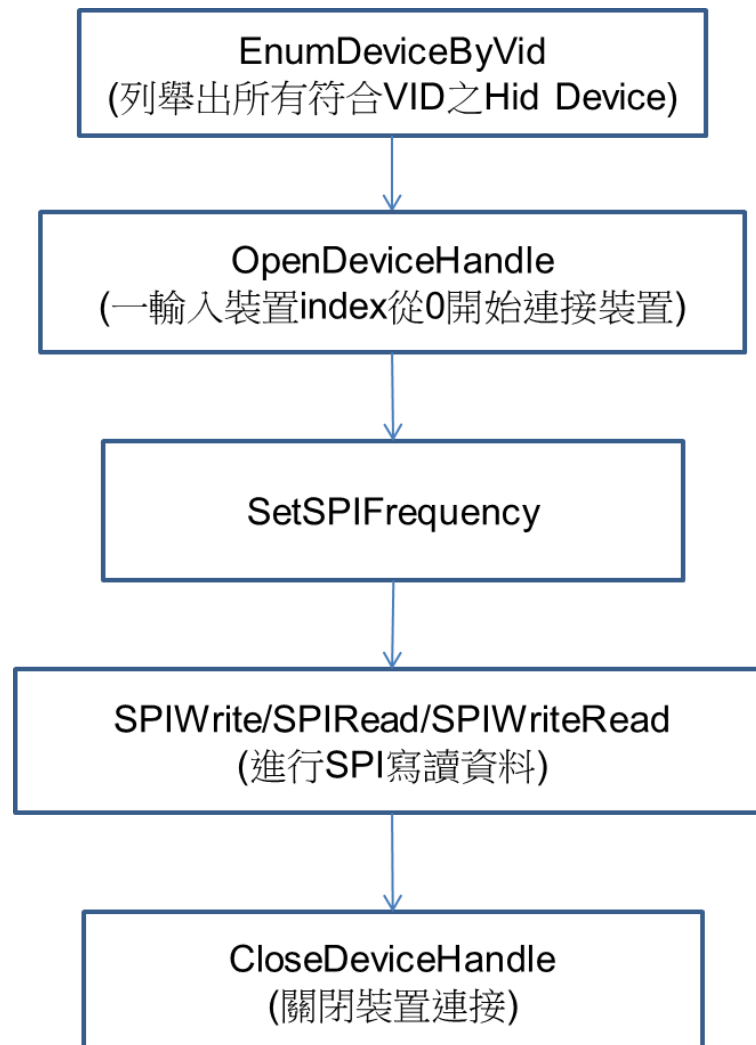
3.4.5 SPIWriteRead

- 敘述: 寫入並讀取 SPI data
- 語法: `int32_t SPIWriteRead(HANDLE hDeviceHandle, SPI_SELECT nSelectSPI, uint8_t* WriteBuffer, uint16_t NumberOfBytesToWrite, BYTE* byReadBuffer, WORD NumberOfBytesToRead, WORD* nNumberOfBytesUse, uint32_t TimeOutms)`
- 參數:
 - 輸入: `hDeviceHandle`
 - 輸入: `nSelectSPI` → CS0 or CS1
 - 輸入: `WriteBuffer` 寫入的資料
 - 輸入: `NumberOfBytesToWrite` 寫入資料的位元數
 - 輸出: `byReadBuffer` 讀取資料緩衝區
 - 輸入: `NumberOfBytesToRead` 讀取資料的位元數
 - 輸出: `nNumberOfBytesUse` 實際讀取資料的位元數
 - 輸入: `TimeOutms` 逾時時間, 單位為 `millisecond`
- 回傳值: 0 → 成功

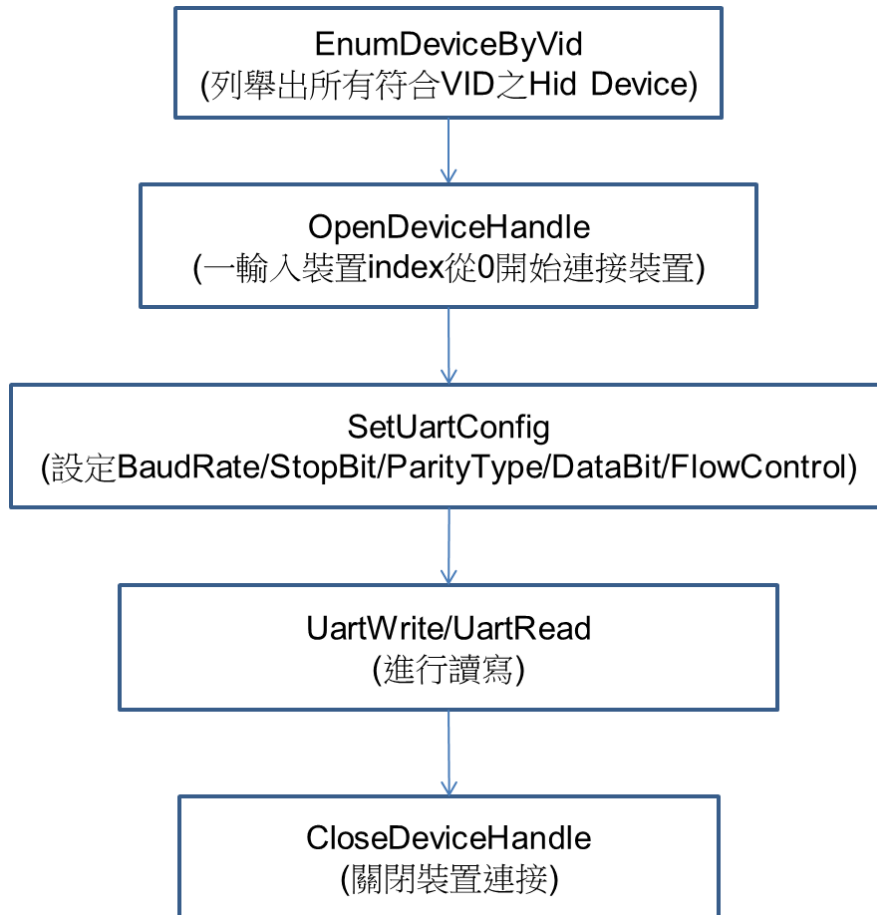
4. I2C API 呼叫流程



5. SPI API 呼叫流程



6. UART API 呼叫流程



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.