

---

# **CH347 应用开发手册**

**V1.4**

# 目录

一、简介.....	3
二、接口说明.....	3
三、同步串行接口.....	4
3.1 相关数据类型.....	4
3.1.1 SPI 控制器信息.....	4
3.1.2 设备信息.....	4
3.2 公共操作函数.....	5
3.2.1 CH347OpenDevice.....	5
3.2.2 CH347CloseDevice.....	5
3.2.3 CH347SetDeviceNotify.....	6
3.2.4 CH347GetDeviceInfor.....	6
3.2.5 CH347GetVersion.....	7
3.2.6 CH347SetTimeout.....	7
3.2.7 接口动态插拔检测.....	8
3.2.8 设备枚举操作.....	8
3.3 SPI 功能函数.....	9
3.3.1 操作流程.....	9
3.3.2 CH347SPI_Init.....	10
3.3.3 CH347SPI_SetDataBits.....	10
3.3.4 CH347SPI_GetCfg.....	10
3.3.5 CH347SPI_ChangeCS.....	11
3.3.6 CH347SPI_SetChipSelect.....	11
3.3.7 CH347SPI_Write.....	12
3.3.8 CH347SPI_Read.....	12
3.3.9 CH347SPI_WriteRead.....	12
3.3.10 CH347StreamSPI4.....	13
3.4 JTAG 功能函数.....	13
3.4.1 操作流程.....	13
3.4.2 CH347Jtag_INIT.....	14
3.4.3 CH347Jtag_TmsChange.....	14
3.4.4 CH347Jtag_IoScan.....	15
3.4.5 CH347Jtag_IoScanT.....	15
3.4.6 CH347Jtag_WriteRead.....	16
3.4.7 CH347Jtag_WriteRead_Fast.....	16
3.4.8 CH347Jtag_SwitchTapState.....	17
3.4.9 CH347Jtag_ByteWriteDR.....	18
3.4.10 CH347Jtag_ByteReadDR.....	18
3.4.11 CH347Jtag_ByteWriteIR.....	18
3.4.12 CH347Jtag_ByteReadIR.....	19
3.4.13 CH347Jtag_BitWriteDR.....	19
3.4.14 CH347Jtag_BitWriteIR.....	19
3.4.15 CH347Jtag_BitReadIR.....	20

3.4.16	CH347Jtag_BitReadDR.....	20
3.5	I2C 功能函数.....	20
3.5.1	操作流程.....	20
3.5.2	相关数据类型.....	21
3.5.3	CH347I2C_Set.....	21
3.5.4	CH347I2C_SetStretch.....	22
3.5.5	CH347I2C_SetDelaymS.....	22
3.5.6	CH347StreamI2C.....	22
3.5.7	CH347StreamI2C_RetACK.....	23
3.5.8	CH347ReadEEPROM.....	23
3.5.9	CH347WriteEEPROM.....	24
四、	异步串行接口函数.....	25
4.1	公共函数.....	25
4.1.1	接口动态插拔检测.....	25
4.1.2	设备枚举操作.....	25
4.2	HID/VCP UART 功能函数.....	26
4.2.1	操作流程.....	26
4.2.2	CH347Uart_Open.....	27
4.2.3	CH347Uart_Close.....	27
4.2.4	CH347Uart_SetDeviceNotify.....	27
4.2.5	CH347Uart_Init.....	27
4.2.6	CH347Uart_SetTimeout.....	28
4.2.7	CH347Uart_Read.....	28
4.2.8	CH347Uart_Write.....	29
4.2.9	CH347Uart_QueryBufUpload.....	29
4.3	GPIO 功能函数.....	30
4.3.1	操作流程.....	30
4.3.2	CH347GPIO_Get.....	30
4.3.3	CH347GPIO_Set.....	31
4.3.4	CH347SetIntRoutine.....	31
4.3.5	CH347ReadInter.....	32
4.3.6	CH347AbortInter.....	32

## 一、简介

CH347是一款USB2.0高速转接芯片，以实现USB-UART (HID串口/VCP串口)、USB-SPI、USB-I2C、USB-JTAG以及USB-GPIO等接口，CH347F可同时支持如上接口，无需选择工作模式，CH347T支持4种工作模式，需要单独选择。

CH347DLL用于为CH347芯片提供操作系统端的UART/SPI/I2C/JTAG/BitStream等接口操作函数，支持厂商/HID/VCP驱动接口，使用时无需区分驱动接口和芯片工作模式。

## 二、接口说明

根据CH347所支持的USB转接接口特性，CH347DLL提供了USB-UART (HID串口/VCP串口)、USB-SPI、USB-I2C、USB-JTAG以及USB-GPIO的接口功能函数，包括基本功能函数与对应的功能函数，如EEPROM读写，JTAG应用中的SHIFT-DR状态读写等。

CH347F无需切换模式即可使用全部接口，所支持接口如下表所示：

功能接口说明	驱动接口	API
接口 0: USB2.0 转高速串口 0	CH343SER (VCP)	系统内原生串口 API 或 CH347DLL 内 CH347UART_xxx
接口 1: USB2.0 转高速串口 1		
接口 2: USB2.0 转 JTAG+SPI+I2C 等	CH347PAR	CH347DLL 内 CH347SPI_xxx、 CH347I2C_xxx、CH347JTAG_xxx

CH347T所支持接口如下表所示，通过上电时MODE配置引脚电平组合来切换不同模式。

工作模式	功能接口说明	驱动接口	API
模式 0	接口 0: USB2.0 转高速串口 0	CH343SER (VCP)	系统内原生串口 API 或 CH347DLL 内 CH347UART_xxx
	接口 1: USB2.0 转高速串口 1		
模式 1	接口 0: USB2.0 转高速串口 1	CH343SER (VCP)	系统内原生串口 API 或 CH347DLL 内 CH347UART_xxx
	接口 1: USB2.0 转 SPI+I2C	CH347PAR	CH347DLL 内 CH347SPI_xxx、 CH347I2C_xxx
模式 2	接口 0: USB2.0 HID 转高速串口 1	系统自带 HID 驱动	CH347UART_xxx
	接口 1: USB2.0 HID 转 SPI+I2C		CH347DLL 内 CH347SPI_xxx、 CH347I2C_xxx
模式 3	接口 0: USB2.0 转高速串口 1	CH343SER (VCP)	系统内原生串口 API 或 CH347DLL 内 CH347UART_xxx
	接口 1: USB2.0 转 JTAG+I2C	CH347PAR	CH347DLL 内 CH347JTAG_xxx、 CH347I2C_xxx

Table. CH347 接口功能 API 表

## 三、同步串行接口

### 3.1 相关数据类型

```
//驱动接口
#define CH347_USB_CH341      0
#define CH347_USB_HID        2
#define CH347_USB_VCP        3

//芯片功能接口号
#define CH347_FUNC_UART      0
#define CH347_FUNC_SPI_IIC    1
#define CH347_FUNC_JTAG_IIC   2
#define CH347_FUNC_JTAG_IIC_SPI 3    //CH347F
```

#### 3.1.1 SPI 控制器信息

```
//SPI 控制器配置
typedef struct _SPI_CONFIG{
    UCHAR          iMode;                // 0-3:SPI Mode0/1/2/3
    UCHAR          iClock;               // 0=60MHz, 1=30MHz, 2=15MHz, 3=7.5MHz, 4=3.75MHz,
                                         5=1.875MHz, 6=937.5KHz, 7=468.75KHz
    UCHAR          iByteOrder;           // 0=低位在前 (LSB), 1=高位在前 (MSB)
    USHORT         iSpiWriteReadInterval; // SPI 接口常规读取写入数据命令, 单位为 uS
    UCHAR          iSpiOutDefaultData;    // SPI 读数据时默认输出数据
    ULONG          iChipSelect;           // 片选控制, 位 7 为 0 则忽略片选控制,
                                         位 7 为 1 则参数有效: 位 1 位 0 为 00/01 分别选择
                                         CS1/CS2 引脚作为低电平有效片选
    UCHAR          CS1Polarity;           // 位 0: 片选 CS1 极性控制:
                                         0: 低电平有效; 1: 高电平有效;
    UCHAR          CS2Polarity;           // 位 0: 片选 CS2 极性控制:
                                         0: 低电平有效; 1: 高电平有效;
    USHORT         iIsAutoDeactiveCS;     // 操作完成后是否自动撤消片选
    USHORT         iActiveDelay;           // 设置片选后执行读写操作的延时时间, 单位 us
    ULONG          iDelayDeactive;        // 撤消片选后执行读写操作的延时时间, 单位 us
}mSpiCfgS, *mPSpiCfgS;
```

#### 3.1.2 设备信息

```
typedef struct _DEV_INFOR{
    UCHAR          iIndex;                // 当前打开序号
    UCHAR          DevicePath[MAX_PATH];  // 设备链接名, 用于 CreateFile
    UCHAR          UsbClass;              // 驱动类别, 0:CH347_USB_CH341,
                                         2:CH347_USB_HID, 3:CH347_USB_VCP
    UCHAR          FuncType;              // 功能类别, 0:CH347_FUNC_UART,
                                         1:CH347_FUNC_SPI_I2C, 2:CH347_FUNC_JTAG_I2C
```

```

CHAR        DeviceID[64];          // USB\VID_xxxx&PID_xxxx
UCHAR       ChipMode;              // 芯片工作模式, 0:Mode0(UART0/1),
                                   1:Mode1(Uart1+SPI+I2C),
                                   2:Mode2(HID Uart1+SPI+I2C),
                                   3:Mode3(Uart1+Jtag+IIC),
                                   4:CH347F(Uart*2+Jtag/SPI/IIC)

HANDLE      DevHandle;             // 设备句柄
USHORT      BulkOutEndpMaxSize;    // 批量上传端点大小
USHORT      BulkInEndpMaxSize;    // 批量下载端点大小
UCHAR       UsbSpeedType;          // USB 速度类型, 0:FS, 1:HS, 2:SS
UCHAR       CH347IfNum;           // USB 接口号:
                                   CH347T: IF0:UART; IF1:SPI/IIC/JTAG/GPIO
                                   CH347F: IF0:UART0; IF1:UART1;
                                   IF 2:SPI/IIC/JTAG/GPIO

UCHAR       DataUpEndp;            // 批量上传端点地址
UCHAR       DataDnEndp;           // 批量下载端点地址
CHAR        ProductString[64];    // USB 产品字符串
CHAR        ManufacturerString[64]; // USB 厂商字符串
ULONG       WriteTimeout;          // USB 写超时
ULONG       ReadTimeout;           // USB 读超时
CHAR        FuncDescStr[64];       // 接口功能描述符
UCHAR       FirmwareVer;           // 固件版本, 十六进制值
}mDeviceInforS, *mPDeviceInforS;

```

## 3.2 公共操作函数

### 3.2.1 CH347OpenDevice

#### 功能描述

该函数用于打开 CH347 设备, 支持 CH347 所有模式下的 SPI/I2C/JTAG 接口的打开

#### 函数定义

```

HANDLE WINAPI
CH347OpenDevice(ULONG DevI);

```

#### 参数说明

DevI: 指定操作设备序号

#### 返回值

执行成功返回设备序号

### 3.2.2 CH347CloseDevice

#### 功能描述

该函数用于关闭 CH347 设备, 支持 CH347 所有模式下 SPI/I2C/JTAG 接口的关闭

#### 函数定义

```

BOOL WINAPI

```

```
CH347CloseDevice (ULONG iIndex)
```

### 参数说明

iIndex: 指定操作设备序号

### 返回值

执行成功返回 1，失败返回 0

## 3.2.3 CH347SetDeviceNotify

### 功能描述

该函数用于指定设备事件通知程序，可用于 CH347 所有模式下 SPI/I2C/JTAG 接口的动态插拔检测

### 函数定义

```
BOOL WINAPI  
CH347SetDeviceNotify (ULONG iIndex,  
                      PCHAR iDeviceID,  
                      mPCH347_NOTIFY_ROUTINE iNotifyRoutine)
```

### 参数说明

iIndex: 指定操作设备序号  
iDeviceID: 可选参数, 指向字符串, 指定被监控的设备的 ID, 字符串以\0 终止  
iNotifyRoutine: 指定设备事件回调程序, 为 NULL 则取消事件通知,  
否则在检测到事件时调用该程序

### 返回值

执行成功返回 1，失败返回 0

### 注解

iDeviceID 该参数为可变参数，若需实现 CH347 设备的插拔检测，可定义宏如下

```
#define CH347DevID "VID_1A86&PID_55D\0"
```

传参时 iDeviceID 替换为 CH347DevID 即可实现对 CH347 同步串行接口的动态插拔检测

若需准确检测各模式下接口的插拔动作，可写下完整的 USBID，以模式 1 中 SPI 接口为例，可定义下方宏：

```
#define USBID_VEN_SPI_I2C "VID_1A86&PID_55DB&MI_02\0"
```

传参时 iDeviceID 替换为 USBID\_VEN\_SPI\_I2C 即可实现对 CH347 模式 1 的 SPI&I2C 接口的动态插拔检测

其他接口设置可参考 [3.2.7 接口动态插拔检测](#)

## 3.2.4 CH347GetDeviceInfor

### 功能描述

该函数用于获取设备当前接口模式、VID/PID 等信息

### 函数定义

```
BOOL WINAPI  
CH347GetDeviceInfor (ULONG iIndex,  
                    mDeviceInforS *DevInformation)
```

**参数说明**

iIndex: 指定操作设备序号  
DevInformation: 设备信息结构体

**返回值**

执行成功返回 1，失败返回 0

**注解**

设备信息结构体，可参考 [DEV\\_INFOR](#)

### 3.2.5 CH347GetVersion

**功能描述**

该函数用于获得驱动版本、库版本、设备版本、芯片类型(CH341 (FS)/CH347 (HS))

**函数定义**

```
BOOL WINAPI  
CH347GetVersion(ULONG iIndex,  
                PCHAR iDriverVer,  
                PCHAR iDLLVer,  
                PCHAR ibcdDevice,  
                PCHAR iChipType)
```

**参数说明**

iIndex: 指定操作设备序号  
iDriverVer: 驱动版本信息  
iDLLVer: 库版本信息  
ibcdDevice: 设备版本信息  
iChipType: 芯片类型

**返回值**

执行成功返回 1，失败返回 0

### 3.2.6 CH347SetTimeout

**功能描述**

该函数用于设置 USB 数据读写的超时

**函数定义**

```
BOOL WINAPI  
CH347SetTimeout(ULONG iIndex,  
                ULONG iWriteTimeout,  
                ULONG iReadTimeout)
```

**参数说明**

iIndex: 指定操作设备序号  
iWriteTimeout: 指定 USB 写出数据块的超时时间，以毫秒 ms 为单位，  
0xFFFFFFFF 指定不超时(默认值)  
iReadTimeout: 指定 USB 读取数据块的超时时间，以毫秒 ms 为单位，

0xFFFFFFFF 指定不超时(默认值)

## 返回值

执行成功返回 1，失败返回 0

### 3.2.7 接口动态插拔检测

检测同步串行接口动态插拔信息可通过 [CH347SetDeviceNotify](#) 函数来实现，代码参考如下：

启用 CH347 同步串行接口 USB 的插入和移除的监测：

```
CH347SetDeviceNotify(DevIndex, USBDevID, UsbDevPnpNotify);
```

关闭 CH347 同步串行接口 USB 的插入和移除的监测，在程序退出时一定要关闭。

```
CH347SetDeviceNotify(DevIndex, USBDevID, NULL);
```

// CH347 设备插拔检测通知程序

```
VOID CALLBACK UsbDevPnpNotify (ULONG iEventStatus )
```

```
{
    if(iEventStatus==CH347_DEVICE_ARRIVAL)          // 设备插入事件, 已经插入
        PostMessage(DebugHwnd, WM_CH347DevArrive, 0, 0);
    else if(iEventStatus==CH347_DEVICE_REMOVE)       // 设备拔出事件, 已经拔出
        PostMessage(DebugHwnd, WM_CH347DevRemove, 0, 0);
    return;
}
```

若需做到准确检测各模式下的 SPI/I2C/JTAG 接口插拔信息，可写下如下完整 USBID，在使用 CH347SetDeviceNotify 时将 iDeviceID 替换成相应的 USBID 宏即可。

```
//MODE1 SPI/I2C
```

```
#define USBID_VEN_Mode1_SPI_I2C "VID_1A86&PID_55DB&MI_02\0"
```

```
//MODE2 SPI/I2C
```

```
#define USBID_HID_Mode2_SPI_I2C "VID_1A86&PID_55DC&MI_01\0"
```

```
//MODE3 JTAG/I2C
```

```
#define USBID_VEN_Mode3_JTAG_I2C "VID_1A86&PID_55DA&MI_02\0"
```

### 3.2.8 设备枚举操作

在本接口库中，API 通过指定设备序号实现对应操作，设备序号是设备逐个插入的过程中，根据其插入顺序进行编号产生。实现设备枚举功能可以通过设备 Open 函数打开对应设备序号，根据函数返回值判断设备是否有效且存在。

其中 SPI/I2C/JTAG 接口的打开/关闭函数可用：[CH347OpenDevice](#)/[CH347CloseDevice](#)。

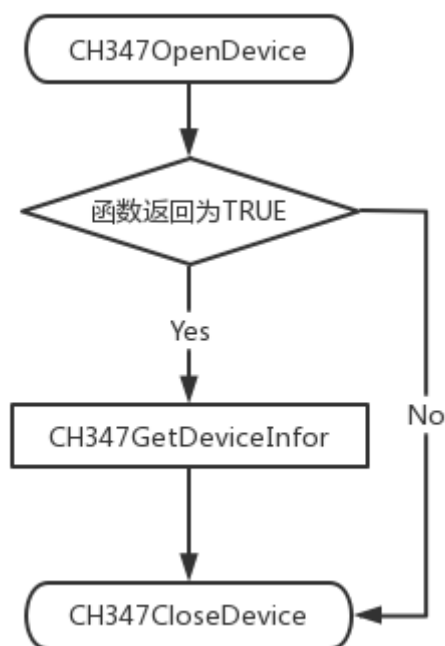


Figure 3.2.8 设备枚举操作流程

### 3.3 SPI 功能函数

#### 3.3.1 操作流程

打开设备后，设置设备 USB 读写超时参数，配置 SPI 控制器参数后进行 SPI 初始化设置，设置成功后即可通过调用 SPI 读写函数与设备进行通讯。

函数调用流程图如下：

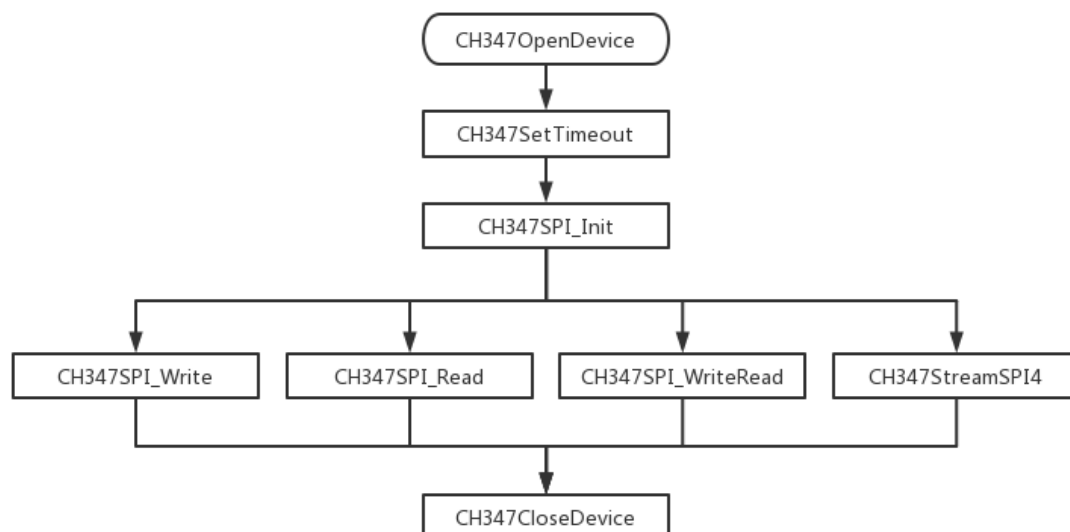


Figure 3.3.1 SPI 函数操作流程

函数具体说明请参考以下内容。

### 3.3.2 CH347SPI\_Init

#### 功能描述

该函数用于对 SPI 控制器进行参数配置

#### 函数定义

```
BOOL WINAPI  
CH347SPI_Init( ULONG iIndex,  
               mSpiCfgS *SpiCfg)
```

#### 参数说明

iIndex: 指定操作设备序号  
SpiCfg: SPI 控制器配置

#### 返回值

执行成功返回 1，失败返回 0

#### 注解

SPI 控制器配置可参考结构体 [SPI\\_CONFIG](#)

### 3.3.3 CH347SPI\_SetDataBits

#### 功能描述

该函数用于 CH347F 设置 SPI 支持数据位数

#### 函数定义

```
BOOL WINAPI  
CH347SPI_SetDataBits( ULONG iIndex,  
                      UCHAR iDataBits)
```

#### 参数说明

iIndex: 指定操作设备序号  
iDataBits: SPI 数据位数，0 表示 8bit，1 表示 16bit

#### 返回值

执行成功返回 1，失败返回 0

### 3.3.4 CH347SPI\_GetCfg

#### 功能描述

该函数用于获取 SPI 控制器当前配置

#### 函数定义

```
BOOL WINAPI  
CH347SPI_GetCfg( ULONG iIndex,  
                 SpiCfgS *SpiCfg)
```

#### 参数说明

iIndex: 指定操作设备序号  
SpiCfg: SPI 控制器配置

#### 返回值

执行成功返回 1，失败返回 0

## 注解

SPI 控制器配置可参考结构体 [SPI\\_CONFIG](#)

### 3.3.5 CH347SPI\_ChangeCS

#### 功能描述

该函数用于设置片选状态,使用前需先调用 [CH347SPI\\_Init](#) 对 CS 进行设置

#### 函数定义

```
BOOL WINAPI
CH347SPI_ChangeCS(ULONG    iIndex,
                  UCHAR     iStatus)
```

#### 参数说明

iIndex: 指定操作设备序号  
iStatus: 0=撤销片选, 1=设置片选

#### 返回值

执行成功返回 1，失败返回 0

### 3.3.6 CH347SPI\_SetChipSelect

#### 功能描述

该函数用于设置 SPI 片选

#### 函数定义

```
BOOL WINAPI
CH347SPI_SetChipSelect(ULONG    iIndex,
                      USHORT     iEnableSelect,
                      USHORT     iChipSelect,
                      ULONG      iIsAutoDeactiveCS,
                      ULONG      iActiveDelay,
                      ULONG      iDelayDeactive);
```

#### 参数说明

iIndex: 指定操作设备序号  
iEnableSelect: 低八位为 CS1，高八位为 CS2；  
字节值为 0=设置 CS, 为 1=忽略此 CS 设置  
iChipSelect: 低八位为 CS1，高八位为 CS2;片选输出，  
0=撤消片选, 1=设置片选  
iIsAutoDeactiveCS: 低 16 位为 CS1，高 16 位为 CS2;操作完成后是否自动撤消片选  
iActiveDelay: 低 16 位为 CS1，高 16 位为 CS2；  
设置片选后执行读写操作的延时时间, 单位 uS  
iDelayDeactive: 低 16 位为 CS1，高 16 位为 CS2；  
撤消片选后执行读写操作的延时时间, 单位 uS

#### 返回值

执行成功返回 1，失败返回 0

### 3.3.7 CH347SPI\_Write

#### 功能描述

该函数用于 SPI 写数据

#### 函数定义

BOOL WINAPI

```
CH347SPI_Write( ULONG    iIndex,  
                ULONG    iChipSelect,  
                ULONG    iLength,  
                ULONG    iWriteStep,  
                PVOID    ioBuffer);
```

#### 参数说明

iIndex: 指定操作设备序号  
iChipSelect: 片选控制, 位 7 为 0 则忽略片选控制, 位 7 为 1 进行片选操作  
iLength: 准备传输的数据字节数  
iWriteStep: 准备读取的单个块的长度  
ioBuffer: 指向一个缓冲区, 放置准备从 MOSI 写出的数据

#### 返回值

执行成功返回 1, 失败返回 0

### 3.3.8 CH347SPI\_Read

#### 功能描述

该函数用于读取 SPI 数据

#### 函数定义

BOOL WINAPI

```
CH347SPI_Read( ULONG    iIndex,  
               ULONG    iChipSelect,  
               ULONG    oLength,  
               PULONG    iLength,  
               PVOID    ioBuffer);
```

#### 参数说明

iIndex: 指定操作设备序号  
iChipSelect: 片选控制, 位 7 为 0 则忽略片选控制, 位 7 为 1 进行片选操作  
oLength: 准备发出的数据字节数  
iLength: 准备读取的数据字长度  
ioBuffer: 指向一个缓冲区, 放置准备从 MOSI 写出的数据,  
返回后是从 MISO 读入的数据

#### 返回值

执行成功返回 1, 失败返回 0

### 3.3.9 CH347SPI\_WriteRead

#### 功能描述

该函数用于写入和读取 SPI 数据流

## 函数定义

```

BOOL WINAPI
CH347SPI_WriteRead( ULONG      iIndex,
                    ULONG      iChipSelect,
                    ULONG      iLength,
                    PVOID      ioBuffer );

```

## 参数说明

iIndex: 指定操作设备序号  
 iChipSelect: 片选控制, 位 7 为 0 则忽略片选控制, 位 7 为 1 进行片选操作  
 iLength: 准备传输的数据字节数  
 ioBuffer: 指向一个缓冲区, 放置准备从 MOSI 写出的数据, 返回后是从 MISO 读入的数据

## 返回值

执行成功返回 1, 失败返回 0

### 3.3.10 CH347StreamSPI4

## 功能描述

该函数用于处理 SPI 数据流, 写入的同时读出数据

## 函数定义

```

BOOL WINAPI
CH347StreamSPI4(ULONG      iIndex,
                ULONG      iChipSelect,
                ULONG      iLength,
                PVOID      ioBuffer );

```

## 参数说明

iIndex: 指定操作设备序号  
 iChipSelect: 片选控制, 位 7 为 0 则忽略片选控制, 位 7 为 1 进行片选操作  
 iLength: 准备传输的字节数  
 ioBuffer: 指向一个缓冲区, 放置准备从 MOSI 写出的数据, 返回后是从 MISO 读入的数据

## 返回值

执行成功返回 1, 失败返回 0

## 3.4 JTAG 功能函数

### 3.4.1 操作流程

打开设备后, 使用 [CH347Jtag\\_INIT](#) 对设备进行初始化操作;

使用 [CH347Jtag\\_SwitchTapState\(0\)](#) 复位目标设备 JTAG TAP 状态为 Test-Logic-Reset 状态, 随后根据操作需求可使用 [CH347Jtag\\_TmsChange](#) 切换到 SHIFT-DR/SHIFT-IR 状态进行读写操作, 其中读写函数为位带方式读写与批量快速读写方式两种, 可根据实际用途进行选择。

函数调用流程图如下:

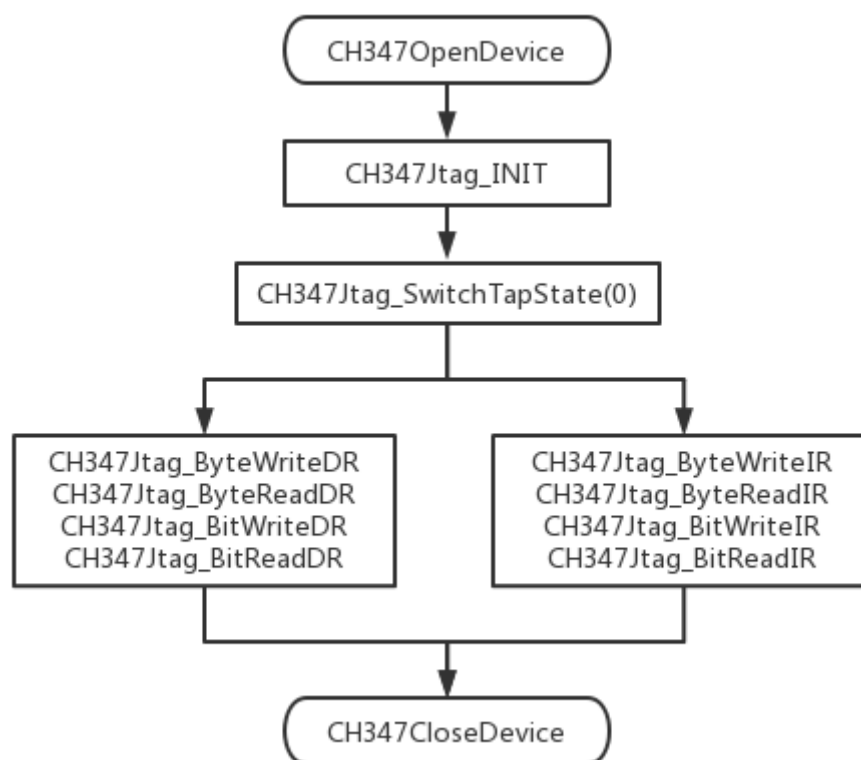


Figure 3.4.1 JTAG 函数操作流程图

函数具体说明请参考以下内容。

### 3.4.2 CH347Jtag\_INIT

#### 功能描述

该函数用于初始化 JTAG 接口与设置通信速度

#### 函数定义

```

BOOL WINAPI
CH347Jtag_INIT( ULONG    iIndex,
                UCHAR    iClockRate);
  
```

#### 参数说明

iIndex: 指定操作设备序号  
iClockRate: 通信速度；有效值为 0-5，值越大通信速度越快

#### 返回值

执行成功返回 1，失败返回 0

### 3.4.3 CH347Jtag\_TmsChange

#### 功能描述

该函数用于传入 TMS 的值来进行对应状态切换，TMS 值参考 JTAG TAP 状态机

#### 函数定义

```

BOOL WINAPI
CH347Jtag_TmsChange( ULONG    iIndex,
  
```

```

        PCHAR    tmsValue,
        ULONG    Step,
        ULONG    Skip);

```

**参数说明**

iIndex: 指定操作设备序号  
 tmsValue: 进行切换的 TMS 位值, 以字节为单位  
 Step: tmsValue 内存储的 TMS 有效位数  
 Skip: 有效起始位

**返回值**

执行成功返回 1, 失败返回 0

**3.4.4 CH347Jtag\_IoScan****功能描述**

该函数主要用于 SHIFT-DR/IR 状态下执行数据读写, 最后读写结束切换至 EXIT-DR/IR, 状态切换可配合 CH347Jtag\_TmsChange 使用

**函数定义**

```

BOOL WINAPI
CH347Jtag_IoScan(ULONG    iIndex,
                  PCHAR    DataBits,
                  ULONG    DataBitsNb,
                  BOOL    IsRead);

```

**参数说明**

iIndex: 指定操作设备序号  
 DataBits: 需要进行传输的数据  
 DataBitsNb: 需要传输数据的位数  
 IsRead: 是否需要读取数据

**返回值**

执行成功返回 1, 失败返回 0

**3.4.5 CH347Jtag\_IoScanT****功能描述**

该函数可在 SHIFT-DR/IR 状态下进行多次调用实现数据读写, 通过 IsLastPkt 判断是否读写结束切换至 EXIT-DR/IR, 状态切换可配合 CH347Jtag\_TmsChange 使用

**函数定义**

```

BOOL WINAPI
CH347Jtag_IoScanT(ULONG    iIndex,
                  PCHAR    DataBits,
                  ULONG    DataBitsNb,
                  BOOL    IsRead,
                  BOOL    IsLastPkt);

```

**参数说明**

iIndex: 指定操作设备序号

**DataBits:** 需要进行传输的数据  
**DataBitsNb:** 需要传输数据的位数  
**IsRead:** 是否需要读取数据  
**IsLastPkt:** 是否为最后一包数据, 若为 TRUE, 则将最后 1bit 数据切换至 EXIT-DR/IR 发送

### 返回值

执行成功返回 1, 失败返回 0

## 3.4.6 CH347Jtag\_WriteRead

### 功能描述

该函数以位带方式进行 SHIFT-DR/IR 状态数据读写。适用于少量数据读写。如指令操作、状态机切换等控制类传输。如批量数据传输, 建议使用 [CH347Jtag\\_WriteRead\\_Fast](#) 命令包以字节为单位进行批量读写。

### 函数定义

```

BOOL WINAPI
CH347Jtag_WriteRead(ULONG iIndex,
                    BOOL IsDR,
                    ULONG iWriteBitLength,
                    PVOID iWriteBitBuffer,
                    PULONG oReadBitLength,
                    PVOID oReadBitBuffer )
  
```

### 参数说明

**iIndex:** 指定操作设备序号  
**IsDR:** 判断切换状态进行读写,  
 TRUE= SHIFT-DR 数据读写, FALSE=SHIFT-IR 数据读写  
**iWriteBitLength:** 准备写出的数据长度  
**iWriteBitBuffer:** 指向一个缓冲区, 放置准备写出的数据  
**oReadBitLength:** 指向长度单元, 返回后为实际读取的长度  
**oReadBitBuffer:** 指向一个足够大的缓冲区, 用于保存读取的数据

### 返回值

执行成功返回 1, 失败返回 0

### 注解

该函数通过 IsDR 的值来判断操作 JTAG 状态切换到 SHIFT-DR 还是 SHIFT-IR 状态, 然后以位带的方式进行数据读写之后再切换回 RUN-TEST 状态, 其状态切换路径如下:

Run-Test->Shift-IR/DR->Exit IR/DR -> Run-Test

## 3.4.7 CH347Jtag\_WriteRead\_Fast

### 功能描述

该函数用于切换至 SHIFT-IR/DR 状态进行数据批量读写, 用于多字节连续读写。如 JTAG 固件下载操作。

### 函数定义

```

BOOL WINAPI
CH347Jtag_WriteRead_Fast (ULONG    iIndex,
                           BOOL     IsDR,
                           ULONG    iWriteBitLength,
                           PVOID     iWriteBitBuffer,
                           PULONG    oReadBitLength,
                           PVOID     oReadBitBuffer );

```

### 参数说明

iIndex: 指定操作设备序号  
 IsDR: 判断切换状态进行读写,  
 TRUE = SHIFT-DR 数据读写, FALSE = SHIFT-IR 数据读写  
 iWriteBitLength: 准备写出的数据长度  
 iWriteBitBuffer: 指向一个缓冲区, 放置准备写出的数据  
 oReadBitLength: 指向长度单元, 返回后为实际读取的长度  
 oReadBitBuffer: 指向一个足够大的缓冲区, 用于保存读取的数据

### 返回值

执行成功返回 1, 失败返回 0

### 注解

该函数功能与 [CH347Jtag\\_WriteRead](#) 相似, 但该函数使用批量读写方式, 以字节格式进行数据读写。

## 3.4.8 CH347Jtag\_SwitchTapState

### 功能描述

该函数用于切换 JTAG 状态机状态

### 函数定义

```

BOOL CH347Jtag_SwitchTapState (UCHAR TapState)

```

### 参数说明

TapState: 通过输入序号进行状态切换

### 返回值

执行成功返回 1, 失败返回 0

### 注解

TapState 状态切换说明如下:

- 0: 复位目标设备状态为 Test-Logic Reset
- 1: 跟随上一状态进入 Run-Test/Idle
- 2: Run-Test/Idle -> Shift-DR
- 3: Shift-DR -> Run-Test/Idle
- 4: Run-Test/Idle -> Shift-IR
- 5: Shift-IR -> Run-Test/Idle
- 6: Exit1-DR -> Run-Test-Idle

### 3. 4. 9 CH347Jtag\_ByteWriteDR

#### 功能描述

该函数用于将 JTAG 状态机切换到 SHIFT-DR 状态, 以字节为单位, 可进行多字节连续读写。

#### 函数定义

```
BOOL WINAPI  
CH347Jtag_ByteWriteDR (ULONG    iIndex,  
                        ULONG     iWriteLength,  
                        PVOID     iWriteBuffer);
```

#### 参数说明

iIndex: 指定操作设备序号  
iWriteLength: 准备写出数据的字节长度  
iWriteBuffer: 指向一个缓冲区, 放置准备写出的数据

#### 返回值

执行成功返回 1, 失败返回 0

### 3. 4. 10 CH347Jtag\_ByteReadDR

#### 功能描述

该函数用于将 JTAG 状态机切换到 SHIFT-DR 状态, 以字节为单位, 可进行多字节连续读写。

#### 函数定义

```
BOOL WINAPI  
CH347Jtag_ByteReadDR ( ULONG    iIndex,  
                      PULONG     oReadLength,  
                      PVOID     oReadBuffer);
```

#### 参数说明

iIndex: 指定操作设备序号  
oReadLength: 准备读取数据的字节长度  
oReadBuffer: 指向一个缓冲区, 放置准备读取的数据

#### 返回值

执行成功返回 1, 失败返回 0

### 3. 4. 11 CH347Jtag\_ByteWriteIR

#### 功能描述

该函数用于将 JTAG 状态机切换到 SHIFT-IR 状态, 以字节为单位, 可进行多字节连续读写。

#### 函数定义

```
BOOL WINAPI  
CH347Jtag_ByteWriteIR (ULONG    iIndex,  
                       ULONG     iWriteLength,  
                       PVOID     iWriteBuffer);
```

#### 参数说明

iIndex: 指定操作设备序号  
iWriteLength: 准备写出数据的字节长度

iWriteBuffer: 指向一个缓冲区, 放置准备写出的数据

### 返回值

执行成功返回 1, 失败返回 0

## 3. 4. 12 CH347Jtag\_ByteReadIR

### 功能描述

该函数用于将 JTAG 状态机切换到 SHIFT-IR 状态, 以字节为单位, 可进行多字节连续读写。

### 函数定义

```
BOOL WINAPI  
CH347Jtag_ByteReadIR(ULONG iIndex,  
                      PULONG oReadLength,  
                      PVOID oReadBuffer);
```

### 参数说明

iIndex: 指定操作设备序号  
oReadLength: 准备读取数据的字节长度  
oReadBuffer: 指向一个缓冲区, 放置准备读取的数据

### 返回值

执行成功返回 1, 失败返回 0

## 3. 4. 13 CH347Jtag\_BitWriteDR

### 功能描述

该函数用于将 JTAG 状态机切换到 SHIFT-DR 状态, 以位带方式进行数据读写。

### 函数定义

```
BOOL WINAPI  
CH347Jtag_BitWriteDR(ULONG iIndex,  
                     ULONG iWriteLength,  
                     PVOID iWriteBuffer);
```

### 参数说明

iIndex: 指定操作设备序号  
iWriteLength: 准备写出数据的字节长度  
iWriteBuffer: 指向一个缓冲区, 放置准备写出的数据

### 返回值

执行成功返回 1, 失败返回 0

## 3. 4. 14 CH347Jtag\_BitWriteIR

### 功能描述

该函数用于将 JTAG 状态机切换到 SHIFT-IR 状态, 以位带方式进行数据读写。

### 函数定义

```
BOOL WINAPI  
CH347Jtag_BitWriteIR(ULONG iIndex,  
                     ULONG iWriteLength,
```

```
PVOID iWriteBuffer);
```

**参数说明**

iIndex: 指定操作设备序号  
 iWriteLength: 准备写出数据的字节长度  
 iWriteBuffer: 指向一个缓冲区, 放置准备写出的数据

**返回值**

执行成功返回 1, 失败返回 0

**3.4.15 CH347Jtag\_BitReadIR****功能描述**

该函数用于将 JTAG 状态机切换到 SHIFT-IR 状态, 以位带方式进行数据读写。

**函数定义**

```
BOOL WINAPI
CH347Jtag_BitReadIR(ULONG iIndex,
                    PULONG oReadLength,
                    PVOID oReadBuffer);
```

**参数说明**

iIndex: 指定操作设备序号  
 oReadLength: 准备读取数据的字节长度  
 oReadBuffer: 指向一个缓冲区, 放置准备读取的数据

**返回值**

执行成功返回 1, 失败返回 0

**3.4.16 CH347Jtag\_BitReadDR****功能描述**

该函数用于将 JTAG 状态机切换到 SHIFT-DR 状态, 以字节为单位, 可进行多字节连续读写。

**函数定义**

```
BOOL WINAPI
CH347Jtag_BitReadDR(ULONG iIndex,
                    PULONG oReadLength,
                    PVOID oReadBuffer);
```

**参数说明**

iIndex: 指定操作设备序号  
 oReadLength: 准备读取数据的字节长度  
 oReadBuffer: 指向一个缓冲区, 放置准备读取的数据

**返回值**

执行成功返回 1, 失败返回 0

**3.5 I2C 功能函数****3.5.1 操作流程**

打开指定操作设备获取设备序号, 设置设备 I2C 接口速度/SCL 频率, 进行 I2C 读写操作,

函数调用流程图如下：

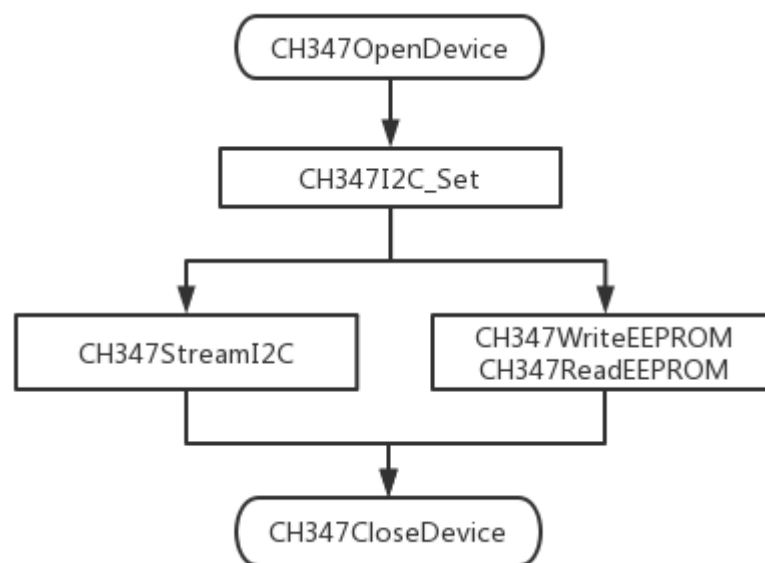


Figure 3.5.1 I2C 操作流程圖

函数具体说明请参考以下内容。

### 3.5.2 相关数据类型

#### EEPROM 类型

```

typedef enum    _EEPROM_TYPE {
    ID_24C01,
    ID_24C02,
    ID_24C04,
    ID_24C08,
    ID_24C16,
    ID_24C32,
    ID_24C64,
    ID_24C128,
    ID_24C256,
    ID_24C512,
    ID_24C1024,
    ID_24C2048,
    ID_24C4096
} EEPROM_TYPE;
  
```

### 3.5.3 CH347I2C\_Set

#### 功能描述

该函数用于指定操作设备并设置 I2C 接口速度/SCL 频率

#### 函数定义

```

BOOL WINAPI
CH347I2C_Set( ULONG    iIndex,
              ULONG    iMode )
  
```

**参数说明**

iIndex: 指定操作设备序号  
iMode: 设置模式  
位 2-0: 000=低速/20KHz, 001=标准/100KHz (默认值),  
010=快速/400KHz, 011=高速/750KHz,  
100=50KHz, 101=200KHz, 110=1MHz  
位 7-3: 保留为 0

**返回值**

执行成功返回 1, 失败返回 0

**3.5.4 CH347I2C\_SetStretch****功能描述**

该函数用于设置时钟延展功能

**函数定义**

```
BOOL WINAPI  
CH347I2C_SetStretch( ULONG iIndex,  
                     BOOL iEnable );
```

**参数说明**

iIndex: 指定操作设备序号  
iEnable: 是否使能时钟延展功能

**返回值**

执行成功返回 1, 失败返回 0

**3.5.5 CH347I2C\_SetDelaymS****功能描述**

该函数用于设置硬件异步延时, 调用后很快返回, 而在下一个流操作之前延时指定毫秒数

**函数定义**

```
BOOL WINAPI  
CH347I2C_SetDelaymS( ULONG iIndex,  
                     ULONG iDelay );
```

**参数说明**

iIndex: 指定操作设备序号  
iDelay: 指定延时的毫秒数

**返回值**

执行成功返回 1, 失败返回 0

**3.5.6 CH347StreamI2C****功能描述**

该函数用于处理 I2C 数据流, 实现 I2C 数据的读取和写入

**函数定义**

```
BOOL WINAPI
```

```

CH347StreamI2C( ULONG    iIndex,
                ULONG    iWriteLength,
                PVOID    iWriteBuffer,
                ULONG    iReadLength,
                PVOID    oReadBuffer )

```

**参数说明**

**iIndex:** 指定操作设备序号  
**iWriteLength:** 准备写出的数据字节数  
**iWriteBuffer:** 指向一个缓冲区，放置准备写出的数据，首字节通常是 I2C 设备地址及读写方向位，若地址长度超过 7 为则此字节仍可写入以此类推  
**iReadLength:** 准备读取的数据字节数  
**oReadBuffer:** 指向一个缓冲区，函数返回后为读入的数据

**返回值**

执行成功返回 1，失败返回 0

**3.5.7 CH347StreamI2C\_RetACK****功能描述**

该函数用于处理 I2C 数据流，实现 I2C 数据的读取和写入，并返回读写操作产生的 ACK 数量

**函数定义**

```

BOOL WINAPI
CH347StreamI2C_RetACK( ULONG    iIndex,
                      ULONG    iWriteLength,
                      PVOID    iWriteBuffer,
                      ULONG    iReadLength,
                      PVOID    oReadBuffer,
                      PULONG    rAckCount)

```

**参数说明**

**iIndex:** 指定操作设备序号  
**iWriteLength:** 准备写出的数据字节数  
**iWriteBuffer:** 指向一个缓冲区，放置准备写出的数据，首字节通常是 I2C 设备地址及读写方向位，若地址长度超过 7 为则此字节仍可写入以此类推  
**iReadLength:** 准备读取的数据字节数  
**oReadBuffer:** 指向一个缓冲区，函数返回后为读入的数据  
**rAckCount:** 指向读写返回的 ACK 数量

**返回值**

执行成功返回 1，失败返回 0

**3.5.8 CH347ReadEEPROM****功能描述**

该函数用于向 EEPROM 中读取数据块

**函数定义**

```
BOOL WINAPI
CH347ReadEEPROM(  ULONG    iIndex,
                   EEPROM_TYPE iEepromID,
                   ULONG    iAddr,
                   ULONG    iLength,
                   PCHAR    iBuffer )
```

### 参数说明

iIndex: 指定操作设备序号  
iEepromID: 指定 EEPROM 型号  
iAddr: 指定数据单元的地址  
iLength: 准备读取的数据字节数  
iBuffer: 指向一个缓冲区, 放置准备读取的数据

### 返回值

执行成功返回 1, 失败返回 0

### 注解

iEepromID 所指定的型号可参考 [EEPROM\\_TYPE](#)

## 3.5.9 CH347WriteEEPROM

### 功能描述

该函数用于向 EEPROM 中写入数据块

### 函数定义

```
BOOL WINAPI
CH347WriteEEPROM(  ULONG    iIndex,
                   EEPROM_TYPE iEepromID,
                   ULONG    iAddr,
                   ULONG    iLength,
                   PCHAR    iBuffer )
```

### 参数说明

iIndex: 指定操作设备序号  
iEepromID: 指定 EEPROM 型号  
iAddr: 指定数据单元的地址  
iLength: 准备写出的数据字节数  
iBuffer: 指向一个缓冲区, 放置准备写出的数据

### 返回值

执行成功返回 1, 失败返回 0

### 注解

iEepromID 所指定的型号可参考 [EEPROM\\_TYPE](#)

## 四、异步串行接口函数

### 4.1 公共函数

#### 4.1.1 接口动态插拔检测

检测 CH347 UART 接口动态插拔信息可通过 [CH347Uart\\_SetDeviceNotify](#) 函数来实现，代码可参考 [3.2.6 接口动态插拔检测](#)。

启用 CH347 UART 串口 USB 的插入和移除的监测：

```
CH347Uart_SetDeviceNotify(DevIndex, USBUartDevID, UsbDevPnpNotify);
```

关闭 CH347 UART 串口 USB 的插入和移除的监测，在程序退出时一定要关闭。

```
CH347Uart_SetDeviceNotify(DevIndex, USBUartDevID, NULL);
```

监视的 USBUartDevID 可为如下字符串或自行定义 ID 内容。

```
//MODE0 UART0
#define USBID_VCP_Mode0_UART0 "VID_1A86&PID_55DA&MI_00\0"
//MODE0 UART1
#define USBID_VCP_Mode0_UART1 "VID_1A86&PID_55DA&MI_01\0"
//MODE1 UART
#define USBID_VEN_Mode1_UART1 "VID_1A86&PID_55DB&MI_00\0"
//MODE2 UART
#define USBID_HID_Mode2_UART1 "VID_1A86&PID_55DB&MI_00\0"
//MODE3 UART
#define USBID_VEN_Mode3_UART1 "VID_1A86&PID_55DB&MI_00\0"
```

#### 4.1.2 设备枚举操作

在本接口库中，API 通过指定设备序号实现对应操作，设备序号是设备逐个插入的过程中，根据其插入顺序进行编号产生。实现设备枚举功能可以通过设备 Open 函数打开对应设备序号，根据函数返回值判断设备是否有效或存在。

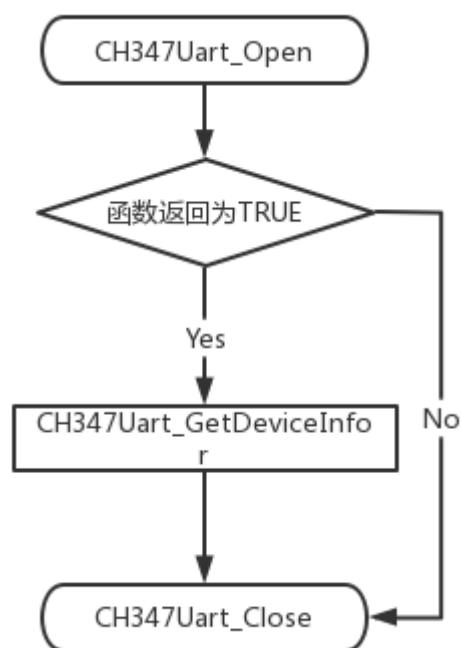


Figure 4.1.2 设备枚举操作流程

## 4.2 HID/VCP UART 功能函数

### 4.2.1 操作流程

打开设备后，使用 [CH347Uart\\_Open](#) 函数打开串口，设置对应串口参数后使用 [CH347Uart\\_Init](#) 函数进行串口设置，然后即可使用 [CH347Uart\\_Write](#) 或 [CH347Uart\\_Read](#) 函数实现串口数据收发。

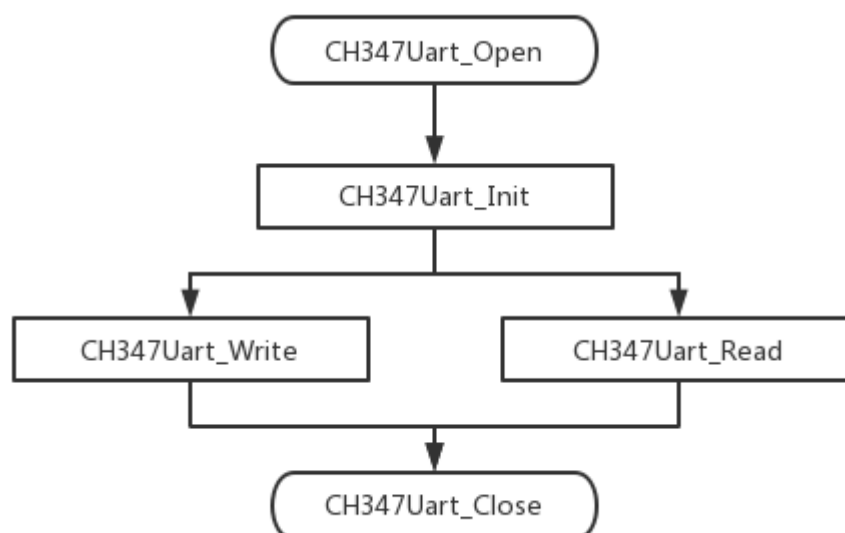


Figure 4.2.1 HID 串口操作流程

函数具体说明请参考以下内容。

#### 4. 2. 2 CH347Uart\_Open

##### 功能描述

该函数用于打开 CH347 串口

##### 函数定义

```
HANDLE WINAPI  
CH347Uart_Open(ULONG iIndex)
```

##### 参数说明

iIndex: 指定操作设备序号

##### 返回值

执行成功返回 1，失败返回 0

#### 4. 2. 3 CH347Uart\_Close

##### 功能描述

该函数用于关闭 CH347 串口

##### 函数定义

```
BOOL WINAPI  
CH347Uart_Close(ULONG iIndex)
```

##### 参数说明

iIndex: 指定操作设备序号

##### 返回值

执行成功返回 1，失败返回 0

#### 4. 2. 4 CH347Uart\_SetDeviceNotify

##### 功能描述

该函数用于设定设备时间通知程序，可用于 CH347 UART 的动态插拔检测

##### 函数定义

```
BOOL WINAPI  
CH347Uart_SetDeviceNotify( ULONG iIndex,  
                           PCHAR iDeviceID,  
                           mPCH347_NOTIFY_ROUTINE iNotifyRoutine)
```

##### 参数说明

iIndex: 指定操作设备序号  
iDeviceID: 可选参数, 指向字符串, 指定被监控的设备的 ID, 字符串以\0 终止  
iNotifyRoutine: 指定设备事件回调程序, 为 NULL 则取消事件通知, 否则在检测到事件时调用该程序

##### 返回值

执行成功返回 1，失败返回 0

#### 4. 2. 5 CH347Uart\_Init

##### 功能描述

该函数用于初始化串口参数

### 函数定义

```

BOOL WINAPI
CH347Uart_Init (ULONG      iIndex,
                DWORD      BaudRate,
                UCHAR      ByteSize,
                UCHAR      Parity,
                UCHAR      StopBits,
                UCHAR      ByteTimeout)

```

### 参数说明

iIndex: 指定操作设备序号  
 BaudRate, : 设置的波特率数值  
 ByteSize: 数据位 (5、6、7、8、16)  
 Parity: 校验位 (0: None; 1: Odd; 2: Even; 3: Mark; 4: Space)  
 StopBits: 停止位数 (0: 停止位; 1: 1.5 停止位; 2: 2 停止位)  
 ByteTimeout: 字节超时时间, 单位 100uS

### 返回值

执行成功返回 1, 失败返回 0

## 4.2.6 CH347Uart\_SetTimeout

### 功能描述

该函数用于设置 USB 数据读写的超时时间

### 函数定义

```

BOOL WINAPI
CH347Uart_SetTimeout (ULONG      iIndex,
                     ULONG      iWriteTimeout,
                     ULONG      iReadTimeout )

```

### 参数说明

iIndex: 指定操作设备序号  
 iWriteTimeout: 指定 USB 写出数据块的超时时间。以毫秒 mS 为单位。  
 0xFFFFFFFF 指定不超时 (默认值)  
 iReadTimeout: 指定 USB 读取数据块的超时时间。以毫秒 mS 为单位。  
 0xFFFFFFFF 指定不超时 (默认值)

### 返回值

执行成功返回 1, 失败返回 0

## 4.2.7 CH347Uart\_Read

### 功能描述

该函数用于读取串口数据

### 函数定义

```

BOOL WINAPI

```

```
CH347Uart_Read( ULONG    iIndex,  
                PVOID     oBuffer,  
                PULONG    ioLength )
```

#### 参数说明

iIndex: 指定操作设备序号  
oBuffer: 指向一个足够大的缓冲区, 用于保存读取的数据  
ioLength: 指向长度单元, 输入时为准备读取的长度, 返回后为实际读取的长度

#### 返回值

执行成功返回 1, 失败返回 0

### 4.2.8 CH347Uart\_Write

#### 功能描述

该函数用于发送串口数据

#### 函数定义

```
BOOL WINAPI  
CH347Uart_Write(ULONG    iIndex,  
                PVOID     iBuffer,  
                PULONG    ioLength )
```

#### 参数说明

iIndex: 指定操作设备序号  
iBuffer: 指向一个缓冲区, 放置准备写出的数据  
ioLength: 指向长度单元, 输入时为准备写出的长度, 返回后为实际写出的长度

#### 返回值

执行成功返回 1, 失败返回 0

### 4.2.9 CH347Uart\_QueryBufUpload

#### 功能描述

该函数用于查询缓冲区还有多少字节未取出

#### 函数定义

```
BOOL WINAPI  
CH347Uart_QueryBufUpload(ULONG    iIndex,  
                          LONGLONG *RemainBytes);
```

#### 参数说明

iIndex: 指定操作设备序号  
RemainBytes: 返回当前缓冲区中未取出字节数量

#### 返回值

执行成功返回 1, 失败返回 0

## 4.3 GPIO 功能函数

### 4.3.1 操作流程

操作 GPIO 时可用 [CH347OpenDevice/CH347Uart\\_Open](#) 打开设备。

使用 [CH347GPIO\\_Get](#) 获取当前 GPIO 状态之后,根据操作需求使用 [CH347GPIO\\_Set](#) 设置 GPIO 的输入输出状态。

实现 GPIO 控制和获取可调用 [CH347GPIO\\_Set](#) 和 [CH347GPIO\\_Get](#) 实现。

实现 GPIO 中断功能可使用 [CH347SetIntRoutine](#) 和 [CH347ReadInter](#) 实现,调用 [CH347AbortInter](#) 则放弃中断数据读取操作。

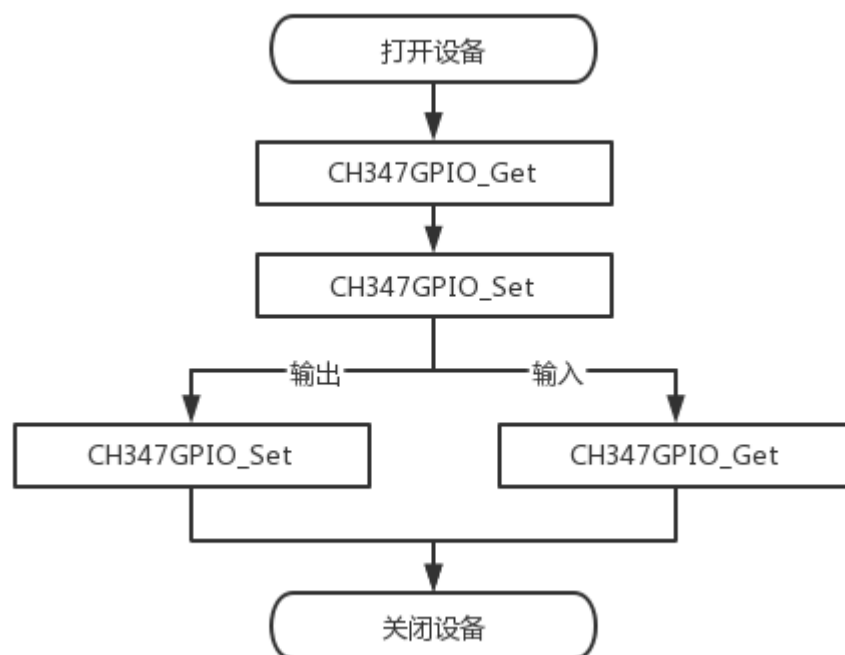


Figure 4.3.1 GPIO 操作流程

函数具体说明请参考以下内容。

### 4.3.2 CH347GPIO\_Get

#### 功能描述

该函数用于获取设备当前的 GPIO 输入输出状态

#### 函数定义

```

BOOL WINAPI
CH347GPIO_Get (ULONG    iIndex,
               UCHAR     *iDir,
               UCHAR     *iData)
  
```

#### 参数说明

`iIndex`: 指定操作设备序号  
`iDir`: 引脚方向:GPIO0-7 对应位 0-7. 0: 输入; 1: 输出  
`iData`: GPIO 电平状态: GPIO 0-7 对应位 0-7, 其中 0 表示低电平, 1 表示高电平

#### 返回值

执行成功返回 1，失败返回 0

### 4.3.3 CH347GPIO\_Set

#### 功能描述

该函数用于设置 CH347-GPIO 的 I/O 方向与输出状态

#### 函数定义

```

BOOL WINAPI
CH347GPIO_Set (ULONG      iIndex,
                UCHAR      iEnable,
                UCHAR      iSetDirOut,
                UCHAR      iSetDataOut)

```

#### 参数说明

iIndex: 指定操作设备序号  
 iEnable: 数据有效标志: 对应位 0-7, 对应 GPIO0-7  
 iSetDirOut: 设置 I/O 方向, 某位清 0 则对应引脚为输入, 某位置 1 则对应引脚为输出。GPIO0-7 对应位 0-7  
 iSetDataOut: 输出数据, 如果 I/O 方向为输出, 那么某位清 0 时对应引脚输出低电平, 某位置 1 时对应引脚输出高电平

#### 返回值

执行成功返回 1，失败返回 0

### 4.3.4 CH347SetIntRoutine

#### 功能描述

该函数用于设定 CH347 - GPIO 中断服务程序

#### 函数定义

```

BOOL WINAPI
CH347SetIntRoutine ( ULONG  iIndex,
                    UCHAR  Int0PinN,
                    UCHAR  Int0TripMode,
                    UCHAR  Int1PinN,
                    UCHAR  Int1TripMode,
                    mPCH347_INT_ROUTINE iIntRoutine );

```

#### 参数说明

iIndex: 指定操作设备序号  
 Int0PinN: 中断 GPIO 引脚号, 大于 7: 不启用此中断源; 为 0-7 对应 gpio0-7  
 Int0TripMode: 中断类型: 00: 下降沿触发; 01: 上升沿触发;  
                   02: 双边沿触发; 03: 保留;  
 Int1PinN: 中断 GPIO 引脚号, 大于 7 则不启用此中断源, 为 0-7 对应 gpio0-7  
 Int1TripMode: 中断类型: 00: 下降沿触发; 01: 上升沿触发;  
                   02: 双边沿触发; 03: 保留;  
 iIntRoutine: 指定中断服务程序, 为 NULL 则取消中断服务, 否则在中断时调用该程序

**返回值**

执行成功返回 1，失败返回 0

**4.3.5 CH347ReadInter****功能描述**

该函数用于读取中断数据

**函数定义**

```
BOOL WINAPI  
CH347ReadInter( ULONG    iIndex,  
                PCHAR    iStatus)
```

**参数说明**

iIndex: 指定操作设备序号  
iStatus : 指向字节单元, 用于保存读取 GPIO 引脚状态数据, 参考下面的位说明

**返回值**

执行成功返回 1，失败返回 0

**4.3.6 CH347AbortInter****功能描述**

该函数用于取消读取中断数据

**函数定义**

```
BOOL WINAPI  
CH347AbortInter(ULONG    iIndex )
```

**参数说明**

iIndex: 指定操作设备序号

**返回值**

执行成功返回 1，失败返回 0