CH347 Application Development Manual

V1.5

Catalog

1. Introduction

CH347 is a USB2.0 high-speed converter chip to implement USB to UART (HID serial port/VCP serial port), USB to SPI, USB to I2C, USB to JTAG and USB to GPIO interfaces, which are included in the chip's four working modes.

CH347DLL is used to provide UART/SPI/I2C/JTAG/BitStream interface operation functions for CH347/CH339W chip on the OS side, and supports CH341 vendor/HID/VCP driver interfaces, so there is no need to distinguish between driver interface and chip working mode when using it.

2. Interface description

According to the characteristics of USB converter interface supported by CH347, CH347DLL provides interface functional functions for USB-UART (HID serial port/VCP serial port), USB-SPI, USB-I2C, USB-JTAG, and USB-GPIO, including the basic functional function and the corresponding functional function, such as EEPROM read/write and SHIFT-DR state read/write in JTAG application.

The CH347F can use all interfaces without switching modes, and the supported interfaces are shown in the table below:

Functional Interface Description	Driver Interface	API
Interface0: USB2.0 to high speed UART0	CH343SER(VCP)	Native serial port API in the system or CH347UART_xxx in CH347DLL
Interface1: USB2.0 to high speed UART1		
Interface2: USB2.0 to JTAG + SPI + I2C, etc.	CH347PAR	CH347SPI_xxx,CH347I2C_xxx,CH347JTAG_xxx in CH347DLL

The following table lists the ports supported by CH347T, switching between modes via MODE configuration pin level combinations at power-on.

Working Mode	Functional Interface Description	Driver Interface	API
Mode 0	Interface 0: USB2.0 to High-speed UART0	CH343SER(VCP)	Native UART API in the system or CH347UART_xxx in CH347DLL
	Interface 1: USB2.0 to High-speed UART1		
Mode 1	Interface 0: USB2.0 to High-speed UART1	CH343SER(VCP)	Native serial port API in the system or CH347UART_xxx in CH347DLL
	Interface 1: USB2.0 to SPI+I2C	CH347PAR	CH347SPI_xxx, CH347I2C_xxx in CH347DLL
Mode 2	Interface 0: USB2.0 HID to High-speed UART1	HID driver (System-provided)	CH347UART_xxx

	Interface 1: USB2.0 HID to SPI+I2C		CH347SPI_xxx, CH347I2C_xxx in CH347DLL
Mode 3	Interface 0: USB2.0 to High-speed UART1	CH343SER(VCP)	Native serial port in the system or CH347UART_xxx in CH347DLL
	Interface 1: USB2.0 to JTAG+I2C	CH347PAR	CH347JTAG_xxx in the CH347DLL CH347I2C_xxx

Table. CH347 Interface function API

3. Synchronous serial interface

3.1 Related data types

```
// Driver interfaces
#define CH347_USB_CH341          0
#define CH347_USB_HID            2
#define CH347_USB_VCP            3

// Chip function interface number
#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

// Chip model definition
#define CHIP_TYPE_CH341          0
#define CHIP_TYPE_CH347          1
#define CHIP_TYPE_CH347F         2
#define CHIP_TYPE_CH339W         3
#define CHIP_TYPE_CH347T CHIP_TYPE_CH347
```

3.1.1 SPI controller information

```
// SPI Controller Configuration
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= LSB, 1= MSB
    USHORT    iSpiWriteReadInterval; // SPI Interface general read and write data
                                    command, the unit is uS
    UCHAR     iSpiOutDefaultData;   // SPI prints data by default when it reads data
    ULONG     iChipSelect;          // Chip selection, bit7 = 0, chip selection control is
```

```
                                    ignored, bit7=1, parameters valid: bit1/0 are 00/01 then
                                    CS1/CS2 pins are selected as low level active chip select
                                    respectively
    UCHAR       CS1Polarity;        // Bit 0: chip selection CS1 polarity control,
                                    0: active low; 1: active high
    UCHAR       CS2Polarity;        // Bit 0: chip selection CS2 polarity control,
                                    0: active low; 1: active high
    USHORT      iIsAutoDeativeCS;   // Whether to automatically undo the chip
                                    selection after the operation is completed
    USHORT      iActiveDelay;       // Delay time for performing read and write
                                    operations after setting the chip selection, the unit is uS.
    ULONG       iDelayDeactive;     // Delay time for executing read/write operations
                                    after undoing chip selection ,the unit is uS
}mSpiCfgS,*mPSpiCfgS;
```

3.1.2 Device information

```
typedef struct _DEV_INFOR{
    UCHAR       iIndex;             // Currently open serial number
    UCHAR       DevicePath[MAX_PATH];   // Device link name, used in CreateFile.
    UCHAR       UsbClass;           // Driver category 0:CH347_USB_CH341,
                                    2:CH347_USB_HID, 3:CH347_USB_VCP
    UCHAR       FuncType;           // Functional category 0:CH347_FUNC_UART,
                                    1:CH347_FUNC_SPI_I2C,2:CH347_FUNC_JTAG_I2C
                                    3:CH347_FUNC_JTAG_IIC_SPI
    CHAR        DeviceID[64];       // USB\VID_xxxx&PID_xxxx
    UCHAR       Mode;               // Chip working mode
                                    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;          // The device handle
USHORT      BulkOutEndpMaxSize;     // Bulk upload endpoint size
USHORT      BulkInEndpMaxSize;      // Bulk download endpoint size
UCHAR       UsbSpeedType;           // USB speed type, 0: FS, 1: HS, 2: SS
    UCHAR       CH347FuncType;      // USB interface number
                                    CH347T: IF0:UART;   IF1:SPI/IIC/JTAG/GPIO
                                    CH347F: IF0:UART0;   IF1:UART1;
                                            IF2:SPI/IIC/JTAG/GPIO
UCHAR       DataUpEndp;             // Bulk upload endpoint address
UCHAR       DataDnEndp;             // Bulk download endpoint address
CHAR        ProductString[64];      // USB product string
CHAR        ManufacturerString[64]; // USB vendor string
ULONG       WriteTimeout;           // USB write timeout
ULONG       ReadTimeout;            // USB read timeout
CHAR        FuncDescStr[64];        // Interface functional descriptor
```

```
UCHAR    FirewareVer;              // Firmware version, hexadecimal value
}mDeviceInforS,*mPDeviceInforS
```

3.2 Public operation functions

3.2.1 CH347OpenDevice

Function description

This function is used to turn on CH347 device, supports the opening of SPI/I2C/JTAG interfaces in all modes of CH347.

Function definitions

```
HANDLE  WINAPI
CH347OpenDevice(    ULONG    DevI);
```

Parameter description

DevI: Specify the serial number of the operating device

Return value

Returns the device serial number if the execution is successful.

3.2.2 CH347CloseDevice

Function description

This function is used to close CH347 device, you can disable SPI/I2C/JTAG interfaces in all CH347 modes.

Function definitions

```
BOOL  WINAPI
CH347CloseDevice(    ULONG  iIndex)
```

Parameter description

iIndex: Specify the serial number of the operating device

Return value

The return value is 1 on success and 0 on failure

3.2.3 CH347SetDeviceNotify

Function description

This function is used to specify the device event notification function, it can be used for dynamic hot plug detection of SPI/I2C/JTAG interfaces in all modes of CH347.

Function definitions

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

Parameter descriptions

iIndex: Specify the serial number of the operating device

iDeviceID: Optional parameter, pointing to a string, specifies the ID of the monitored device, the string terminated with \0.

iNotifyRoutine: Specify the device event callback program. If it is NULL, event notification is cancelled. Otherwise the program is called when the event is detected.

Return value

The return value is 1 on success and 0 on failure

Annotations

iDeviceID is a variable parameter. To implement CH347 device hot plug detection, you can define macros as follows

#define CH347DevID "VID_1A86&PID_55D\0"

During parameter transmission, replace iDeviceID with CH347DevID to implement dynamic hot plug detection for CH347 synchronous serial interface.

To accurately detect the plugging and unplugging action actions of interfaces in each mode, write down the complete USBID, taking the SPI interface in mode 1 as an example, you can define the following macro.

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

During parameter transmission, replace iDeviceID with USBID_VEN_SPI_I2C to implement dynamic hot plug detection for SPI&I2C interfaces in CH347 mode 1.

For other interface settings, see 3.2.9 Interface dynamic hot plug detection.

3.2.4 CH347GetDeviceInfor

Function description

This function is used to get the current interface mode and VID/PID of the device.

Function definitions

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

Parameter descriptions

iIndex:	Specify the serial number of the operating device
DevInformation:	Device information structure

Return value

The return value is 1 on success and 0 on failure

Annotations

Device information structure, see DEV_INFOR

3.2.5 CH347GetSerialNumber

Function description

This function is used to get the USB serial number of the device.

Function definitions

```
BOOL    WINAPI
CH347GetSerialNumber(ULONG       iIndex,
                     PUCHAR      iSerialNumberStr)
```

Parameter descriptions

iIndex:	Specify the serial number of the operating device
iSerialNumberStr:	Point to the obtained device serial number

Return value

The return value is 1 on success and 0 on failure

3.2.6 CH347GetChipType

Function description

This function is used to get CH347 type:

0:CHIP_TYPE_CH341,
1:CHIP_TYPE_CH347/CHIP_TYPE_CH347T,
2:CHIP_TYPE_CH347F,
3:CHIP_TYPE_CH339W.

Function definitions

```
BOOL    WINAPI
CH347GetChipType(    ULONG      iIndex)
```

Parameter descriptions

iIndex: Specify the serial number of the operating device

Return value

The return value is 1 on success and 0 on failure

Annotations

Return UCHAR type, meaning refers to the definition of the reference chip model.

3.2.7 CH347GetVersion

Function description

This function is used to get driver version, library version, device version, chip type (CH341(FS)/ CH347HS).

Function definitions

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

Parameter descriptions

iIndex: Specify the serial number of the operating device
iDriverVer: Driver version information
iDLLVer: Library version information
ibcdDevice: Device version information
iChipType: The chip type

Return value

The return value is 1 on success and 0 on failure.

3.2.8 CH347SetTimeout

Function description

This function is used to set timeout for USB data reads and writes.

Function definitions

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

Parameter descriptions

iIndex: Specify the serial number of the operating device
iWriteTimeout: Specify the timeout for USB write-out data blocks, the unit is millisecond (mS), 0xFFFFFFFF specifies no timeout (default)
iReadTimeout: Specify the timeout for USB read data blocks, the unit is millisecond (mS),

0xFFFFFFFF specifies no timeout (default)

Return value

The return value is 1 on success and 0 on failure

3.2.9 Interface dynamic hot plug detection

Detection of synchronous serial interface dynamic hot plug information can be achieved through the CH347SetDeviceNotify function, the code reference is as follows.

Enable the monitoring of USB plug and unplug of CH347 synchronous serial port:

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

Disable the monitoring of USB plug and unplug on CH347 synchronous serial port, be sure to close the program when it exits.

```
CH347SetDeviceNotify(DevIndex, USBDevID, NULL);
// CH347 device hot plug detection notification program
VOID    CALLBACK    UsbDevPnpNotify (ULONG iEventStatus )
{
    // Device plug event, already plugged
        if(iEventStatus==CH347_DEVICE_ARRIVAL)
        PostMessage(DebugHwnd,WM_CH347DevArrive,0,0);
    // Device unplug event, already unplugged
      else if(iEventStatus==CH347_DEVICE_REMOVE)
        PostMessage(DebugHwnd,WM_CH347DevRemove,0,0);
    return;
}
```

To accurately detect the SPI/I2C/JTAG interface plug and unplug information in each mode, write the following complete USBID. Replace iDeviceID with the corresponding USBID macro when using CH347SetDeviceNotify.

```
//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"
```

3.2.10 Device enumeration operation

In this library, the API implements corresponding operations by specifying device serial numbers. The device serial number is generated based on the sequence of devices being inserted one by one. The device enumeration function can be implemented by opening the corresponding device serial number through the device Open function and determining whether the device exists and is valid according to the return value of the function.

The SPI/I2C/JTAG interface is turned on/off by CH347OpenDevice/ CH347CloseDevice.

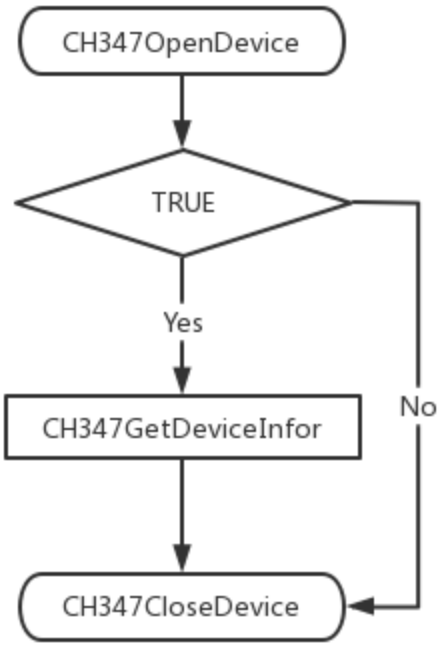


Figure 3.2.10 Device enumeration flowchart

3.3 SPI functions

3.3.1 Operation process

After the device is enabled, set the device USB read and write timeout parameters, configure the SPI controller parameters for SPI initialization settings, after successful setup, you can communicate with the device by calling the SPI read and write function.

The function call flowchart is as follows:

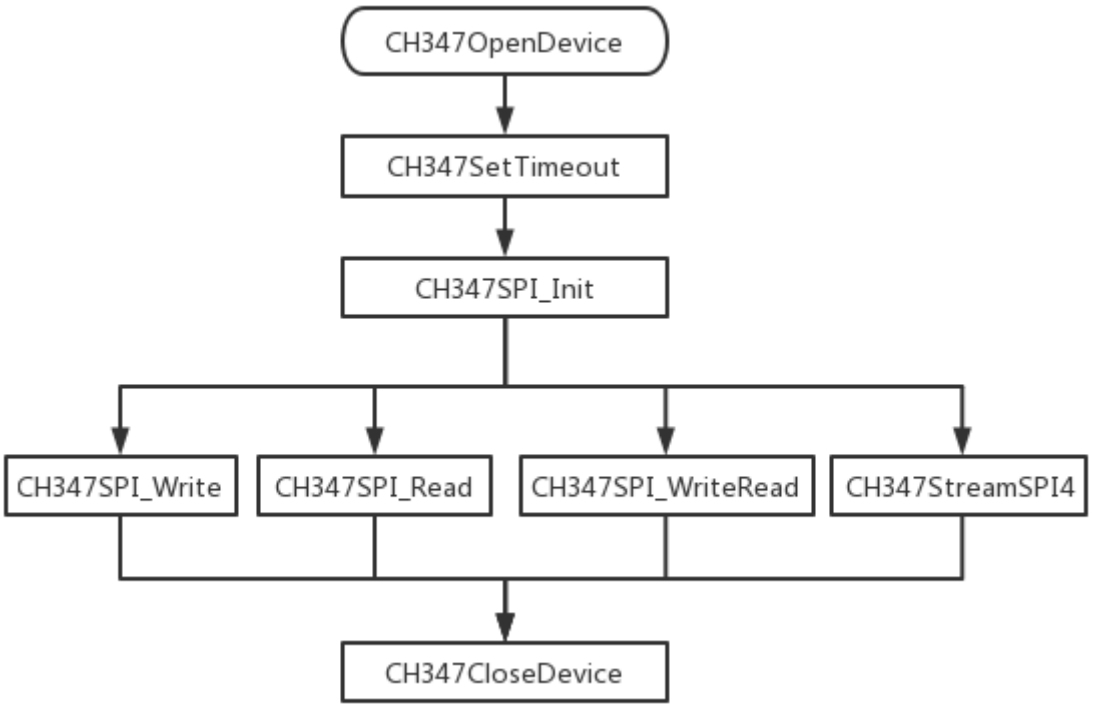


Figure 3.3.1 SPI Function operation flowchart

For details about the function, see the following.

3.3.2 CH347SPI_Init

Function description

This function is used to configure parameters on the SPI controller.

Function definitions

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

Parameter descriptions

iIndex:	Specify the serial number of the operating device
SpiCfg:	SPI controller configuration

Return value

The return value is 1 on success and 0 on failure

Annotations

For the configuration of the SPI controller, see structure _SPI_CONFIG

3.3.3 CH347SPI_SetFrequency

Function description

This function is used to set the SPI clock frequency, and after calling this interface, you need to call CH347SPI_Init again for reinitialisation.

Function definitions

```
BOOL    WINAPI
CH347SPI_SetFrequency( ULONG    iIndex,
                       ULONG    iSpiSpeedHz)
```

Parameter descriptions

iIndex：	Specify the serial number of the operating device
iSpiSpeedHz：	Set the SPI clock, with the unit in Hz

Return value

The return value is 1 on success and 0 on failure

3.3.4 CH347SPI_SetDataBits

Function description

This function is used to set the number of bits of SPI supported data for the CH347F

Function definitions

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

Parameter descriptions

iIndex：	Specify the serial number of the operating device
iDataBits：	SPI data bits, 0 means 8bit, 1 means 16bit

Return value

The return value is 1 on success and 0 on failure

3.3.5 CH347SPI_GetCfg

Function description

This function is used to get the current configuration of the SPI controller.

Function definitions

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

Parameter descriptions

iIndex:	Specify the serial number of the operating device
SpiCfg:	SPI controller configuration

Return value

The return value is 1 on success and 0 on failure

Annotations

For the configuration of the SPI controller, see structure _SPI_CONFIG

3.3.6 CH347SPI_ChangeCS

Function description

This function is used to set the chip selection status, you need to call CH347SPI_Init to set the CS before use

Function definitions

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

Parameter descriptions

iIndex:	Specify the serial number of the operating device
iStatus:	0 = Undo chip selection, 1 = Set chip selection

Return value

The return value is 1 on success and 0 on failure

3.3.7 CH347SPI_SetChipSelect

Function description

This function is used to set the SPI chip selection.

Function definitions

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

Parameter descriptions

iIndex:	Specify the serial number of the operating device
iEnableSelect:	The lower 8 bits are CS1 and the higher 8 bits are CS2; byte value of 0= set CS, 1= ignore this CS setting
iChipSelect:	The lower octet is CS1 and the higher octet is CS2. Piece of selected output, 0= Undo chip selection, 1=set chip selection
iIsAutoDeativeCS:	The lower 16 bits are CS1, the higher 16 bits are CS2; whether to undo chip

selection automatically after the operation is complete.

iActiveDelay: The lower 16 bits are CS1, the higher 16 bits are CS2;
Delay time for performing read and write operations after setting the chip selection, the unit is uS.

iDelayDeactive: The lower 16 bits are CS1, the higher 16 bits are CS2;delay time for read and write operations after chip selection is unselected, the unit is uS.

Return value

The return value is 1 on success and 0 on failure

3.3.8 CH347SPI_Write

Function description

This function is used to the SPI write data

Function definitions

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

Parameter descriptions

iIndex: Specify the serial number of the operating device

iChipSelect: Chip selection, bit 7 is 0 to ignore chip select control,
bit 7 is 1 for chip select operation.

iLength: Number of bytes of data to be transferred

iWriteStep: The length of a single block to be read

ioBuffer: Point to a buffer, place the data to be written-out from MOSI

Return value

The return value is 1 on success and 0 on failure

3.3.9 CH347SPI_Read

Function description

This function is used to read SPI data

Function definitions

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

Parameter descriptions

iIndex: Specify the serial number of the operating device

iChipSelect: Chip selection, bit 7 is 0 to ignore chip select control,
bit 7 is 1 for chip select operation.

oLength: Number of bytes of data to send

iLength: The length of data to be read in bytes

ioBuffer: Point to a buffer, place the data to be written-out from MOSI,

the returned data is the data read-in from MISO.

Return value

The return value is 1 on success and 0 on failure

3.3.10 CH347SPI_WriteRead

Function description

This function is used to write and read SPI data streams

Function definitions

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

Parameter descriptions

iIndex:	Specify the serial number of the operating device
iChipSelect:	Chip selection, bit 7 is 0 to ignore chip select control, bit 7 is 1 for chip select operation.
iLength:	Number of bytes of data to send
ioBuffer:	Point to a buffer, place the data to be written-out from MOSI, the returned data is the data read-in from MISO.

Return value

The return value is 1 on success and 0 on failure

3.3.11 CH347StreamSPI4

Function description

This function is used to process the SPI data stream, read data while writing

Function definitions

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

Parameter descriptions

iIndex:	Specify the serial number of the operating device
iChipSelect:	Chip selection, bit 7 is 0 to ignore chip select control, bit 7 is 1 for chip select operation.
iLength:	Number of bytes of data to send
ioBuffer:	Point to a buffer, place the data to be written-out from MOSI, the returned data is the data read-in from MISO.

Return value

The return value is 1 on success and 0 on failure

3.4 JTAG functions

3.4.1 Operation process

After turning on the device, Use CH347Jtag_INIT to initialize the device;

Use CH347Jtag_SwitchTapState(0) to reset the JTAG TAP status of the target device to Test-Logic-Reset, you can use the corresponding function to switch to SHIFT-DR/SHIFT-IR for read/write operations as required, there are two ways to read/write, which are bitband mode and batch fast mode, select according to actual use.

The function call flowchart is as follows:

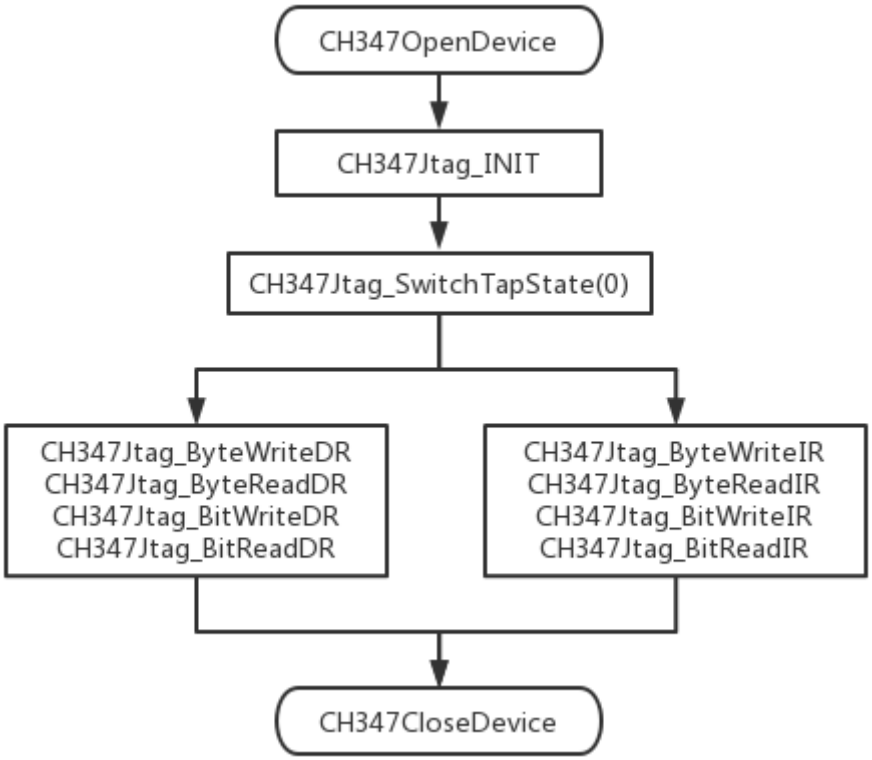


Figure 3. 4.1 JTAG Function operation flowchart

For details about the function, see the following.

3.4.2 CH347Jtag_INIT

Function description

This function is used to initialize the JTAG interface and set the communication speed.

Function definitions

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

Parameter descriptions

iIndex:	Specify the serial number of the operating device
iClockRate:	Communication speed; The value ranges from 0 to 5, a larger value indicates a faster communication speed

Return value

The return value is 1 on success and 0 on failure

3.4.3 CH347Jtag_TmsChange

Function description

This function is used to pass in the TMS value for the corresponding state switching, the TMS value refers to the JTAG TAP state machine.

Function definitions

```
BOOL    WINAPI
CH347Jtag_TmsChange( ULONG      iIndex,
                     PUCHAR     tmsValue,
                     ULONG      Step,
                     ULONG      Skip);
```

Parameter descriptions

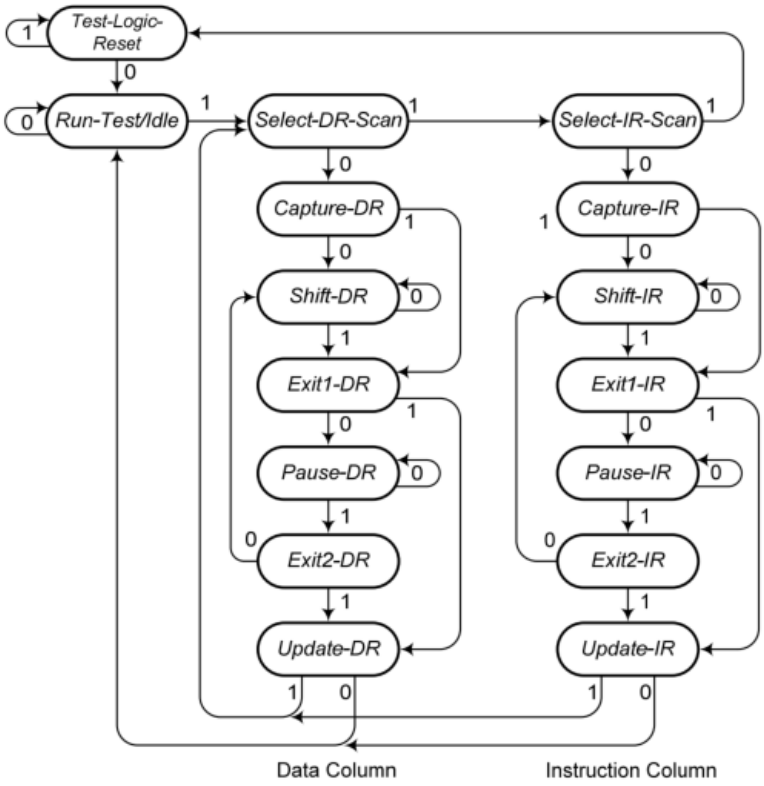
iIndex：	Specify the serial number of the operating device
tmsValue：	TMS bit value for switching, in bytes
Step：	Number of valid TMS bits stored within tmsValue
Skip：	Valid start bit

Return value

The return value is 1 on success and 0 on failure

Example

Refer to the JTAG TAP state machine as shown in the figure below. The example uses CH347Jtag_TmsChange to complete the transition from the IDLE state to Shift-IR for reading and writing, then switches to Shift-DR for reading and writing, and finally returns to the IDLE state.



NOTE—The value shown adjacent to each state transition in this figure represents the signal present at TMS at the time of a rising edge at TCK.

Pseudocode example

```
// Enter the processing flow (TMS values can refer to the TAP state machine in the above image).

// Initialise TMS value.
tmsValue = [0x03]
// State transition：IDLE --> Select DR --> Select IR --> Capture IR --> Shift-IR
// TMS Value：          1          1          0          0
call CH347Jtag_TmsChange(iIndex, tmsValue, 4, 0)
// Perform read and write operations on Shift-IR.
call CH347Jtag_IoScan(iIndex, ir_code, ir_len, true)

// Reinitialise TMS value.
tmsValue = [0x03]
// State transition：Exit-IR --> Update IR --> Select DR --> Capture DR --> Shift-DR
// TMS Value：          1          1          0          0
call CH347Jtag_TmsChange(iIndex, tmsValue, 4, 0)
// Perform read and write operations on Shift-DR.
call CH347Jtag_IoScan(iIndex, dr_code, dr_len, true)

// Reinitialise TMS value.
tmsValue = [0x01]
// State transition：Exit-DR --> Update DR --> IDLE
// TMS Value：          1          0
call CH347Jtag_TmsChange(iIndex, tmsValue, 2, 0)
```

3.4.4 CH347Jtag_IoScan

Function description

This function is mainly used for SHIFT-DR/IR state to perform data reading and writing, and finally switch to EXIT-DR/IR at the end of reading and writing, state switching can be used with CH347Jtag_TmsChange.

Function definitions

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

Parameter descriptions

iIndex：	Specify the serial number of the operating device
DataBits：	Data that needs to be transferred
DataBitsNb：	Number of bits of data to be transmitted
IsRead：	Whether the data needs to be read

Return value

The return value is 1 on success and 0 on failure

3.4.5 CH347Jtag_IoScanT

Function description

This function can be called several times in the SHIFT-DR/IR state to realize data reading and writing, through

the IsLastPkt to determine whether the end of reading and writing to switch to EXIT-DR/IR, state switching can be used with CH347Jtag_TmsChange.

Function definitions

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

Parameter descriptions

iIndex：Specify the serial number of the operating device

DataBits：Data that needs to be transferred

DataBitsNb：Number of bits of data to be transmitted

IsRead：Whether the data needs to be read

IsLastPkt: Whether it is the last packet of data, if TRUE, the last 1bit of data will be switched to EXIT-DR/IR for transmission

Return value

The return value is 1 on success and 0 on failure

3.4.6 CH347Jtag_Reset

Function description

This function can reset the Tap status function. The TMS is high for more than six clocks, Tap state on Test-Logic Reset

Function definitions

```
BOOL    WINAPI
CH347Jtag_Reset(    ULONG    iIndex);
```

Parameter descriptions

iIndex：Specify the serial number of the operating device

Return value

The return value is 1 on success and 0 on failure

3.4.7 CH347Jtag_ResetTrst

Function description

This function perform TRST operations to reset hardware

Function definitions

```
BOOL    WINAPI
CH347Jtag_ResetTrst(  ULONG    iIndex,
                      BOOL     iLevel);
```

Parameter descriptions

iIndex：Specify the serial number of the operating device

iLevel：0=Set lower,1=Set High

Return value

The return value is 1 on success and 0 on failure

3.4.8 CH347Jtag_WriteRead

Function description

This function reads and writes SHIFT-DR /IR state data in bitband mode, suitable for read and write small amounts of data. Such as command operation, state machine switching and other control transmission operations. If you need to transfer bulk data, you can use the CH347Jtag_WriteRead_Fast command package to transfer data in bytes.

Function definitions

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

Parameter descriptions

iIndex:	Specify the serial number of the operating device
IsDR:	Determine the switchover status for read and write, TRUE= SHIFT-DR, FALSE=SHIFT-IR
iWriteBitLength:	The length of data to be written
iWriteBitBuffer:	Point to a buffer, place the data to be written-out.
oReadBitLength:	Point to a length element, the return value is the actual length of data read.
oReadBitBuffer:	Point to a large enough buffer, used to save data that has been read.

Return value

The return value is 1 on success and 0 on failure

Annotations

This function uses the value of IsDR to determine whether to operate the JTAG state to switch to SHIFT-DR or SHIFT-IR state, and then switch back to RUN-TEST state after read and write data in bitband mode, the status switch path is as follows:

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

3.4.9 CH347Jtag_WriteRead_Fast

Function description

This function is used to switch to the SHIFT-IR /DR state for batch data read/write, for multi-byte sequential read/write, such as JTAG firmware download operation.

Function definitions

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

Parameter descriptions

iIndex:	Specify the serial number of the operating device.
IsDR:	Determine the switchover status for read and write.

TRUE = SHIFT-DR, FALSE = SHIFT-IR.

iWriteBitLength: The length of bytes to write out data.

iWriteBitBuffer: Point to a buffer, place the data to be written-out.

oReadBitLength: Point to a length element, the return value is the actual length of data read.

oReadBitBuffer: Point to a large enough buffer, used to save data that has been read.

Return value

The return value is 1 on success and 0 on failure

Annotations

This function is similar to CH347Jtag_WriteRead, but this function is used for bulk data reads and writes, read and write data in byte.

3.4.10 CH347Jtag_WriteReadEx

Function description

This function performs shift IR/DR data read and write in bit-banding mode. It is suitable for reading and writing small amounts of data, such as control transfers for instruction operations and state machine switching. For bulk data transfers, it is recommended to use CH347Jtag_WriteReadEx_Fast.

CH347Jtag_WriteRead extension function, supports continuous read and write while remaining in the Shift-DR/IR state, can be used in conjunction with CH347Jtag_TmsChange.

Function definitions

```
BOOL    WINAPI
CH347Jtag_WriteReadEx( ULONG     iIndex,
                       BOOL      IsInDrOrIr,
                       BOOL      IsDR,
                       ULONG     iWriteBitLength,
                       PVOID     iWriteBitBuffer,
                       PULONG    oReadBitLength,
                       PVOID     oReadBitBuffer );
```

Parameter descriptions

iIndex: Specify the serial number of the operating device.

IsInDrOrIr: TRUE: In SHIFT-DR read/write

FALSE: Run-Test->Shift-IR/DR.(data read/write).->Exit IR/DR -> Run-Test

IsDR: Determine the switchover status for read and write.

TRUE = SHIFT-DR, FALSE = SHIFT-IR.

iWriteBitLength: The length of bytes to write out data.

iWriteBitBuffer: Point to a buffer, place the data to be written-out.

oReadBitLength: Point to a length element, the return value is the actual length of data read.

oReadBitBuffer: Point to a large enough buffer, used to save data that has been read.

Return value

The return value is 1 on success and 0 on failure

3.4.11 CH347Jtag_WriteRead_FastEx

Function description

This function is used to switch to the SHIFT-IR /DR state for batch data read/write, for multi-byte sequential read/write, such as JTAG firmware download operation.

The CH347Jtag_WriteRead_Fast extended function supports continuous reading and writing while remaining in the Shift-DR/IR state, and can be used in conjunction with CH347Jtag_TmsChange.

Function definitions

```
BOOL    WINAPI
CH347Jtag_WriteRead_FastEx( ULONG     iIndex,
                            BOOL      IsInDrOrIr,
                            BOOL      IsDR,
                            ULONG     iWriteBitLength,
                            PVOID     iWriteBitBuffer,
                            PULONG    oReadBitLength,
                            PVOID     oReadBitBuffer );
```

Parameter descriptions

iIndex:	Specify the serial number of the operating device.
IsInDrOrIr:	TRUE: In SHIFT-DR read/write FALSE: Run-Test->Shift-IR/DR.(data read/write).->Exit IR/DR -> Run-Test
IsDR:	Determine the switchover status for read and write. TRUE = SHIFT-DR, FALSE = SHIFT-IR.
iWriteBitLength:	The length of bytes to write out data.
iWriteBitBuffer:	Point to a buffer, place the data to be written-out.
oReadBitLength:	Point to a length element, the return value is the actual length of data read.
oReadBitBuffer:	Point to a large enough buffer, used to save data that has been read.

Return value

The return value is 1 on success and 0 on failure

Annotations

This function is similar to CH347Jtag_WriteReadEx, but this function is used for bulk data reads and writes, read and write data in byte.

3.4.12 CH347Jtag_SwitchTapState

Function description

This function is used to switch the JTAG state machine state

Function definitions

```
BOOL CH347Jtag_SwitchTapState(UCHAR    TapState)
```

Parameter description

TapState: Enter the serial number to switch the status.

Return value

The return value is 1 on success and 0 on failure

Annotations

The TapState status switch is described as follows:

0: Reset the status of the target device to Test-Logic Reset
1: Follow the previous state to enter 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/IR -> Update-DR/IR ->Run-Test-Idle

3.4.13 CH347Jtag_SwitchTapStateEx

Function description

This function is single-step switching of the JTAG state machine allows for the specification of the operating device iIndex.

Function definitions

```
BOOL CH347Jtag_SwitchTapState( ULONG     iIndex,
                               UCHAR    TapState)
```

Parameter description

iIndex:	Specify the serial number of the operating device.
TapState:	Enter the serial number to switch the status.

Return value

The return value is 1 on success and 0 on failure

3.4.14 CH347Jtag_ByteWriteDR

Function description

This function is used to switch the JTAG state machine to SHIFT-DR state in byte units, allowing for multi-byte sequential read and write.

Function definitions

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

Parameter descriptions

iIndex:	Specify the serial number of the operating device
iWriteLength:	The length of bytes to write-out data.
iWriteBuffer:	Point to a buffer, place the data to be written-out.

Return value

The return value is 1 on success and 0 on failure

3.4.15 CH347Jtag_ByteReadDR

Function description

This function is used to switch the JTAG state machine to SHIFT-DR state in byte units, allowing for multi-byte sequential read and write.

Function definitions

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

Parameter descriptions

iIndex:	Specify the serial number of the operating device
oReadLength:	The length of bytes to be read
oReadBuffer:	Point to a buffer, place the data to be read.

Return value

The return value is 1 on success and 0 on failure

3.4.16 CH347Jtag_ByteWriteIR

Function description

This function is used to switch the JTAG state machine to SHIFT-IR state in byte units, allowing for multi-

byte sequential read and write.

Function definitions

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

Parameter descriptions

iIndex:	Specify the serial number of the operating device
iWriteLength:	The length of bytes to write-out data.
iWriteBuffer:	Point to a buffer, place the data to be written-out.

Return value

The return value is 1 on success and 0 on failure

3.4.17 CH347Jtag_ByteReadIR

Function description

This function is used to switch the JTAG state machine to SHIFT-IR state in byte units, allowing for multi-byte sequential read and write.

Function definitions

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

Parameter descriptions

iIndex:	Specify the operating device number
oReadLength:	The length of bytes to be read.
oReadBuffer:	Point to a buffer, place the data to be read

Return value

The return value is 1 on success and 0 on failure

3.4.18 CH347Jtag_BitWriteDR

Function description

This function is used to switch the JTAG state machine to shift-DR state, data is read and write in bitband mode.

Function definitions

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

Parameter descriptions

iIndex:	Specify the serial number of the operating device.
iWriteLength:	The length of bytes to write-out data.
iWriteBuffer:	Point to a buffer, place the data to be written-out.

Return value

The return value is 1 on success and 0 on failure

3.4.19 CH347Jtag_BitWriteIR

Function description

This function is used to switch the JTAG state machine to shift-IR state, data is read and write in bitband mode.

Function definitions

```
BOOL    WINAPI
CH347Jtag_BitWriteIR(ULONG     iIndex,
                     ULONG     iWriteLength,
                     PVOID     iWriteBuffer);
```

Parameter descriptions

iIndex:	Specify the serial number of the operating device.
iWriteLength:	The length of bytes to write-out data.
iWriteBuffer:	Point to a buffer, place the data to be written-out.

Return value

The return value is 1 on success and 0 on failure

3.4.20 CH347Jtag_BitReadIR

Function description

This function is used to switch the JTAG state machine to SHIFT-IR state, data is read and write in bitband mode.

Function definitions

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

Parameter descriptions

iIndex:	Specify the serial number of the operating device.
oReadLength:	The length of bytes to be read.
oReadBuffer:	Point to a buffer, place the data to be read.

Return value

The return value is 1 on success and 0 on failure

3.4.21 CH347Jtag_BitReadDR

Function description

This function is used to switch the JTAG state machine to SHIFT-DR state in byte units, allowing for multi-byte sequential read and write.

Function definitions

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

Parameter descriptions

iIndex:	Specify the serial number of the operating device.
oReadLength:	The length of bytes to be read.
oReadBuffer:	Point to a buffer, place the data to be read.

Return value

The return value is 1 on success and 0 on failure

3.5 I2C functions

3.5.1 Operation process

Open the specified operating device to get the serial number of the device, set the I2C interface speed/SCL frequency of the device, and perform I2C read/write operations. The function call flowchart is as follows:

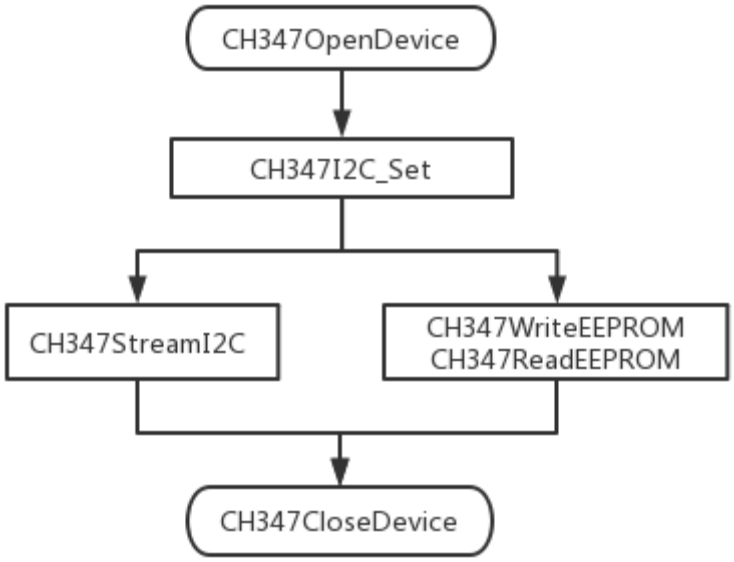


Figure 3.5.1 I2C operation flowchart

For details about the function, see the following.

3.5.2 Related data types

EEPROM types

```
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

Function description

This function is used to specify the operating device and set the I2C interface speed/SCL frequency.

Function definitions

```
BOOL    WINAPI
```

CH347I2C_Set(ULONG iIndex,
ULONG iMode)

Parameter descriptions

iIndex: Specify the serial number of the operating device

iMode: Set the mode
Bit 2-0: 000 = low speed /20KHz,
001 = standard /100KHz(default),
010 = fast speed /400KHz,
011 = high speed /750KHz,
100 = 50KHz, 101 = 200KHz, 110 = 1MHz
Bit 7-3: Reserved as 0

Return value

The return value is 1 on success and 0 on failure

3.5.4 CH347I2C_SetStretch

Function description

This function is used to set the clock extension function.

Function definitions

BOOL WINAPI
CH347I2C_SetStretch(ULONG iIndex,
BOOL iEnable);

Parameter descriptions

iIndex Specify the serial number of the operating device

iEnable Whether to enable the clock extension function

Return value

The return value is 1 on success and 0 on failure

3.5.5 CH347I2C_SetDelaymS

Function description

This function is used to set the hardware asynchronous delay and will return soon after being called, specifying the number of milliseconds of delay before the next stream operation.

Function definitions

BOOL WINAPI
CH347I2C_SetDelaymS(ULONG iIndex,
ULONG iDelay) ;

Parameter descriptions

iIndex: Specify the serial number of the operating device.

iDelay: Specifies the number of milliseconds to delay.

Return value

The return value is 1 on success and 0 on failure

3.5.6 CH347I2C_SetDeiverMode

Function description

This function is used to set the I2C pins drive mode.

Function definitions

```
BOOL    WINAPI
CH347I2C_SetDeiverMode( ULONG       iIndex,
                        UCHAR       iMode) ;
```

Parameter descriptions

iIndex:	Specify the serial number of the operating device.
iMode:	0=open-drain mode; 1=push-pull mode

Return value

The return value is 1 on success and 0 on failure

3.5.7 CH347I2C_SetAckClk_DelayuS

Function description

This function is used to set the low cycle delay time for the 8th clock, applicable only to CH347T.

Function definitions

```
BOOL    WINAPI
CH347I2C_SetAckClk_DelayuS(    ULONG       iIndex,
                               ULONG       iDelay) ;
```

Parameter descriptions

iIndex:	Specify the serial number of the operating device.
iDelay:	Specified delay in microseconds.

Return value

The return value is 1 on success and 0 on failure

3.5.8 CH347StreamI2C

Function description

This function is used to process I2C data streams and implement I2C data read/write

Function definitions

```
BOOL    WINAPI
CH347StreamI2C( ULONG       iIndex,
                ULONG       iWriteLength,
                PVOID       iWriteBuffer,
                ULONG       iReadLength,
                PVOID       oReadBuffer )
```

Parameter descriptions

iIndex:	Specify the serial number of the operating device
iWriteLength:	The length of bytes to write-out data
iWriteBuffer:	Point to a buffer, place the data to be written-out. The first byte is usually the I2C device address and read/write direction bit, if the address length exceeds 7, this byte can still be written, and so on.
iReadLength:	The length of bytes to be read
oReadBuffer:	Point to a buffer, the function returns the data read in

Return value

The return value is 1 on success and 0 on failure

3.5.9 CH347StreamI2C_RetACK

Function description

This function is used to process the I2C data stream, realize the reading and writing of I2C data, and return the number of ACKs generated by the read and write operations.

Function definitions

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

Parameter descriptions

iIndex:	Specify the serial number of the operating device
iWriteLength:	Number of data bytes to be written out
iWriteBuffer:	Points to a buffer, placed ready to write the data, the first byte is usually the I2C device address and read/write direction bit, if the address length is more than 7, then this byte can still be written and so on.
iReadLength:	Number of data bytes to be read
oReadBuffer:	Points to a buffer where the function returns with the read data.
rAckCount:	Points to the number of ACKs returned for reads and writes

Return value

The return value is 1 on success and 0 on failure

3.5.10 CH347ReadEEPROM

Function description

This function is used to read data blocks to EEPROM

Function definitions

```
BOOL    WINAPI
CH347WriteEEPROM( ULONG       iIndex,
                  EPROM_TYPE  iEepromID,
                  ULONG       iAddr,
                  ULONG       iLength,
                  PUCHAR      iBuffer )
```

Parameter descriptions

iIndex:	Specify the serial number of the operating device
iEepromID:	Specify the EEPROM model
iAddr:	Specify the address of the data unit
iLength:	The length of bytes to be read
iBuffer:	Point to a buffer, place the data to be read.

Return value

The return value is 1 on success and 0 on failure

Annotations

Refer to _EEPROM_TYPE for the models specified by iEepromID

3.5.11 CH347WriteEEPROM

Function description

This function is used to write data blocks to EEPROM

Function definitions

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

Parameter descriptions

iIndex:	Specify the serial number of the operating device
iEepromID:	Specify the EEPROM model
iAddr:	Specify the address of the data unit
iLength:	Length of data bytes to be written-out
iBuffer:	Point to a buffer, place the data to be written-out.

Return value

The return value is 1 on success and 0 on failure

Annotations

Refer to _EEPROM_TYPE for the models specified by iEepromID

4. Asynchronous serial interface functions

4.1 Public functions

4.1.1 Interface dynamic hot plug detection

Detecting the dynamic hot plug information of CH347 UART interface can be implemented by CH347Uart_SetDeviceNotify function, the code can be referred to 3.2.7 Interface dynamic hot plug detection.

Enable CH347 UART serial port USB plug and unplug monitoring:

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

Disable CH347 UART serial port USB plug and unplug monitoring, be sure to close the program when it exits.

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

The monitored USBUartDevID can be the following string or your own ID content.

```
//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 Device enumeration operation

In this interface library, the API implements corresponding operation by specifying the device serial number, the device serial number is generated during the insertion of devices one by one according to their insertion order. The device enumeration function can be implemented by opening the corresponding device serial number through the device Open function and determining whether the device exists or is valid according to the function return value.

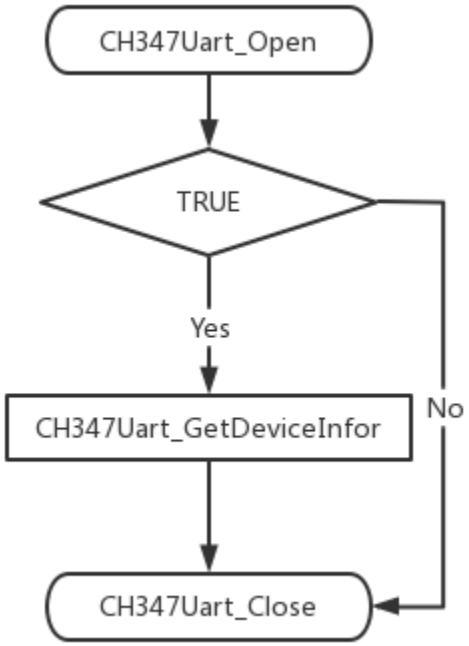


Figure 4.1.2 Device enumeration flowchart

4.2 HID/VCP UART functions

4.2.1 Operation process

After the device is enabled, use the CH347Uart_Open function to open the serial port, set the corresponding serial port parameters and then use the CH347Uart_Init function to set the serial port, then you can use the CH347Uart_Write or CH347Uart_Read function to send and receive serial port data.

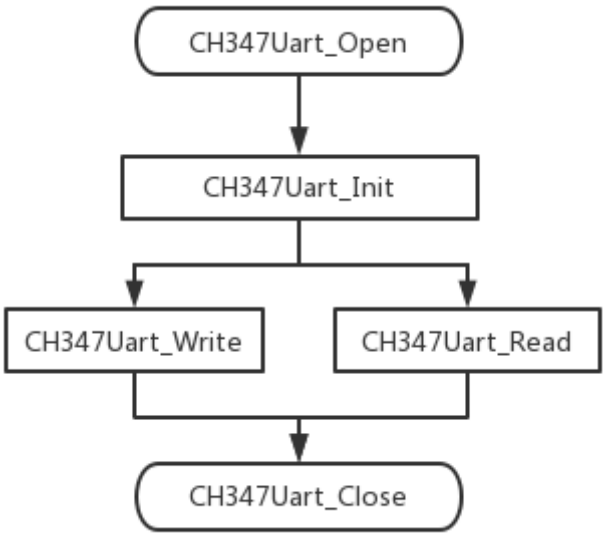


Figure 4.2.1 HID Serial port operation flowchart

For details about the function, see the following.

4.2.2 CH347Uart_Open

Function description

This function is used to open CH347 serial port

Function definitions

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

Parameter description

iIndex: Specify the serial number of the operating device

Return value

The return value is 1 on success and 0 on failure

4.2.3 CH347Uart_Close

Function description

This function is used to close HID serial port

Function definitions

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

Parameter description

iIndex: Specify the serial number of the operating device

Return value

The return value is 1 on success and 0 on failure

4.2.4 CH347Uart_SetDeviceNotify

Function description

This function is used to set the device time notification program, can be used for dynamic hot plug detection of CH347 UART.

Function definitions

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

Parameter descriptions

iIndex:	Specify the serial number of the operating device.
iDeviceID:	Optional parameter, pointing to a string, specifies the ID of the monitored device, the string terminated with \0.
iNotifyRoutine:	Specify the device event callback program. If it is NULL, event notification is cancelled. Otherwise the program is called when the event is detected.

Return value

The return value is 1 on success and 0 on failure

4.2.5 CH347Uart_Init

Function description

This function is used to initialize serial port parameters

Function definitions

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

Parameter descriptions

iIndex:	Specify the serial number of the operating device.
BaudRate:	Baud rate value.
ByteSize:	Data bits（5, 6, 7, 8, 16）
Parity:	Parity bits（0: None; 1: Odd; 2: Even; 3: Mark; 4: Space）
StopBits:	Stop bits (0: stop bit; 1: .5 stop bit; 2: stop bit)
ByteTimeout:	Byte timeout time, the unit is 100uS.

Return value

The return value is 1 on success and 0 on failure

4.2.6 CH347Uart_SetTimeout

Function description

This function is used to set the timeout for USB data read/write

Function definitions

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

Parameter descriptions

iIndex:	Specify the serial number of the operating device
iWriteTimeout:	Specify the timeout for USB write-out data blocks, the unit is millisecond (mS) 0xFFFFFFFF specifies no timeout (default)
iReadTimeout:	Specify the timeout for USB read data blocks, the unit is milliseconds (mS) 0xFFFFFFFF specifies no timeout (default)

Return value

The return value is 1 on success and 0 on failure

4.2.7 CH347Uart_Read

Function description

This function is used to read serial port data

Function definitions

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

Parameter descriptions

iIndex:	Specify the serial number of the operating device

oBuffer: Point to a large enough buffer, place the data to be read.
ioLength: Point to the length unit. The input is the length to be read and the return is the actual read length.

Return value

The return value is 1 on success and 0 on failure

4.2.8 CH347Uart_Write

Function description

This function is used to send serial port data

Function definitions

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

Parameter descriptions

iIndex: Specify the serial number of the operating device
iBuffer: Point to a buffer, place the data to be written-out.
ioLength: Point to the length unit. The input is the length to be written-out, and the return is the actual length.

Return value

The return value is 1 on success and 0 on failure

4.2.9 CH347Uart_QueryBufUpload

Function description

This function is used to query how many bytes are unfetched in the buffer

Function definitions

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

Parameter descriptions

iIndex: Specify the serial number of the operating device
RemainBytes: Returns the number of unfetched bytes in the current buffer

Return value

The return value is 1 on success and 0 on failure

4.3 GPIO functions

4.3.1 Operation process

When operating GPIO, use CH347OpenDevice/CH347Uart_Open to open the device.

After using CH347GPIO_Get to get the current GPIO status, use CH347GPIO_Set to set the input and output status of GPIO as required.

You can call CH347GPIO_Get and CH347GPIO_Set to get and control GPIO.

The GPIO interrupt function can be realized by using CH347SetIntRoutine and CH347ReadInter, and the call to CH347AbortInter to give up the interrupt data read operation.

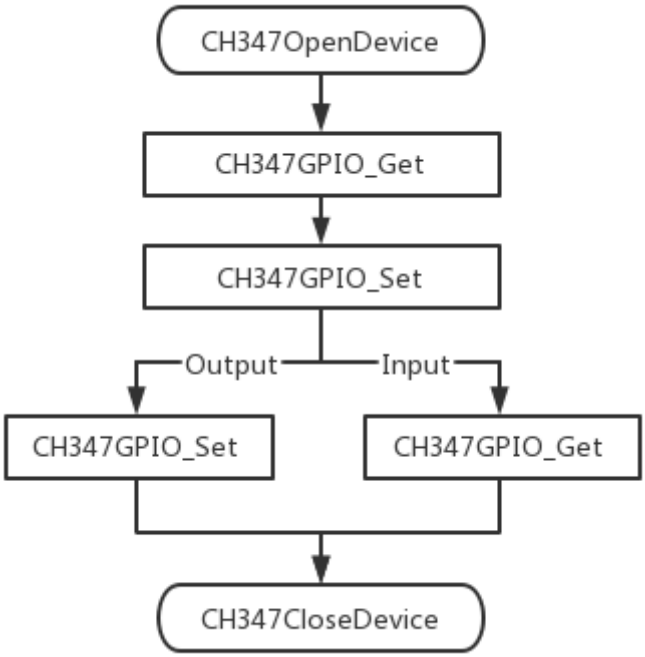


Figure 4.3.1 GPIO Operation flowchart

For details about the function, see the following.

4.3.2 CH347GPIO_Get

Function description

This function is used to get the current GPIO input/output status of the device

Function definitions

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

Parameter descriptions

iIndex:	Specify the serial number of the operating device
iDir:	Pin direction: GPIO 0-7 corresponding bit 0-7. 0: input; 1: output
iData:	GPIO level status: GPIO 0-7 corresponds to bits 0-7, where 0 indicates low level and 1 indicates high level

Return value

The return value is 1 on success and 0 on failure

4.3.3 CH347GPIO_Set

Function description

This function is used to set the I/O direction and output state of CH347-GPIO

Function definitions

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

Parameter descriptions

iIndex:	Specify the serial number of the operating device
iEnable:	Data validity flag: corresponding bit 0-7, corresponding to GPIO 0-7.
iSetDirOut:	Set the I/O direction, If a bit is 0, the corresponding pin is an input pin, if a bit is 1, the corresponding pin is an output pin. GPIO 0-7 corresponds to bits 0-7
iSetDataOut:	Output data. If the I/O direction is output, then the corresponding pin outputs low when a bit is 0 and high when it is 1.

Return value

The return value is 1 on success and 0 on failure

4.3.4 CH347SetIntRoutine

Function description

This function is used to set the CH347 - GPIO interrupt service program.

Function definitions

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

Parameter descriptions

iIndex：	Specify the serial number of the operating device
Int0PinN：	Interrupt GPIO pin number, greater than 7: do not enable this interrupt source; for 0-7 corresponds to GPIO 0-7
Int0TripMode：	Interrupt Type: 00: Falling edge trigger; 01: Rising edge trigger. 02: double edge trigger; 03: Reserved.
Int1PinN：	Interrupt GPIO pin number, greater than 7: do not enable this interrupt source; for 0-7 corresponds to GPIO 0-7
Int1TripMode：	Interrupt Type: 00: Falling edge trigger; 01: Rising edge trigger. 02: double edge trigger; 03: Reserved.
iIntRoutine：	Specify the interrupt service program, if it is NULL, the interrupt service will be canceled, otherwise, the program will be called at the time of interrupt.

Return value

The return value is 1 on success and 0 on failure

4.3.5 CH347ReadInter

Function description

This function is used to read interrupt data

Function definitions

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

Parameter descriptions

iIndex：	Specify the serial number of the operating device
iStatus ：	Points to the byte unit, used to save the read GPIO pin status data, refer

to the bit description below.

Return value

The return value is 1 on success and 0 on failure

4.3.6 CH347AbortInter

Function description

This function is used to cancel reading interrupt data

Function definitions

BOOL WINAPI

CH347AbortInter(ULONG iIndex)

Parameter descriptions

iIndex: Specify the serial number of the operating device

Return value

The return value is 1 on success and 0 on failure