CH34X Application Development Manual for Linux

1. CH346

1.1. Introduction

CH346 is a USB2.0 high speed converter chip to realize USB-UART, USB-SPI, USB-FIFO and USB-GPIO interfaces. CH346C supports 3 working modes, which need to be selected according to the application.

"libch347.so/libch347.a" is used to provide the CH346 chip with the parallel FIFO and SPI interface operation functions on the operating system side, and supports the vendor's driver interface, so there is no need to distinguish between the driver interface and the chip's operating mode when using it.

1.2. Interface Description

According to the characteristics of USB converter interface supported by CH346, "libch347" provides interface functional functions for parallel FIFO and SPI, including the basic functional function and the corresponding functional function.

The interfaces supported by the CH346C are shown in the table below. Different operating modes can be switched by configuring the pin level status via MODE at power-up or by configuring the EEPROM.

Working Mode	Functional Interface Description	Driver Interface	API
Mode 0	Interface 0: USB2.0 to High-speed UART0	CH343SER(VCP)	Native termios API in the system
	Interface 1: USB2.0 to Passive Parallel FIFO	CH341PAR	in ch346_lib.h
Mode 1	Interface 0: USB2.0 to High-speed UART0	CH343SER(VCP)	Native termios API in the system
	Interface 1: USB2.0 to Passive SPI interface	CH341PAR	in ch346_lib.h
Mode 2	Interface 0: USB2.0 to High-speed UART0	CH343SER(VCP)	Native termios API in the system
	Interface 1:USB2.0 to High-speed UART1		

Table. CH346 Interface function API

For details about the function, see the following.

1.3. Public Operation Functions

1.3.1. CH346GetLibInfo

Function description	This function is used to get library information.
Function definition	const char *CH346GetLibInfo(void);

Parameter description	null
Return value	Library information string.

1.3.2. CH346OpenDevice

Function description	This function is used to open the CH346 device, supports opening in CH346 FIFO/SPI interface mode.
Function definition	int CH346OpenDevice(const char *pathname);
Parameter description	pathname:device path in /dev directory
Return value	The function return positive file descriptor if successful, others if fail.

1.3.3. CH346CloseDevice

Function description	This function is used to close the CH346 device, supports closing in CH346 FIFO/SPI interface mode.
Function definition	bool CH346CloseDevice(int fd);
Parameter description	fd: file descriptor of device
Return value	The function return true if successful, false if fail.

1.3.4. CH34xSetTimeout

Function description	This function is used to set USB data read and write timeout.
Function definition	bool CH34xSetTimeout(int fd, uint32_t iWriteTimeout, uint32_t iReadTimeout);
Parameter description	fd: file descriptor of device iWriteTimeout: data download timeout in milliseconds iReadTimeout: data upload timeout in milliseconds
Return value	The function return true if successful, false if fail.

1.3.5. CH34x_GetDriverVersion

Function description	This function is used to get vendor driver version.
Function definition	bool CH34x_GetDriverVersion(int fd, unsigned char *Drv_Version);
Parameter description	fd: file descriptor of device Drv_Version: pointer to version string
Return value	The function return true if successful, false if fail.

1.3.6. CH34x_GetChipType

Function description	This function is used to get chip type.
Function definition	bool CH34x_GetChipType(int fd, CHIP_TYPE *ChipType);
Parameter description	fd: file descriptor of device ChipType: pointer to chip type
Return value	The function return true if successful, false if fail.

1.3.7. CH34x_GetDeviceID

Function description	This function is used to get device VID and PID.
Function definition	fd: file descriptor of device id: pointer to store id which contains VID and PID
Parameter description	bool CH34x_GetDeviceID(int fd, uint32_t *id);
Return value	The function return true if successful, false if fail.

1.4. FIFO/SPI Interface

1.4.1. Operation Process

After the device is enabled, set the device USB read and write timeout parameters, then configure the interface mode (mode 0: parallel FIFO, mode 1: SPI interface)， After successful setup, you can communicate with the device by calling the FIFO/SPI read/write function.

The function call flowchart is as follows:

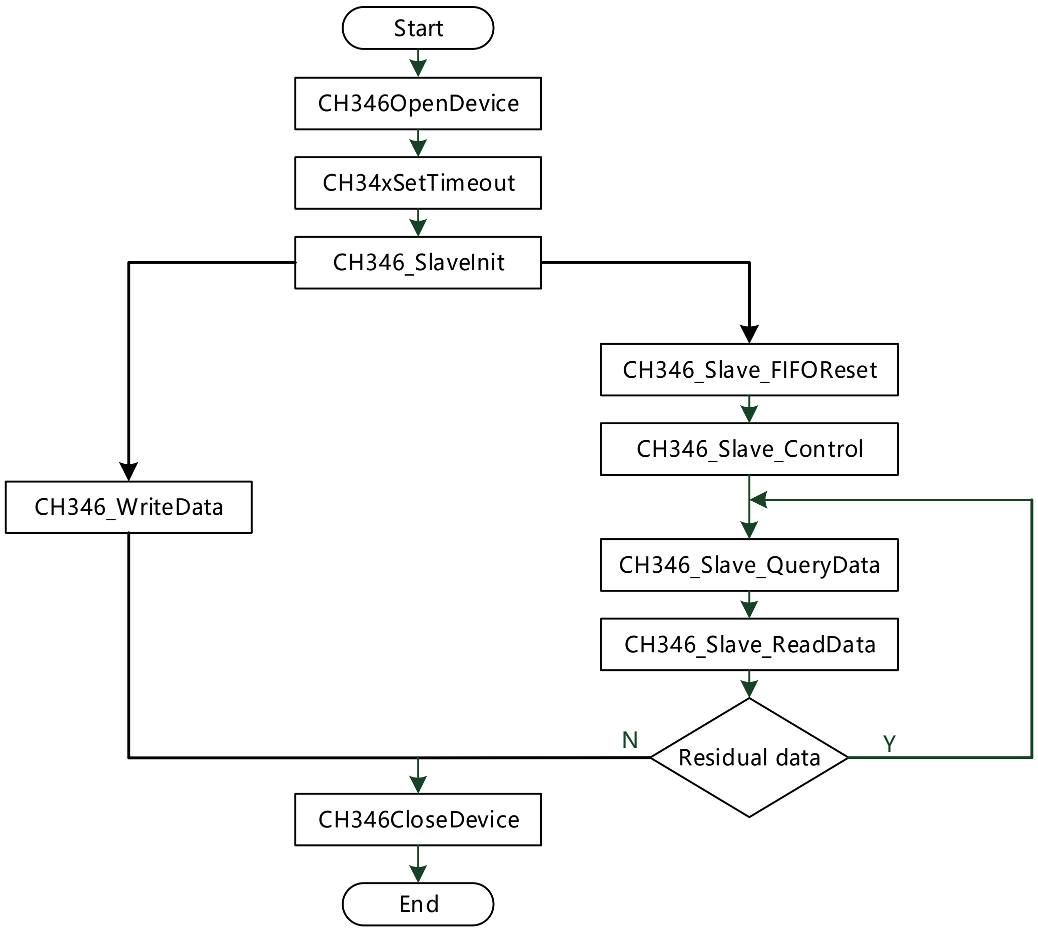

Figure 1.1 FIFO/SPI function operation flowchart

For details about the function, see the following.

1.4.2. CH346_SetMode

Function description	This function is used to set chip work mode.
Function definition	bool CH346_SetMode(int fd, uint8_t Mode);
Parameter description	fd: file descriptor of device Mode: work mode, 0->mode0, 1->mode1
Return value	The function return true if successful, false if fail.

1.4.3. CH346_GetMode

Function description	This function is used to get chip work mode.
Function definition	bool CH346_GetMode(int fd, uint8_t *Mode);
Parameter description	fd: file descriptor of device Mode: pointer to work mode, 0->mode0, 1->mode1
Return value	The function return true if successful, false if fail.

1.4.4. CH346_SlaveInit

Function description	This function is used to initialize the CH346 FIFO/SPI interface.
Function definition	bool CH346_SlaveInit(int fd);
Parameter description	fd: file descriptor of device
Return value	The function return true if successful, false if fail.

1.4.5. CH346_Slave_Control

Function description	This function is used to switch of reading FIFO data from master.
Function definition	bool CH346_Slave_Control(int fd, bool enable);
Parameter description	fd: file descriptor of device enable: true: start reading continuously, false: stop reading
Return value	The function return true if successful, false if fail.

1.4.6. CH346_Slave_WriteData

Function description	This function is used to write FIFO/SPI data.
Function definition	bool CH346_Slave_WriteData(int fd, void *iBuffer, uint32_t *ioLength);
Parameter description	fd: file descriptor of device iBuffer: pointer to write buffer ioLength: pointer to write length
Return value	The function return true if successful, false if fail.

1.4.7. CH346_Slave_QueryData

Function description	This function is used to get FIFO/SPI data length.
Function definition	bool CH346_Slave_QueryData(int fd, uint32_t *oLength);
Parameter description	fd: file descriptor of device oLength: pointer to read length
Return value	The function return true if successful, false if fail.

1.4.8. CH346_Slave_FIFOReset

Function description	This function is used to reset FIFO/SPI data fifo.
Function definition	bool CH346_Slave_FIFOReset(int fd);
Parameter description	fd: file descriptor of device

Return value	The function return true if successful, false if fail.

1.4.9. CH346_Slave_ReadData

Function description	This function is used to read FIFO/SPI data in slave mode.
Function definition	bool CH346_Slave_ReadData(int fd, void *oReadBuffer, uint32_t *oReadLength);
Parameter description	fd: file descriptor of device oReadBuffer: pointer to read buffer oReadLength: pointer to read length
Return value	The function return true if successful, false if fail.

2. CH347

2.1 Introduction

CH347 is a USB2.0 high-speed converter chip to realize 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.

“libch347.so/libch347.a” is used to provide UART/SPI/I2C/JTAG/GPIO interface operation functions for CH347/CH339W chip on the OS side, there is no need to distinguish between driver interface and chip working mode when using it.

2.2 Interface Description

According to the characteristics of USB converter interface supported by CH347, “libch347” 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 termios API in the system or CH347UART_xxx in ch347_lib.h
Interface1: USB2.0 to high speed UART1		
Interface2: USB2.0 to JTAG + SPI + I2C, etc.	CH341PAR	CH347SPI_xxx, CH347I2C_xxx, CH347JTAG_xxx in ch347_lib.h

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 termios API in the system or CH347UART_xxx in ch347_lib.h
	Interface 1: USB2.0 to High-speed UART1		
Mode 1	Interface 0: USB2.0 to High-speed UART1	CH343SER(VCP)	Native termios API in the system or CH347UART_xxx in ch347_lib.h
	Interface 1: USB2.0 to SPI+I2C	CH341PAR	CH347SPI_xxx, CH347I2C_xxx in ch347_lib.h
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 ch347_lib.h
Mode 3	Interface 0: USB2.0 to High-speed UART1	CH343SER(VCP)	Native termios API in the system or CH347UART_xxx in ch347_lib.h
	Interface 1: USB2.0 to JTAG+I2C	CH341PAR	CH347JTAG_xxx in the ch347_lib.h or CH347I2C_xxx

Table. CH347 Interface function API

2.3 Public Functions

2.3.1. CH347GetLibInfo

Function description	This function is used to get library information.
Function definition	const char *CH347GetLibInfo(void);
Parameter description	null
Return value	Library information string.

2.3.2. CH347OpenDevice

Function description	This function is used to open device.
Function definition	int CH347OpenDevice(const char *pathname);
Parameter description	pathname: device path in /dev directory

Return value	The function return positive file descriptor if successful, others if fail.

2.3.3. CH347CloseDevice

Function description	This function is used to close device.
Function definition	bool CH347CloseDevice(int fd);
Parameter description	fd: file descriptor of device
Return value	The function return true if successful, false if fail.

2.3.4. CH34xSetTimeout

Function description	This function is used to set USB data read and write timeout.
Function definition	bool CH34xSetTimeout(int fd, uint32_t iWriteTimeout, uint32_t iReadTimeout);
Parameter description	fd: file descriptor of device iWriteTimeout: data download timeout in milliseconds iReadTimeout: data upload timeout in milliseconds
Return value	The function return true if successful, false if fail.

2.3.5. CH34x_GetDriverVersion

Function description	This function is used to get vendor driver version.
Function definition	bool CH34x_GetDriverVersion(int fd, unsigned char *Drv_Version);
Parameter description	fd: file descriptor of device Drv_Version: pointer to version string
Return value	The function return true if successful, false if fail.

2.3.6. CH34x_GetChipType

Function description	This function is used to get chip type.
Function definition	bool CH34x_GetChipType(int fd, CHIP_TYPE *ChipType);
Parameter description	fd: file descriptor of device ChipType: pointer to chip type
Return value	The function return true if successful, false if fail.

2.3.7. CH34x_GetDeviceID

Function description	This function is used to get device VID and PID.

Function definition	bool CH34x_GetDeviceID(int fd, uint32_t *id);
Parameter description	fd: file descriptor of device id: pointer to store id which contains VID and PID
Return value	The function return true if successful, false if fail.

2.3.8. CH347_OE_Enable

Function description	This function is used to CH347F chip OE function switch.
Function definition	bool CH347_OE_Enable(int fd);
Parameter description	fd: file descriptor of device
Return value	The function return true if successful, false if fail.

2.4. SPI Functions

2.4.1. 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:

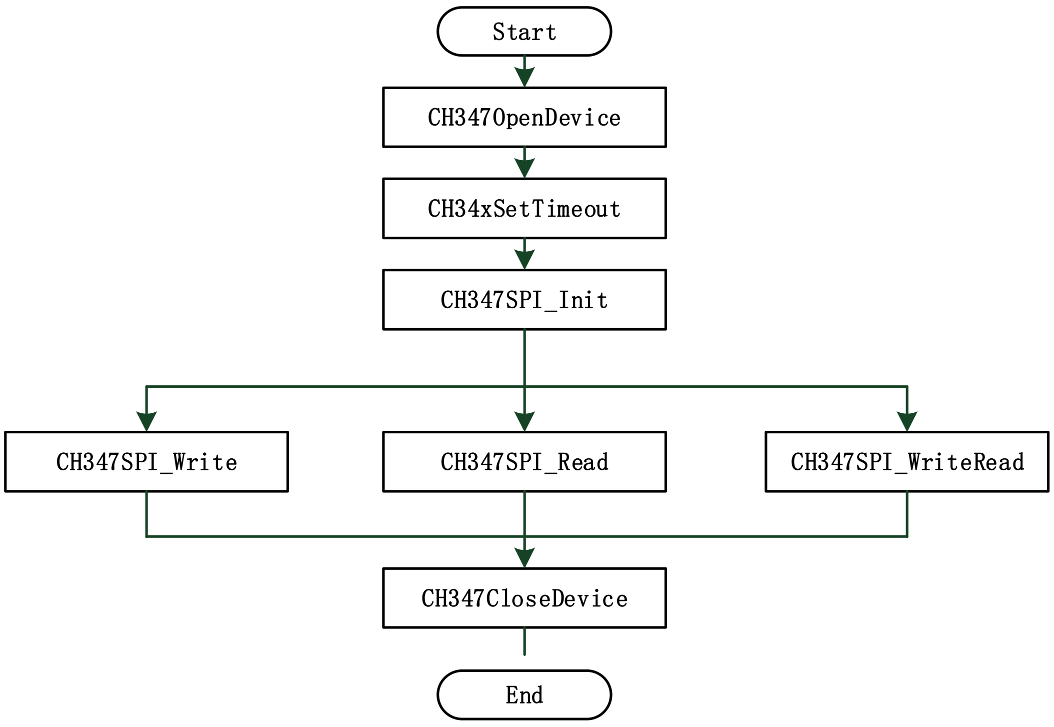

Figure 2.1 SPI Function operation flowchart

2.4.2. CH347SPI_GetHwStreamCfg

Function description	This function is used to get SPI setting from hardware.
Function definition	bool CH347SPI_GetHwStreamCfg(int fd, StreamHwCfgS *StreamCfg);
Parameter description	fd: file descriptor of device StreamCfg: pointer to SPI stream configuration
Return value	The function return true if successful, false if fail.

2.4.3. CH347SPI_SetFrequency

Function description	This function is used to SPI frequency setting.
Function definition	bool CH347SPI_SetFrequency(int fd, uint32_t iSpiSpeedHz);
Parameter description	fd: file descriptor of device iSpiSpeedHz: SPI frequency value, 60e6/30e6/15e6/75e5/375e4/1875e3/9375e2/46875e1
Return value	The function return true if successful, false if fail.

2.4.4. CH347SPI_SetAutoCS

Function description	This function is used to SPI auto chipselect setting for CH347SPI_WriteRead.
Function definition	bool CH347SPI_SetAutoCS(int fd, bool disable);
Parameter description	fd: file descriptor of device disable: SPI auto chipselect setting switch, true on disable CS automatic control
Return value	The function return true if successful, false if fail.

2.4.5. CH347SPI_SetDataBits

Function description	This function is used to SPI data bits setting.
Function definition	bool CH347SPI_SetDataBits(int fd, uint8_t iDataBits);
Parameter description	fd: file descriptor of device iDataBits: 0: 8bit, 1: 16bit
Return value	The function return true if successful, false if fail.

2.4.6. CH347SPI_Init

Function description	This function is used to SPI interface initialization.
Function definition	bool CH347SPI_Init(int fd, mSpiCfgS *SpiCfg);

Parameter description	fd: file descriptor of device SpiCfg: pointer to SPI configuration, SPI frequency could be set by SpiCfg->iClock or CH347SPI_SetFrequency API, the latter is preferred.
Return value	The function return true if successful, false if fail.

2.4.7. CH347SPI_GetCfg

Function description	This function is used to get SPI configuration.
Function definition	bool CH347SPI_GetCfg(int fd, mSpiCfgS *SpiCfg);
Parameter description	fd: file descriptor of device SpiCfg: pointer to SPI configuration
Return value	The function return true if successful, false if fail.

2.4.8. CH347SPI_ChangeCS

Function description	This function is used to SPI CS setting, must call CH347SPI_Init first.
Function definition	bool CH347SPI_ChangeCS(int fd, uint8_t iStatus);
Parameter description	fd: file descriptor of device iStatus: 0:cancel CS, 1: set CS
Return value	The function return true if successful, false if fail.

2.4.9. CH347SPI_Write

Function description	This function is used to write SPI data.
Function definition	bool CH347SPI_Write(int fd, bool ignoreCS uint8_t iChipSelect, int iLength, int iWriteStep, void *ioBuffer);
Parameter description	fd: file descriptor of device ignoreCS: ignore SPI chip select while true, else auto set CS iChipSelect: SPI chip select, ignore while BIT7 is 0, valid while BIT7 is 1 iLength: length to write iWriteStep: per write length ioBuffer: pointer to write buffer
Return value	The function return true if successful, false if fail.

2.4.10. CH347SPI_Read

Function description	This function is used to read SPI data.

Function definition	bool CH347SPI_Read(int fd, bool ignoreCS, uint8_t iChipSelect, int iLength, uint32_t *oLength, void *ioBuffer);
Parameter description	fd: file descriptor of device ignoreCS: ignore SPI chip select while true, else auto set CS iChipSelect: SPI chip select, ignore while BIT7 is 0, valid while BIT7 is 1 iLength: length to write oLength: pointer to read length ioBuffer: pointer to buffer, store data to be written from MOSI, and return data to be read from MISO
Return value	The function return true if successful, false if fail.

2.4.11. CH347SPI_WriteRead

Function description	This function is used to write then read SPI data.
Function definition	bool CH347SPI_WriteRead(int fd, bool ignoreCS, uint8_t iChipSelect, int iLength, void *ioBuffer);
Parameter description	fd: file descriptor of device ignoreCS: ignore SPI chip select while true, else auto set CS iChipSelect: SPI chip select, ignore while BIT7 is 0, valid while BIT7 is 1 iLength: data length to xfer ioBuffer: pointer to buffer, store data to be written from MOSI, and return data to be read from MISO
Return value	The function return true if successful, false if fail.

2.5. Jtag Functions

2.5.1. 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:

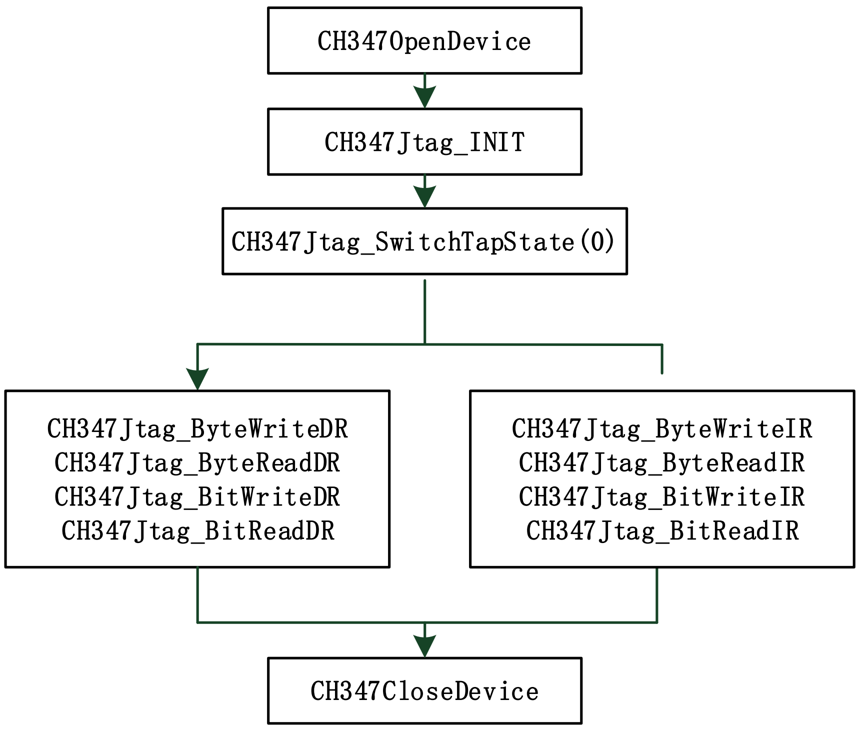

Figure 2.2 JTAG Function operation flowchart

For details about the function, see the following.

2.5.2. CH347Jtag_Reset

Function description	This function is used to Reset Tap Status, more than six consecutive TCK and TMS is high will set the state machine to the Test-Logic Reset state.
Function definition	int CH347Jtag_Reset(int fd);
Parameter description	fd: file descriptor of device
Return value	The function return true if successful, false if fail.

2.5.3. CH347Jtag_ResetTrst

Function description	This function is used to Hard-reset JTAG device.
Function definition	bool CH347Jtag_ResetTrst(int fd, bool TRSTLevel);
Parameter description	fd: file descriptor of device TRSTLevel: reset level, true on high, false on low
Return value	The function return true if successful, false if fail.

2.5.4. CH347Jtag_INIT

Function description	This function is used to JTAG interface initialization, mode and speed setting.

Function definition	CH347Jtag_INIT(int fd, uint8_t iClockRate);
Parameter description	fd: file descriptor of device
Return value	The function return true if successful, false if fail.

2.5.5. CH347Jtag_GetCfg

Function description	This function is used to get JTAG speed setting.
Function definition	bool CH347Jtag_GetCfg(int fd, uint8_t *ClockRate);
Parameter description	fd: file descriptor of device iClockRate: pointer to communication speed, valid value is 0-5, the higher the value, the faster the speed
Return value	The function return true if successful, false if fail.

2.5.6. CH347Jtag_ClockTms

Function description	This function is used to change TMS value on the rising edge of TCK to switch its Tap state.
Function definition	uint32_t CH347Jtag_ClockTms(uint8_t *BitBangPkt, uint32_t Tms, uint32_t BI);
Parameter description	BitBangPkt: protocol package Tms: TMS value to change BI: length of protocol package
Return value	The function return length of protocol package.

2.5.7. CH347Jtag_IdleClock

Function description	This function is used to ensure the clock is in low status.
Function definition	uint32_t CH347Jtag_IdleClock(uint8_t *BitBangPkt, uint32_t BI);
Parameter description	BitBangPkt: protocol package BI: length of protocol package
Return value	The function return true if successful, false if fail.

2.5.8. CH347Jtag_TmsChange

Function description	This function is used to change TMS value to switch state.
Function definition	bool CH347Jtag_TmsChange(int fd, uint8_t *tmsValue, uint32_t Step, uint32_t Skip);
Parameter description	fd: file descriptor of device tmsValue: pointer to TMS value, unit: byte Step: The valid bits which stored in tmsValue

	Skip: valid start bit
Return value	The function return true if successful, false if fail.

2.5.9. CH347Jtag_IoScan

Function description	This function is used to read and write in the Shift-DR/IR state, and switch to Exit DR/IR after execution State machine change: Shift-DR/IR.RW.->Exit DR/IR
Function definition	bool CH347Jtag_IoScanT(int fd, uint8_t *DataBits, uint32_t DataBitsNb, bool IsRead, bool IsLastPkt);
Parameter description	fd: file descriptor of device DataBits: data bits to be transmitted DataBitsNb: number of bits to be transmitted IsRead: whether to read data
Return value	The function return true if successful, false if fail.

2.5.10. CH347Jtag_IoScanT

Function description	This function is used to read and write in the Shift-DR/IR state, if it is the last package, switch to Exit DR/IR; if not, stop at Shift-DR/IR.
Function definition	bool CH347Jtag_IoScanT(int fd, uint8_t *DataBits, uint32_t DataBitsNb, bool IsRead, bool IsLastPkt);
Parameter description	fd: file descriptor of device DataBits: data bits to be transmitted DataBitsNb: number of bits to be transmitted IsRead: whether to read data IsLastPkt: whether the last package
Return value	The function return true if successful, false if fail.

2.5.11. CH347Jtag_WriteRead

Function description	This function is used to bitband mode JTAG IR/DR data read and write which is applicable for a small amount of data. Exp: command operation, state machine switching and other control transmission. For batch data transmission, it is recommended to use CH347Jtag_ WriteRead_Fast. Command packets are read and written in batches in 4096 bytes. State machine: Run-Test -> Shift-IR/DR.. -> exit IR/DR -> Run-Test
Function definition	bool CH347Jtag_WriteRead(int fd, bool IsDR,

	int iWriteBitLength, void *iWriteBitBuffer, uint32_t *oReadBitLength, void *oReadBitBuffer);
Parameter description	fd: file descriptor of device IsDR: true: DR data read and write, false: IR data read and write iWriteBitLength: write length iWriteBitBuffer: pointer to write buffer iReadTimes: read times oReadBitLength: pointer to read length, returns the actual number of bytes read on success oReadBitBuffer: pointer to read buffer
Return value	The function return true if successful, false if fail.

2.5.12. CH347Jtag_WriteRead_Fast

Function description	This function is used to JTAG IR/DR data read and write in batches for multi-byte continuous operation. Exp: JTAG firmware download operation. Hardware has a 4K buffer, such as write then read, the length should not exceed 4096 bytes. The buffer size can be adjusted. State machine: Run-Test -> Shift-IR/DR.. -> exit IR/DR -> Run-Test
Function definition	bool CH347Jtag_WriteRead_Fast(int fd, bool IsDR, int iWriteLength, void *iWriteBuffer, uint32_t *oReadLength, void *oReadBuffer);
Parameter description	fd: file descriptor of device IsDR: true: DR data read and write, false: IR data read and write iWriteBitLength: write length iWriteBitBuffer: pointer to write buffer iReadTimes: read times oReadBitLength: pointer to read length, returns the actual number of bytes read on success oReadBitBuffer: pointer to read buffer
Return value	The function return true if successful, false if fail.

2.5.13. CH347Jtag_SwitchTapState

Function description	This function is used to switch JTAG state machine.
Function definition	bool CH347Jtag_SwitchTapState(int fd, uint8_t TapState);
Parameter description	fd: file descriptor of device TapState: machine state
Return value	The function return true if successful, false if fail.

2.5.14. CH347Jtag_ByteWriteDR

Function description	This function is used to JTAG DR write in bytes which used for multi-byte continuous operation. Exp: JTAG firmware download operation. State machine: Run-Test -> Shift-DR.. -> exit DR -> Run-Test
Function definition	bool CH347Jtag_ByteWriteDR(int fd, int iWriteLength, *iWriteBuffer);
Parameter description	fd: file descriptor of device oReadLength: pointer to read length, returns the actual number of bytes read on success oReadBuffer: pointer to read buffer
Return value	The function return true if successful, false if fail.

2.5.15. CH347Jtag_ByteWriteIR

Function description	This function is used to JTAG IR write in bytes which used for multi-byte continuous operation. State machine: Run-Test -> Shift-IR.. -> exit IR -> Run-Test
Function definition	bool CH347Jtag_ByteWriteIR(int fd, int iWriteLength, void *iWriteBuffer);
Parameter description	fd: file descriptor of device iWriteLength: pointer to write length iWriteBuffer: pointer to write buffer
Return value	The function return true if successful, false if fail.

2.5.16. CH347Jtag_ByteReadIR

Function description	This function is used to JTAG IR read in bytes which used for multi-byte continuous operation. State machine: Run-Test -> Shift-IR.. -> exit IR -> Run-Test
Function definition	bool CH347Jtag_ByteReadIR(int fd, uint32_t *oReadLength, void *oReadBuffer);
Parameter description	fd: file descriptor of device oReadLength: pointer to read length, returns the actual number of bytes read on success oReadBuffer: pointer to read buffer
Return value	The function return true if successful, false if fail.

2.5.17. CH347Jtag_ByteReadDR

Function description	This function is used to JTAG DR read in bytes which used for multi-byte continuous operation. State machine: Run-Test -> Shift-DR.. -> exit DR -> Run-Test
Function definition	bool CH347Jtag_BitWriteDR(int fd, int iWriteBitLength, void *iWriteBitBuffer);

Parameter description	fd: file descriptor of device oReadLength: pointer to read length, returns the actual number of bytes read on success oReadBuffer: pointer to read buffer
Return value	The function return true if successful, false if fail.

2.5.18. CH347Jtag_BitWriteDR

Function description	This function is used to bitband mode JTAG DR data write which is applicable for a small amount of data. Exp: command operation, state machine switching and other control transmission. For batch data transmission, it is recommended to use USB20Jtag_ByeWriteDR. State machine: Run-Test -> Shift-DR.. -> exit DR -> Run-Test
Function definition	bool CH347Jtag_BitWriteDR(int fd, int iWriteBitLength, void *iWriteBitBuffer);
Parameter description	fd: file descriptor of device iWriteBitLength: pointer to write length iWriteBitBuffer: pointer to write buffer
Return value	The function return true if successful, false if fail.

2.5.19. CH347Jtag_BitWriteIR

Function description	This function is used to bitband mode JTAG IR data write which is applicable for a small amount of data. Exp: command operation, state machine switching and other control transmission. For batch data transmission, it is recommended to use USB20Jtag_ByeWriteIR. State machine: Run-Test -> Shift-IR.. -> exit IR -> Run-Test
Function definition	bool CH347Jtag_BitWriteIR(int fd, int iWriteBitLength, void *iWriteBitBuffer);
Parameter description	fd: file descriptor of device iWriteBitLength: pointer to write length iWriteBitBuffer: pointer to write buffer
Return value	The function return true if successful, false if fail.

2.5.20. CH347Jtag_BitReadIR

Function description	This function is used to bitband mode JTAG IR data read which is applicable for a small amount of data. Exp: command operation, state machine switching and other control transmission. For batch data transmission, it is recommended to use USB20Jtag_ByteReadIR. State machine: Run-Test -> Shift-IR.. -> exit IR -> Run-Test

Function definition	bool CH347Jtag_BitReadIR(int fd, uint32_t *oReadBitLength, void *oReadBitBuffer);
Parameter description	fd: file descriptor of device oReadBitLength: pointer to read length, returns the actual number of bytes read on success oReadBitBuffer: pointer to read buffer
Return value	The function return true if successful, false if fail.

2.5.21. CH347Jtag_BitReadDR

Function description	This function is used to bitband mode JTAG DR data read which is applicable for a small amount of data. Exp: command operation, state machine switching and other control transmission. For batch data transmission, it is recommended to use USB20Jtag_ByteReadDR. State machine: Run-Test -> Shift-DR.. -> exit DR -> Run-Test
Function definition	bool CH347Jtag_BitReadDR(int fd, uint32_t *oReadBitLength, void *oReadBitBuffer);
Parameter description	fd: file descriptor of device oReadBitLength: pointer to read length, returns the actual number of bytes read on success oReadBitBuffer: pointer to read buffer
Return value	The function return true if successful, false if fail.

2.6. GPIO Functions

2.6.1. 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.

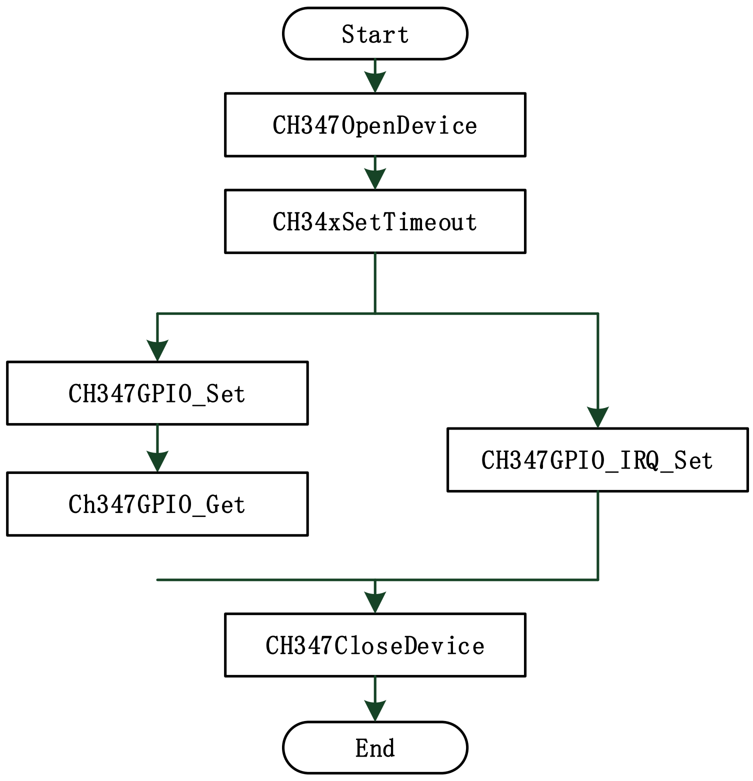

Figure 2.3 GPIO Operation flowchart

For details about the function, see the following.

2.6.2. CH347GPIO_Get

Function description	This function is used to get GPIO status.
Function definition	bool CH347GPIO_Get(int fd, uint8_t *iDir, uint8_t *iData);
Parameter description	fd: file descriptor of device iDir: gpio direction bits, bits0-7 on gpio0-7, 1 on ouput, 0 on input iData: gpio level bits, bits0-7 on gpio0-7, 1 on high, 0 on low
Return value	The function return true if successful, false if fail.

2.6.3. CH347GPIO_Set

Function description	This function is used to GPIO setting.
Function definition	bool CH347GPIO_Set(int fd, uint8_t iEnable, uint8_t iSetDirOut, uint8_t iSetDataOut);
Parameter description	fd: file descriptor of device iEnable: gpio function enable bits, bits0-7 on gpio0-7, 1 on enable iSetDirOut: gpio direction bits, bits0-7 on gpio0-7, 1 on ouput, 0 on input iSetDataOut: gpio output bits, bits0-7 on gpio0-7, if gpio

	direction is output, 1 on high, 0 on low
Return value	The function return true if successful, false if fail.

2.6.4. CH347GPIO_IRQ_Set

Function description	This function is used to GPIO interrupt function setting.
Function definition	bool CH347GPIO_IRQ_Set(int fd, uint8_t gpioindex, bool enable, uint8_t irqtype, void *isr_handler);
Parameter description	fd: file descriptor of device gpioindex: 0, 2, 3, 4, 5, 6 and 7 are valid enable: 0 : disable, 1 : enable irqtype: IRQ_TYPE_EDGE_FALLING, IRQ_TYPE_EDGE_RISING, IRQ_TYPE_EDGE_BOTH isr_handler: handler to call when interrupt occurs, if isr disable, the routine will be ignored.
Return value	The function return true if successful, false if fail.

2.7. HID/VCP UART Functions

2.7.1. 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.

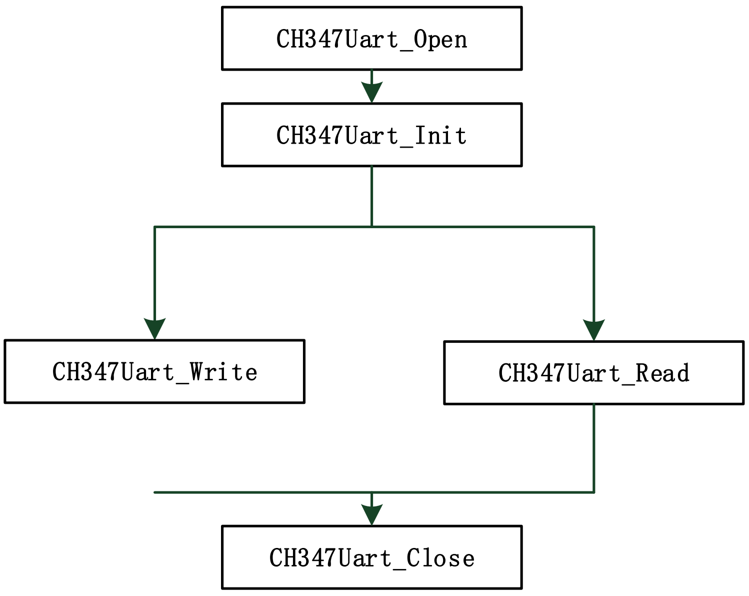

Figure 2.5 HID Serial port operation flowchart

For details about the function, see the following.

2.7.2. CH347Uart_Open

Function description	This function is used to open tty device.
Function definition	int CH347Uart_Open(const char *pathname);
Parameter description	pathname: device path in /dev directory
Return value	The function return positive file descriptor if successful, others if fail.

2.7.3. CH347Uart_Close

Function description	This function is used to close tty device.
Function definition	bool CH347Uart_Close(int fd);
Parameter description	fd: file descriptor of device
Return value	The function return true if successful, false if fail.

2.7.4. CH347Uart_GetCfg

Function description	This function is used to read UART setting.
Function definition	bool CH347Uart_GetCfg(int fd, uint32_t *BaudRate, uint8_t *ByteSize, uint8_t *Parity, uint8_t *StopBits, uint8_t *ByteTimeout);
Parameter description	fd: file descriptor of device BaudRate: pointer to uart baudrate ByteSize: pointer to data bits --> 0 : 5 bits --> 1 : 6 bits --> 2 : 7 bits --> 3 : 8 bits --> 4 : 16 bits Parity: pointer parity --> 0 : none --> 1 : odd --> 2 : even --> 3 : mark --> 4 : space StopBits: pointer to stop bits --> 0 : 1 bit --> 1 : 1.5 bits --> 2 : 2 bits ByteTimeout: pointer to receive timeout value, unit: 100us
Return value	The function return true if successful, false if fail.

2.7.5. CH347Uart_Init

Function description	This function is used to initial UART.
Function definition	bool CH347Uart_Init(int fd, int BaudRate, uint8_t ByteSize, uint8_t Parity, uint8_t StopBits, uint8_t ByteTimeout);
Parameter description	fd: file descriptor of device BaudRate: uart baudrate ByteSize: data bits setting --> 0 : 5 bits --> 1 : 6 bits --> 2 : 7 bits --> 3 : 8 bits --> 4 : 16 bits Parity: parity setting --> 0 : none --> 1 : odd --> 2 : even --> 3 : mark --> 4 : space StopBits: stop bits setting --> 0 : 1 bit --> 1 : 1.5 bits --> 2 : 2 bits ByteTimeout: receive timeout value, unit: 100us
Return value	The function return true if successful, false if fail.

2.7.6. CH347Uart_Read

Function description	This function is used to read for UART operation.
Function definition	bool CH347Uart_Read(int fd, void *oBuffer, uint32_t *ioLength);
Parameter description	fd: file descriptor of device oBuffer: pointer to read buffer ioLength: pointer to read length
Return value	The function return true if successful, false if fail.

2.7.7. CH347Uart_Write

Function description	This function is used to write data for UART operation.
Function definition	bool CH347Uart_Write(int fd, void *iBuffer, uint32_t *ioLength);
Parameter description	fd: file descriptor of device iBuffer: pointer to write buffer ioLength: pointer to write length

Return value	The function return true if successful, false if fail.

2.8. I2C Functions

2.8.1. 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:

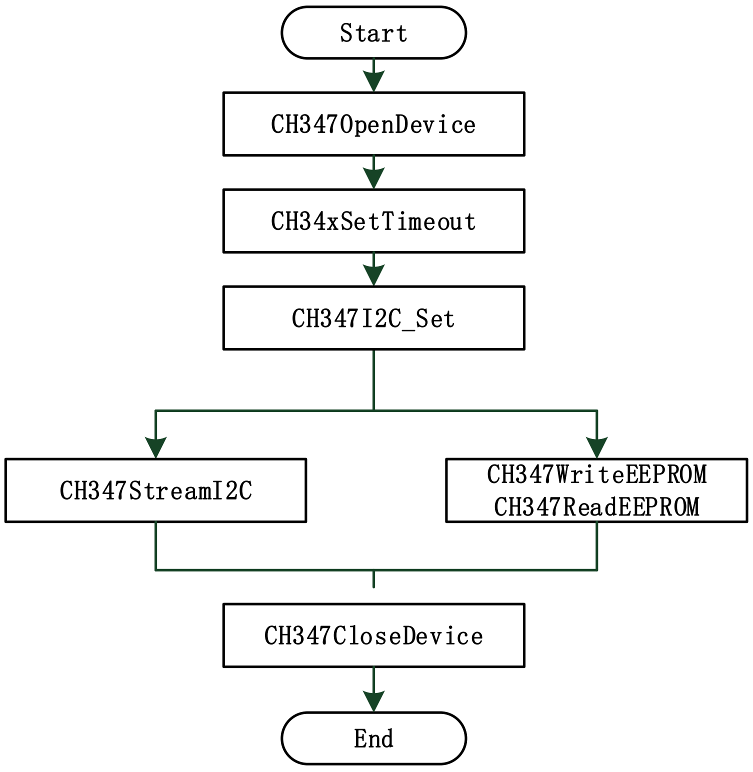

Figure 2.6 I2C operation flowchart

For details about the function, see the following.

2.8.2. CH347I2C_Set

Function description	This function is used to configure I2C interface in stream mode.
Function definition	bool CH347I2C_Set(int fd, int iMode);
Parameter description	fd: file descriptor of device iMode: stream mode ->bit0~2: set I2C SCL rate --> 000 : low rate 20KHz --> 001 : standard rate 100KHz --> 010 : fast rate 400KHz --> 011 : high rate 750KHz --> 100 : rate 50KHz --> 101 : standard rate 200KHz

	--> 110 : fast rate 1MHz other bits must keep 0
Return value	The function return true if successful, false if fail.

2.8.3. CH347I2C_SetStretch

Function description	This function is used to CH347T chip I2C Clock Stretch function control.
Function definition	bool CH347I2C_SetStretch(int fd, bool enable);
Parameter description	fd: file descriptor of device enable: I2C Clock Stretch enable, 1 : enable, 0 : disable
Return value	The function return true if successful, false if fail.

2.8.4. CH347I2C_SetDriveMode

Function description	This function is used to CH347T chip I2C signal drive mode control.
Function definition	bool CH347I2C_SetDriveMode(int fd, uint8_t mode);
Parameter description	fd: file descriptor of device mode: 0: open-drain, 1: push-pull
Return value	The function return true if successful, false if fail.

2.8.5. CH347I2C_SetIgnoreNack

Function description	This function is used to control CH347T chip I2C signal whether continues when it detects NACK.
Function definition	bool CH347I2C_SetIgnoreNack(int fd, uint8_t mode);
Parameter description	fd: file descriptor of device mode: 0: stop, 1: continue
Return value	The function return true if successful, false if fail.

2.8.6. CH347I2C_SetDelaymS

Function description	This function is used to I2C delay setting.
Function definition	bool CH347I2C_SetDelaymS(int fd, int iDelay);
Parameter description	fd: file descriptor of device iDelay: delay time in millseconds, 0~500 valid
Return value	The function return true if successful, false if fail.

2.8.7. CH347I2C_SetAckClk_DelayuS

Function description	This function is used to setting delay time between the 8th and 9th I2C clock.
Function definition	bool CH347I2C_SetAckClk_DelayuS(int fd, int iDelay);
Parameter description	fd: file descriptor of device iDelay: delay time in microseconds, max: 0x3ff
Return value	The function return true if successful, false if fail.

2.8.8. CH347StreamI2C

Function description	This function is used to write/read I2C in stream mode.
Function definition	bool CH347StreamI2C(int fd, int iWriteLength, void *iWriteBuffer, int iReadLength, void *oReadBuffer);
Parameter description	fd: file descriptor of device iWriteLength: write length iWriteBuffer: pointer to write buffer iReadLength: read length, 0: write data only oReadBuffer: pointer to read buffer
Return value	The function return true if successful, false if fail.

2.8.9. CH347StreamI2C_RetAck

Function description	This function is used to write/read I2C in stream mode.
Function definition	bool CH347StreamI2C_RetAck(int fd, int iWriteLength, void *iWriteBuffer, int iReadLength, void *oReadBuffer, int *retAck);
Parameter description	fd: file descriptor of device iWriteLength: write length iWriteBuffer: pointer to write buffer iReadLength: read length oReadBuffer: pointer to read buffer retAck: pointer to available ack count
Return value	The function return true if successful, false if fail.

2.8.10. CH347ReadEEPROM

Function description	This function is used to read data from EEPROM.
Function definition	bool CH347ReadEEPROM(int fd, EEPROM_TYPE iEepromID, int iAddr, int iLength, uint8_t *oBuffer);

Parameter description	fd: file descriptor of device iEepromID: EEPROM type iAddr: address of EEPROM iLength: read length oBuffer: pointer to read buffer
Return value	The function return true if successful, false if fail.

2.8.11. CH347WriteEEPROM

Function description	This function is used to write data to EEPROM.
Function definition	bool CH347WriteEEPROM(int fd, EEPROM_TYPE iEepromID, int iAddr, int iLength, uint8_t *iBuffer);
Parameter description	fd: file descriptor of device iEepromID: EEPROM type iAddr: address of EEPROM iLength: write length iBuffer: pointer to write buffer
Return value	The function return true if successful, false if fail.

3. CH341

3.1. Introduction

CH341 is a USB2.0 full-speed converter chip to realize USB to UART, USB to SPI, USB to I2C, USB to printer port and USB to EPP/MEM parallel port. CH341A/B/F can simultaneously support the above interfaces with three working modes, which need to be selected separately, CH341T supports UART and I2C interfaces.

“libch347.so/libch347.a” provides operating system-side interface operation functions for SPI/I2C/EPP/MEM parallel port etc. for the CH341 chip.

3.2. Interface Description

According to the USB interface features supported by CH341, “libch347” provides USB-SPI, USB-I2C, and USB-EPP/MEM parallel port interface function.

SCL and SDA pin states	**Driver Interface**	API
SDA is suspended and SCL is suspended	CH341SER(VCP)	Native termios API in the system
SDA is connected to low level and SCL is suspended	CH341PAR	CH34X_xxx in ch341_lib.h
The SDA is directly connected to the SCL	USB Printer Device Class driver	

Table. CH341 Interface function API

3.3. Public Functions

3.3.1. Operation process

First user needs to call CH34xOpenDevice to open the device to be operated, CH34xSetTimeout is optional to set the timeout, then the read and write functions should be called to read and write, CH34xCloseDevice is the last API to be called to close the device after the operation is completed.

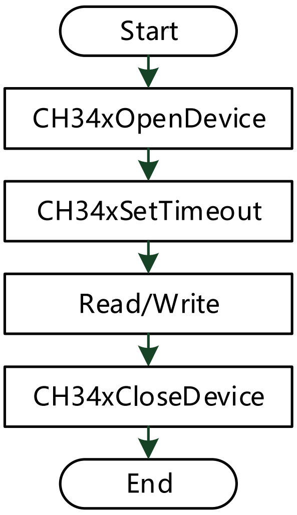

Figure 3.1 ch341lib operation flowchart

3.3.2. CH34xOpenDevice

Function description	This function is used to open device.
Function definition	int CH34xOpenDevice(const char *pathname);
Parameter description	pathname: device path in /dev directory
Return value	The function return positive file descriptor if successful, others if fail.

3.3.3. CH34xCloseDevice

Function description	This function is used to close device.
Function definition	bool CH34xCloseDevice(int fd);
Parameter description	fd: file descriptor of device
Return value	The function return true if successful, false if fail.

3.3.4. CH34x_GetDriverVersion

Function description	This function is used to get vendor driver version.
Function definition	bool CH34x_GetDriverVersion(int fd, unsigned char

	*Drv_Version);
Parameter description	fd: file descriptor of device Drv_Version: pointer to version string
Return value	The function return true if successful, false if fail.

3.3.5. CH34x_GetChipType

Function description	This function is used to get chip type.
Function definition	bool CH34x_GetChipType(int fd, CHIP_TYPE *ChipType);
Parameter description	fd: file descriptor of device ChipType: pointer to chip type
Return value	The function return true if successful, false if fail.

3.3.6. CH34x_GetDeviceID

Function description	This function is used to get device VID and PID.
Function definition	bool CH34x_GetDeviceID(int fd, uint32_t *id);
Parameter description	fd: file descriptor of device id: pointer to store id which contains VID and PID
Return value	The function return true if successful, false if fail.

3.4. EPP Functions

3.4.1. Operation Process

First user needs to call CH34xOpenDevice to open the device to be operated, CH34xSetTimeout is optional to set the timeout, then call CH34xSetParaMode to set the parallel port mode (EPP/MEM) and CH34xInitParallel to initialize the parallel port. After that the read and write functions should be called to read and write the parallel port, CH34xCloseDevice is the last API to be called to close the device after the operation is completed.

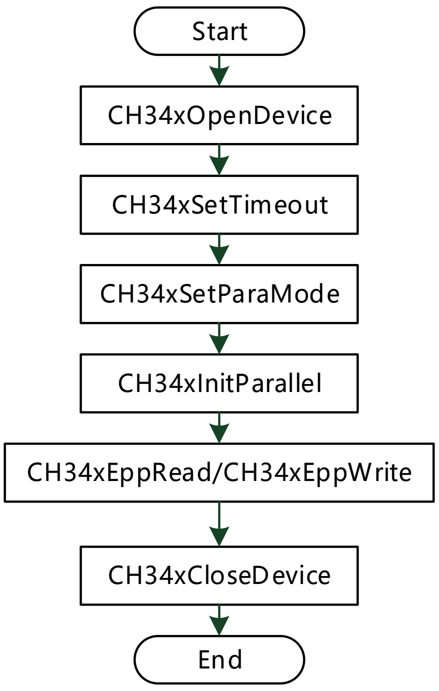

Figure 3.2 EPP port operation flowchart

3.4.2. CH34xSetParaMode

Function description	This function is used to set chip parallel port work mode.
Function definition	bool CH34xSetParaMode(int fd, uint8_t Mode);
Parameter description	fd: file descriptor of device Mode: work mode, 0/1->EPP mode, 2->MEM mode
Return value	The function return true if successful, false if fail.

3.4.3. CH34xInitParallel

Function description	This function is used to initial chip parallel work mode.
Function definition	bool CH34xInitParallel(int fd, uint8_t Mode);
Parameter description	fd: file descriptor of device Mode: work mode, 0/1->EPP mode, 2->MEM mode
Return value	The function return true if successful, false if fail.

3.4.4. CH34xEppRead

Function description	This function is used to read data or address in parallel EPP mode.
Function definition	int CH34xEppRead(int fd, uint8_t *oBuffer, uint32_t ioLength, uint8_t PipeMode);
Parameter description	fd: file descriptor of device oBuffer: pointer to read buffer

	ioLength: read length PipeMode: 0->read pipe0 data, 1->read pipe1 address
Return value	The function return read 0 if successful, others if fail.

3.4.5. CH34xEppWrite

Function description	This function is used to write data or address in parallel EPP mode.
Function definition	int CH34xEppWrite(int fd, uint8_t *iBuffer, uint32_t ioLength, uint8_t PipeMode);
Parameter description	fd: file descriptor of device iBuffer: pointer to write buffer ioLength: write length PipeMode: 0->write pipe0 data, 1->write pipe1 address
Return value	The function return true if successful, false if fail.

3.4.6. CH34xEppWriteData

Function description	This function is used to write data in parallel EPP mode.
Function definition	int CH34xEppWriteData(int fd, uint8_t *iBuffer, uint32_t ioLength);
Parameter description	fd: file descriptor of device iBuffer: pointer to write buffer ioLength: write length
Return value	The function return 0 if successful, others if fail.

3.4.7. CH34xEppReadData

Function description	This function is used to read data in parallel EPP mode.
Function definition	int CH34xEppReadData(int fd, uint8_t *oBuffer, uint32_t ioLength);
Parameter description	fd: file descriptor of device oBuffer: pointer to read buffer ioLength: read length
Return value	The function return 0 if successful, others if fail.

3.4.8. CH34xEppWriteAddr

Function description	This function is used to write address in parallel EPP mode.
Function definition	int CH34xEppWriteAddr(int fd, uint8_t *iBuffer, uint32_t ioLength);
Parameter description	fd: file descriptor of device iBuffer: pointer to write buffer

	ioLength: write length
Return value	The function return 0 if successful, others if fail.

3.4.9. CH34xEppReadAddr

Function description	This function is used to read address in parallel EPP mode.
Function definition	int CH34xEppReadAddr(int fd, uint8_t *oBuffer, uint32_t ioLength);
Parameter description	fd: file descriptor of device oBuffer: pointer to read buffer ioLength: read length
Return value	The function return true if successful, false if fail.

3.4.10. CH34xEppSetAddr

Function description	This function is used to set address in parallel EPP mode.
Function definition	int CH34xEppSetAddr(int fd, uint32_t iAddr);
Parameter description	fd: file descriptor of device iAddr: address data
Return value	The function return 0 if successful, others if fail.

3.4.11. CH34xSetTimeout

Function description	This function is used to set USB data read and write timeout.
Function definition	bool CH34xSetTimeout(int fd, uint32_t iWriteTimeout, uint32_t iReadTimeout);
Parameter description	fd: file descriptor of device iWriteTimeout: data download timeout in milliseconds iReadTimeout: data upload timeout in milliseconds
Return value	The function return true if successful, false if fail.

3.5. MEM Functions

3.5.1. CH34xInitMEM

Function description	This function is used to initial chip in parallel MEM mode.
Function definition	bool CH34xInitMEM(int fd);
Parameter description	fd: file descriptor of device
Return value	The function return true if successful, false if fail.

3.5.2. CH34xMEMReadData

Function description	This function is used to read data in parallel MEM mode.
Function definition	int CH34xMEMReadData(int fd, uint8_t *oBuffer, uint32_t ioLength, uint8_t PipeMode);
Parameter description	fd: file descriptor of device oBuffer: pointer to read buffer ioLength: read length PipeMode: 0->read pipe0, 1->read pipe1
Return value	The function return 0 if successful, others if fail.

3.5.3. CH34xMEMWriteData

Function description	This function is used to write data in parallel MEM mode.
Function definition	int CH34xMEMWriteData(int fd, uint8_t *iBuffer, uint32_t ioLength, uint32_t PipeMode);
Parameter description	fd: file descriptor of device iBuffer: pointer to write buffer ioLength: write length PipeMode: 0->write pipe0, 1->write pipe1
Return value	The function return 0 if successful, others if fail.

3.6. I2C/SPI Functions

3.6.1. CH34xSetStream

Function description	This function is used to configure SPI/I2C interface in stream mode.
Function definition	bool CH34xSetStream(int fd, uint8_t Mode);
Parameter description	fd: file descriptor of device Mode: stream mode ->bit0~1: set I2C SCL rate --> 00 : low rate 20KHz --> 01 : standard rate 100KHz --> 10 : fast rate 400KHz --> 11 : high rate 750KHz ->bit2: set spi mode --> 0 : one in one out(D3: clock, D5: out, D7: in) --> 1 : two in two out(D3 :clock, D4/D5: out, D6/D7 :in) ->bit7: set spi data mode --> 0 : low bit first --> 1 : high bit first other bits must keep 0

Return value	The function return true if successful, false if fail.

3.6.2. CH34xSetDelay_mS

Function description	This function is used to delay operation.
Function definition	bool CH34xSetDelaymS(int fd, uint32_t iDelay);
Parameter description	fd: file descriptor of device iDelay: delay time in millseconds
Return value	The function return true if successful, false if fail.

3.6.3. CH34xGetInput

Function description	This function is used to get IO status.
Function definition	bool CH34xGetInput(int fd, uint32_t *iStatus);
Parameter description	fd: file descriptor of deviceiStatus: pointer to IO status Note: Bit7~Bit0<==>D7-D0, Bit8<==>ERR#, Bit9<==>PEMP, Bit10<==>INT# Bit11<==>SLCT, Bit13<==>WAIT#, Bit14<==>DATAS#/READ# Bit15<==>ADDRS#/ADDR/ALE, Bit23<==>SDA
Return value	The function return true if successful, false if fail.

3.6.4. CH34xSetOutput

Function description	This function is used to set IO direction and value.
Function definition	bool CH34xSetOutput(int fd, uint32_t iEnable, uint32_t iSetDirOut, uint32_t iSetDataOut);
Parameter description	fd: file descriptor of device iEnable: set direction and data enable --> Bit16 High : effect on Bit15~8 of iSetDataOut --> Bit17 High : effect on Bit15~8 of iSetDirOut --> Bit18 High : effect on Bit7~0 of iSetDataOut --> Bit19 High : effect on Bit7~0 of iSetDirOut --> Bit20 High : effect on Bit23~16 of iSetDataOut iSetDirOut: set IO direction -- > Bit High : Output -- > Bit Low : Input iSetDataOut: set IO value

	-- > Bit High : High level -- > Bit Low : Low level Note: Bit7~Bit0<==>D7-D0, Bit8<==>ERR#, Bit9<==>PEMP, Bit10<==>INT# Bit11<==>SLCT, Bit13<==>WAIT#, Bit14<==>DATAS#/READ#, Bit15<==>ADDRS#/ADDR/ALE The pins below can only be used in output mode: Bit16<==>RESET#, Bit17<==>WRITE#, Bit18<==>SCL, Bit29<==>SDA
Return value	The function return true if successful, false if fail.

3.6.5. CH34xSet_D5_D0

Function description	This function is used to set direction and value of D5-D0.
Function definition	bool CH34xSet_D5_D0(int fd, uint8_t iSetDirOut, uint8_t iSetDataOut);
Parameter description	fd: file descriptor of device iSetDirOut: set IO direction -- > Bit High : Output -- > Bit Low : Input iSetDataOut: set IO value -- > Bit High : High level -- > Bit Low : Low level
Return value	The function return true if successful, false if fail.

3.6.6. CH34xStreamI2C

Function description	This function is used to write/read I2C in stream mode.
Function definition	bool CH34xStreamI2C(int fd, uint32_t iWriteLength, void *iWriteBuffer, uint32_t iReadLength, void *oReadBuffer);
Parameter description	fd: file descriptor of device iWriteLength: write length iWriteBuffer: pointer to write buffer iReadLength: read length oReadBuffer: pointer to read buffer

Return value	The function return true if successful, false if fail.

3.6.7. CH34xReadEEPROM

Function description	This function is used to read data from EEPROM.
Function definition	bool CH34xReadEEPROM(int fd, EEPROM_TYPE iEepromID, uint32_t iAddr, uint32_t iLength, uint8_t *oBuffer);
Parameter description	fd: file descriptor of device iEepromID: EEPROM type iAddr: address of EEPROM iLength: read length oBuffer: pointer to read buffer
Return value	The function return true if successful, false if fail.

3.6.8. CH34xWriteEEPROM

Function description	This function is used to write data to EEPROM.
Function definition	bool CH34xWriteEEPROM(int fd, EEPROM_TYPE iEepromID, uint32_t iAddr, uint32_t iLength, uint8_t *iBuffer);
Parameter description	fd: file descriptor of device iEepromID: EEPROM type iAddr: address of EEPROM iLength: write length iBuffer: pointer to write buffer
Return value	The function return true if successful, false if fail.

3.6.9. CH34xStreamSPI4

Function description	This function is used to write/read SPI in 4-line stream mode.
Function definition	bool CH34xStreamSPI4(int fd, uint32_t iChipSelect, uint32_t iLength, void *ioBuffer);
Parameter description	fd: file descriptor of device iChipSelect: cs enable iLength: the length of data ioBuffer: one in one out buffer
Return value	The function return true if successful, false if fail.

3.6.10. CH34xStreamSPI5

Function description	This function is used to write/read SPI in 5-line stream mode.

Function definition	bool CH34xStreamSPI5(int fd, uint32_t iChipSelect, uint32_t iLength, void *ioBuffer, void *ioBuffer2);
Parameter description	fd: file descriptor of device iChipSelect: cs enable, bit7: 0->ignore cs, 1->cs valid, bit[1:0]: 00/01/10 select D0/D1/D2 as active chip select iLength: the length of data ioBuffer: first buffer of two in two out buffer, write from DOUT, read from DIN ioBuffer2: second buffer of two in two out buffer, write from DOUT2, read from DIN2
Return value	The function return true if successful, false if fail.