USB to JTAG and SPI Communication Protocol

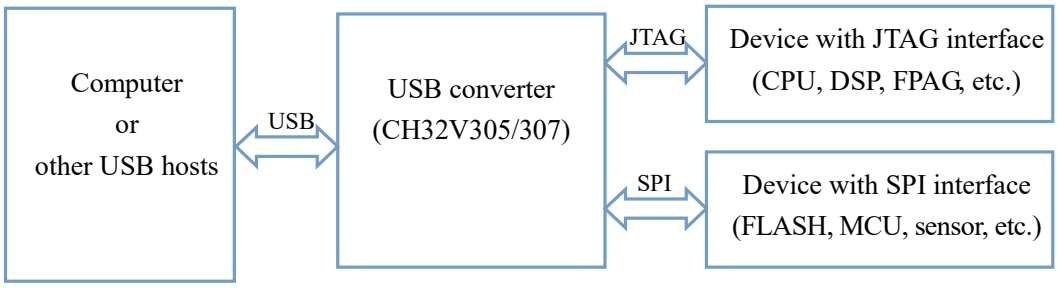
1. Communication structure

This manual mainly describes the USB to high-speed JTAG interface and high-speed SPI interface communication protocol, using the WCH MCU with built-in USB high-speed (480Mbps) interface (CH32V305/CH32V307).

CH32V305/307 is an industrial-grade general-purpose MCU based on Qingke 32-bit RISC-V, equipped with hardware stack and fast interrupt entry. The interrupt response speed is greatly improved on the basis of standard RISC-V. Equipped with V4F core. The system clock frequency can be 144MHz. Independent GPIO power supply. Built-in two 12-bit ADC modules, two 12-bit DAC modules, several timers, multi-channel capacitive touch-key detection (TKEY) and other functions. It also provides standard and dedicated communication interfaces: I2C, I2S, SPI, USART, SDIO, CAN controller, USB2.0 full-speed host/device controller, USB2.0 (480Mbps) high-speed host/device controller, digital video port, Gigabit Ethernet controller and so on.

Hereinafter, CH32V305/307 is referred to as USB converter for short.

The figure below shows the structure.



2. Communication method

High-speed USB interface communication is used between a USB converter and a computer (or other USB host). JTAG interface communication is used between a USB converter and a device with JTAG interface, to operate CPU, DSP, FPGA and CPLD and other devices. SPI interface communication is used between a USB converter and a device with SPI interface, to operate FLASH, MCU and sensors.

2.1. Frame format description

The communication between a USB converter and a computer takes the frame as a unit, that is, it is sent in the form of data packets, and each frame of data has a command code, data length and subsequent data.

Hereinafter, the communication frame initiated by computer is referred to as a "command packet", and the communication frame returned by USB converter is referred to as a "response packet". Note: All data described below are in hexadecimal format.

Data format of the command packet and the response packet:

CMD	LEN	DATA
Command code	Data length	Subsequent data
1-byte	2-byte (low byte in front)	0 ~ 507 bytes (high speed) 0 ~ 59 bytes (full speed)

Command code: 1-byte. Valid range: 0xC0---0xE0.
Data length: 2-byte. Mainly used to record the actual length of the subsequent data of the packet, including only the subsequent data, excluding the command code and the data length itself.
Subsequent data: N-byte.
In high speed mode, each USB packet is up to 512 bytes, and 2 bytes are reserved. So each frame is up to 510 bytes.
In full speed mode, each USB packet is up to 64 bytes, and 2 bytes are reserved. So each frame is up to 62 bytes.

DEF_FS_PACK_MAX_LEN = 59
DEF_HS_PACK_MAX_LEN = 507

2.2. Command code description

Table 1 Command code table

No.	Command code	Command description
1	0xC0	**SPI initialization command (DEF_CMD_SPI_INIT)** Used to initialize SPI interface, set the data bits, divided clock frequency, high/low byte order and other parameters of the SPI interface. For details, see the following detailed description.
2	0xC1	**SPI control command (DEF_CMD_SPI_CONTROL)** Used to control the output level and level delay time of the Chip Select pin of the SPI interface.
3	0xC2	**SPI normal read and write command (DEF_CMD_SPI_RD_WR)** Used for SPI normal read and write operations, usually for short general command operations. This command writes N bytes of data and reads back N bytes of data at the same time.
4	0xC3	**SPI bulk read command (DEF_CMD_SPI_BLCK_RD)** Used for SPI to read data in bulk, usually for bulk data read operations. After enabling this command to read data, the read data is packaged and uploaded in the maximum package size until all read data is returned.
5	0xC4	**SPI bulk write command (DEF_CMD_SPI_BLCK_WR)** Used for SPI to write data in bulk, usually for bulk data write operations
6	0xCA	**Obtain interface parameters (DEF_CMD_INFO_RD)** Used to obtain firmware version, SPI, IIC, JTAG interface related parameters
7	0xD0	**JTAG initialization command (DEF_CMD_JTAG_INIT)**

		Used to initialize JTAG interface. Mainly used to set mode and speed.
8	0xD1	**JTAG pin bit control command (DEF_CMD_JTAG_BIT_OP)** Used for computer to directly control the corresponding pins of the chip, output high/low level.
9	0xD2	**JTAG pin bit control and read command (DEF_CMD_JTAG_BIT_OP_RD)** Used for computer to directly control the corresponding pins of the chip, output high/low level, and read back data
10	0xD3	**JTAG data shift command (DEF_CMD_JTAG_DATA_SHIFT)** Used to shift data out through JTAG interface
11	0xD4	**JTAG data shift and read command (DEF_CMD_JTAG_DATA_SHIFT_RD)** Used to shift data out through JTAG interface, and read back data

2.2.1. SPI initialization command (DEF_CMD_SPI_INIT)

This command is mainly used to set the operating mode of SPI interface (always Host), data bits, divided clock frequency, high/low byte order and other parameters. For details, see the header file: Appendix 1.

Command packet:

CMD	LEN	DATA
Command code	Data length	Subsequent data
0xC0	2-byte (0x1A, 0x00)	26-byte parameter

Subsequent data: 26 bytes

(1) 18-byte SPI initialization parameter. For details, see the SPI_InitTypeDef structure in "Appendix 1".

(2) Delay value between reading/writing 2 bytes of data (only for **SPI normal read and write command (DEF_CMD_SPI_RD_WR)**), the unit is uS.

(3) 1-byte SPI transmit default byte.

(4) 1-byte miscellaneous control.

Bit 7: CS1 chip select polarity control: 0: Active low. 1: Active high.

Bit 6: CS2 chip select polarity control: 0: Active low. 1: Active high.

Bits 5-0: Reserved.

(5) 4 bytes reserved.

Response packet:

CMD	LEN	DATA
Command code	Data length	Subsequent data
0xC0	2-byte (0x01, 0x00)	1-byte execution status

Subsequent data: 1-byte. Return 11-byte execution status. 0x00: Success. 0x01: Failure.

2.2.2. SPI control command (DEF_CMD_SPI_CONTROL)

This command is used to control the output level and level delay time of the chip select pin of SPI interface

Command packet:

CMD	LEN	DATA
Command code	Data length	Subsequent data
0xC1	2-byte (0x0A, 0x00)	10-byte parameter

Subsequent data: 10 bytes:

(1) 1- byte chip select (CS1) pin control and subsequent control:

Bit 7: Whether the CS1 chip select control bit is valid. 0: Invalid (ignore bit 6). 1: Valid (Output level depends on the value of bit 6).

Bit 6: CS1 chip select control bit. 0: Output low level. 1: Output high level.

Bit 5: Whether the CS1 chip select needs to be automatically canceled after the next SPI read/write operation command is executed.

0: Not cancel automatically. 1: Cancel automatically.

Bits 4-0: Reserved.

(2) 2-byte chip select (CS1) pin current control delay time. The unit is uS. Low byte in front. 0x0000 means no delay.

For example, after pulling down the chip select, delay 1000uS before performing read/write operations, and the 2 bytes are 0xE8 and 0x03.

(3) 2-byte chip select (CS1) pin subsequent control delay time. The unit is uS. Low byte in front. 0x0000 means no delay.

For example, after the read/write operations are performed, the chip select pin needs to be automatically cancelled, and delay 500uS before cancellation. The 2 bytes can be set to: 0xF4, 0x01.

(4) 1-byte chip select (CS2) pin control and subsequent control:

Bit 7: Whether the CS2 chip select control bit is valid. 0: Invalid (ignore bit 6). 1: Valid (Output level depends on the value of bit 6).

Bit 6: CS2 chip select control bit. 0: Output low level. 1: Output high level.

Bit 5: Whether the CS2 chip select needs to be automatically canceled after the next SPI read/write operation command is executed.

0: Not cancel automatically. 1: Cancel automatically.

Bits 4-0: Reserved.

(5). 2-byte chip select (CS2) pin current control delay time. The unit is uS. Low byte in front. 0x0000 means no delay.

For example, after pulling down the chip select, delay 1000uS before performing read/write operations, and the 2 bytes are 0xE8 and 0x03.

(6) 2-byte chip select (CS2) pin subsequent control delay time. The unit is uS. Low byte in front. 0x0000 means no delay.

For example, after the read/write operations are performed, the chip select pin needs to be automatically cancelled, and delay 500uS before cancellation. The 2 bytes can be set to: 0xF4, 0x01.

Response packet: None.

2.2.3. SPI normal read and write command (DEF_CMD_SPI_RD_WR)

This command is used for SPI normal read and write operations, and usually for short general command operations.

When this command writes N bytes of data, it reads back N bytes of data.

Command packet:

CMD	LEN	DATA
Command code	Data length	Subsequent data
0xC2	2-byte	N-byte

Subsequent data: N bytes: N-byte SPI data to be transmitted: Valid range of N:

Full speed mode: 0<N<= DEF_FS_PACK_MAX_LEN.

High speed mode: 0<N<= DEF_HS_PACK_MAX_LEN.

Response packet:

CMD	LEN	DATA
Command code	Data length	Subsequent data
0xC2	2-byte	N-byte

Subsequent data: N bytes: N-byte SPI data to be read back.

2.2.4. SPI bulk read command (DEF_CMD_SPI_BLCK_RD)

This command is used for SPI to read data in bulk, and usually for bulk data read operations.

After enabling this command, the chip automatically reads data from the SPI interface chip according to the length of the data to be read, and then packs and uploads the read data in the maximum packet size until all data is read and returned.

Command packet:

CMD	LEN	DATA
Command code	Data length	Subsequent data
0xC3	2-byte (0x04, 0x00)	4-byte data length to be read (Low byte in front)

Subsequent data: 4 bytes: 4-byte data length to be read. Low byte in front. For example, to read 2048 bytes, the 4 bytes are in order: 0x00, 0x08 0x00, 0x00.

Response packet:

CMD	LEN	DATA
Command code	Data length	Subsequent data
0xC3	2-byte	N-byte read data

Full speed mode: 0<N<= DEF_FS_PACK_MAX_LEN.

High speed mode: 0<N<= DEF_HS_PACK_MAX_LEN.

For example, to read 2048 bytes of data, return the read data in the maximum packet size in order.

2.2.5. SPI bulk write command (DEF_CMD_SPI_BLCK_WR)

This command is used for SPI to write data in bulk, and usually for bulk data write operations.
Command packet:

CMD	LEN	DATA
Command code	Data length	Subsequent data
0xC4	2-byte	N-byte

Subsequent data: N bytes:

(1) N-byte data to be written. After the chip receives this command, it directly enables DMA to send N bytes of data to the SPI interface chip directly. At the end of transmission, answer the computer.

Note: Since general FLASH operations are performed by page (256 bytes), it is recommended to write data up to 256 bytes at a time.

Response packet:

CMD	LEN	DATA
Command code	Data length	Subsequent data
0xC4	2-byte	1-byte execution status

Subsequent data: 1-byte. Return 1-byte execution status. 0x00: Success. 0x01: Failure.

2.2.6. Obtain interface parameter (DEF_CMD_INFO_RD)

This command is used to obtain firmware version, SPI, JTAG interface related parameters, etc.
Command packet:

CMD	LEN	DATA
Command code	Data length	Subsequent data
0xCA	2-byte (0x01, 0x00)	1-byte parameter

Subsequent data: 1-byte. It indicates the type of parameter obtained.

0x00: Chip-related information.

0x01: SPI-related information.

0x02: JTAG-related information.

Response packet:

CMD	LEN	DATA
Command code	Data length	Subsequent data
0xCA	2-byte	N-byte returned data

Subsequent data: N bytes. Depending on the parameter type to be obtained, the returned data is different.

(1) If the parameter type obtained is 0x00, the returned data occupies 4 bytes:

(a) 1-byte firmware version.

(b) 3-byte byte reserved.

(2) If the parameter type obtained is 0x01, the returned data occupies 26 bytes:

(a) 18-byte SPI initialization parameter. For details, see the SPI_InitTypeDef structure in

"Appendix 1".

(b) 2-byte delay value between data read/write operations. The unit is uS.

(c) 1-byte SPI transmit default byte.

(d) 1-byte miscellaneous control.

(e) 4 bytes reserved.

(3) If the parameter type obtained is 0x02, the returned data occupies 6 bytes:

(a) 1-byte JTAG mode.

(b) 1-byte JTAG speed.

(c) 4 bytes reserved.

2.2.7. JTAG initialization command (DEF_CMD_JTAG_INIT)

This command is used to initialize the JTAG interface, mainly to set the mode and speed.

Command packet:

CMD	LEN	DATA
Command code	Data length	Subsequent data
0xD0	2-byte (0x06, 0x00)	6-byte

Subsequent data: 6 bytes:

(1) 1 byte: Operating mode.

0x00: Fast mode for custom protocols.

0x01: bit-bang mode.

(2) 1 byte: Communication speed. Valid value: 0 to 5. The larger the value, the faster the communication speed.

Speed 0: 2.25MHz.

Speed 1: 4.5MHz.

Speed 2: 9MHz.

Speed 3: 18MHz.

Speed 4: 36MHz.

Speed 5: 72MHz.

(3) 4 bytes: Reserved.

Response packet:

CMD	LEN	DATA
Command code	Data length	Subsequent data
0xD0	2-byte (0x01, 0x00)	1-byte

Subsequent data: 1-byte command execution status. 0: Success. 1: Failure.

2.2.8. JTAG pin bit control command (DEF_CMD_JTAG_BIT_OP)

This command is mainly used to directly control the corresponding pins of the chip by the computer, to output high/low level.

Command packet:

CMD	LEN	DATA
Command code	Data length	Subsequent data
0xD1	2-byte	N-byte data

Response packet: None.

2.2.9. JTAG pin bit control and read command (DEF_CMD_JTAG_BIT_OP_RD)

This command is mainly used for computer to directly control the corresponding pins of the chip, output high/low level, and read back data.

Command packet:

CMD	LEN	DATA
Command code	Data length	Subsequent data
0xD2	2-byte	N-byte data

Response packet:

CMD	LEN	DATA
Command code	Data length	Subsequent data
0xD2	2-byte	N-byte data

Return N-byte data read from JTAG interface.

2.2.10. JTAG data shift command (DEF_CMD_JTAG_DATA_SHIFT)

This command is mainly used to shift the data out through the JTAG interface.

Command packet:

CMD	LEN	DATA
Command code	Data length	Subsequent data
0xD3	2-byte	N-byte data

Response packet: None.

2.2.11. JTAG data shift and read command (DEF_CMD_JTAG_DATA_SHIFT_RD)

This command is mainly used to shift the data out through the JTAG interface, and read back.

Command packet:

CMD	LEN	DATA
Command code	Data length	Subsequent data
0xD4	2-byte	N-byte data

Response packet:

CMD	LEN	DATA
Command code	Data length	Subsequent data
0xD4	2-byte	N-byte data

Return N-byte data read from JTAG interface.

Appendix 1

```
/* SPI Init structure definition */
typedef struct
{
  uint16_t SPI_Direction;          /* Specifies the SPI unidirectional or
bidirectional data mode.
                                    This parameter can be a value of @ref
SPI_data_direction */

  uint16_t SPI_Mode;               /* Specifies the SPI operating mode.
                                    This parameter can be a value of @ref
SPI_mode */

  uint16_t SPI_DataSize;           /* Specifies the SPI data size.
                                    This parameter can be a value of @ref
SPI_data_size */

  uint16_t SPI_CPOL;               /* Specifies the serial clock steady
state.
                                    This parameter can be a value of @ref
SPI_Clock_Polarity */

  uint16_t SPI_CPHA;               /* Specifies the clock active edge for
the bit capture.
                                    This parameter can be a value of @ref
SPI_Clock_Phase */

  uint16_t SPI_NSS;                /* Specifies whether the NSS signal is
managed by
                                    hardware (NSS pin) or by software using
the SSI bit.
                                    This parameter can be a value of @ref
SPI_Slave_Select_management */

  uint16_t SPI_BaudRatePrescaler;   /* Specifies the Baud Rate prescaler
value which will be
                                    used to configure the transmit and
receive SCK clock.
                                    This parameter can be a value of @ref
SPI_BaudRate_Prescaler.
                                    @note The communication clock is derived
from the master
```

```
                                        clock. The slave clock does not need
to be set. */

  uint16_t SPI_FirstBit;          /* Specifies whether data transfers
start from MSB or LSB bit.
                                   This parameter can be a value of @ref
SPI_MSB_LSB_transmission */

  uint16_t SPI_CRCPolynomial;      /* Specifies the polynomial used for
the CRC calculation. */
}SPI_InitTypeDef;

/* SPI_data_direction */
#define SPI_Direction_2Lines_FullDuplex ((uint16_t)0x0000)
#define SPI_Direction_2Lines_RxOnly     ((uint16_t)0x0400)
#define SPI_Direction_1Line_Rx          ((uint16_t)0x8000)
#define SPI_Direction_1Line_Tx          ((uint16_t)0xC000)

/* SPI_mode */
#define SPI_Mode_Master                 ((uint16_t)0x0104)
#define SPI_Mode_Slave                  ((uint16_t)0x0000)

/* SPI_data_size */
#define SPI_DataSize_16b                ((uint16_t)0x0800)
#define SPI_DataSize_8b                 ((uint16_t)0x0000)

/* SPI_Clock_Polarity */
#define SPI_CPOL_Low                    ((uint16_t)0x0000)
#define SPI_CPOL_High                   ((uint16_t)0x0002)

/* SPI_Clock_Phase */
#define SPI_CPHA_1Edge                  ((uint16_t)0x0000)
#define SPI_CPHA_2Edge                  ((uint16_t)0x0001)

/* SPI_Slave_Select_management */
#define SPI_NSS_Soft                    ((uint16_t)0x0200)
#define SPI_NSS_Hard                    ((uint16_t)0x0000)

/* SPI_BaudRate_Prescaler */
#define SPI_BaudRatePrescaler_2         ((uint16_t)0x0000)
#define SPI_BaudRatePrescaler_4         ((uint16_t)0x0008)
#define SPI_BaudRatePrescaler_8         ((uint16_t)0x0010)
#define SPI_BaudRatePrescaler_16        ((uint16_t)0x0018)
#define SPI_BaudRatePrescaler_32        ((uint16_t)0x0020)
```

```
#define SPI_BaudRatePrescaler_64       ((uint16_t)0x0028)
#define SPI_BaudRatePrescaler_128      ((uint16_t)0x0030)
#define SPI_BaudRatePrescaler_256      ((uint16_t)0x0038)

/* SPI_MSB_LSB_transmission */
#define SPI_FirstBit_MSB              ((uint16_t)0x0000)
#define SPI_FirstBit_LSB              ((uint16_t)0x0080)
```