USB 转 JTAG 和 SPI 接口通信协议

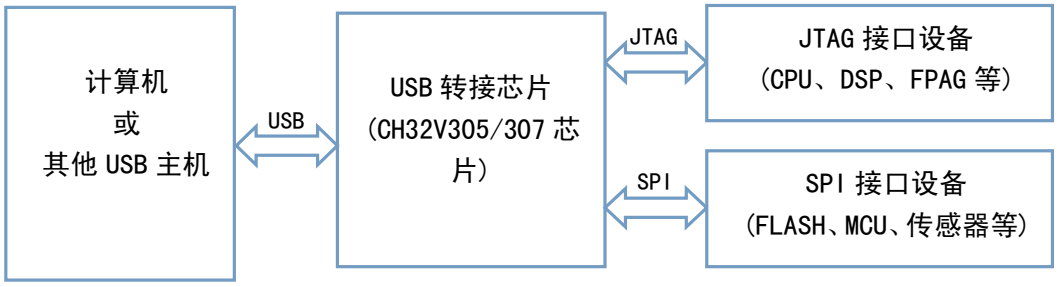
1、通信结构

本文主要描述采用沁恒微电子的内置 USB 高速(480Mbps)接口 MCU(CH32V305/CH32V307 系列芯片)实现 USB 转高速 JTAG 接口和高速 SPI 接口的通信协议。

CH32V305/307 系列芯片是基于青稞 32 位 RISC-V 设计的工业级通用微控制器。配备了硬件堆栈区、快速中断入口，在标准 RISC-V 基础上大大提高了中断响应速度。搭载 V4F 内核，主频支持 144MHz，独立 GPIO 供电，内置 2 个 12 位 ADC 模块、2 个 12 位 DAC 模块、多组定时器、多通道触摸按键电容检测(TKEY)等功能，还包含标准和专用通讯接口：I2C、I2S、SPI、USART、SDIO、CAN 控制器、USB2.0 全速主机/设备控制器、USB2.0(480Mbps)高速主机/设备控制器、数字图像接口、千兆以太网控制器等。

以下简称 CH32V305/307 芯片为 USB 转接芯片。

其结构框图如下：



2、通信方式

USB 转接芯片与计算机(或其他 USB 主机)之间采用高速 USB 接口通信；USB 转接芯片与 JTAG 接口设备之间采用 JTAG 接口通信，可操作 CPU、DSP、FPGA 和 CPLD 等器件；USB 转接芯片与 SPI 接口设备之间采用 SPI 接口通信，可操作 FLASH、MCU 和传感器等器件。

2.1、帧格式说明

USB 转接芯片与计算机之间的通信以帧为单位，即以数据包的形式发送，每帧数据都带有命令码、数据长度和后续数据。

以下将计算机发起的通信帧称为“命令包”，USB 转接芯片返回的通信帧称为“应答包”。

注：以下所有描述的数据均为 16 进制格式。

命令包及应答包数据格式如下：

CMD	LEN	DATA
命令码	数据长度	后续数据
1 个字节	2 个字节(低字节在前)	0-507 个字节(高速模式) 0-59 个字节(全速模式)

命令码：占 1 个字节，命令码有效范围为：0xC0---0xE0。

数据长度：占 2 个字节，主要用于记录该包实际后续数据的长度，仅包含后续数据部分，

不包括命令码和数据长度本身；

后续数据：占 N 个字节。

高速模式下每个 USB 包最大为 512 字节，预留 2 个字节，因此每个帧最多为 510 字节。

全速模式下每个 USB 包最大为 64 字节，预留 2 个字节，因此每个帧最多为 62 字节。

DEF_FS_PACK_MAX_LEN = 59

DEF_HS_PACK_MAX_LEN = 507

2.2、命令码说明

表 1-命令码表

序号	命名码	命令说明
1	0xC0	**SPI 接口初始化命令(DEF_CMD_SPI_INIT)** 用于初始化 SPI 接口，设置 SPI 接口的数据位、时钟分频、高低字节顺序等等参数。具体见后续详解。
2	0xC1	**SPI 接口控制命令(DEF_CMD_SPI_CONTROL)** 用于控制 SPI 接口片选引脚输出高低电平以及电平延时时间。
3	0xC2	**SPI 接口常规读取写入数据命令(DEF_CMD_SPI_RD_WR)** 用于 SPI 接口通用读取写入操作，一般用于简短常规命令操作。 该命令写 N 个字节数据的同时会回读 N 个字节数据。
4	0xC3	**SPI 接口批量读取数据命令(DEF_CMD_SPI_BLCK_RD)** 用于 SPI 接口批量读取数据，一般用于批量数据的读取操作。 启用该命令读取数据后，读取的数据会按最大包大小进行打包上传，直到所有数据读取返回完毕。
5	0xC4	**SPI 接口批量写入数据命令(DEF_CMD_SPI_BLCK_WR)** 用于 SPI 接口批量写入数据，一般用于批量数据的写入操作。
6	0xCA	**接口参数获取(DEF_CMD_INFO_RD)** 用于获取固件版本、SPI、IIC、JTAG 接口相关参数等
7	0xD0	**JTAG 接口初始化命令(DEF_CMD_JTAG_INIT)** 用于对 JTAG 接口进行初始化，主要是设置模式及速度
8	0xD1	**JTAG 接口引脚位控制命令(DEF_CMD_JTAG_BIT_OP)** 用于由计算机直接控制芯片的对应引脚，输出高低电平
9	0xD2	**JTAG 接口引脚位控制并读取命令(DEF_CMD_JTAG_BIT_OP_RD)** 用于由计算机直接控制芯片的对应引脚，输出高低电平，并回读数据
10	0xD3	**JTAG 接口数据移位命令(DEF_CMD_JTAG_DATA_SHIFT)** 用于将数据通过 JTAG 接口移位发送出去
11	0xD4	**JTAG 接口数据移位并读取命令(DEF_CMD_JTAG_DATA_SHIFT_RD)** 用于将数据通过 JTAG 接口移位发送出去，并进行回读

2.2.1、SPI 接口初始化命令(DEF_CMD_SPI_INIT)

该命令主要用于设置 SPI 接口的工作模式(固定为主机)、数据位、时钟分频、高低字节顺序等参数。具体见头文件：附录一。

命令包：

CMD	LEN	DATA
命令码	数据长度	后续数据

0xC0	2 个字节 (0x1A、0x00)	26 个字节参数

后续数据，占 26 个字节，

(1)、18 个字节的 SPI 初始化参数，为 SPI 具体见“附录一”中的结构体 SPI_InitTypeDef；

(2)、2 个字节数据读写之间的延时值(仅针对 **SPI 接口常规读取写入数据命令(DEF_CMD_SPI_RD_WR)**)，单位为 uS；

(3)、1 个字节 SPI 发送默认字节；

(4)、1 个字节杂项控制；

位 7：片选 CS1 极性控制：0：低电平有效；1：有电平有效；

位 6：片选 CS2 极性控制：0：低电平有效；1：有电平有效；

位 5-0：保留；

(5)、4 个字节保留；

应答包：

CMD	LEN	DATA
命令码	数据长度	后续数据
0xC0	2 个字节 (0x01、0x00)	1 个字节执行状态

后续数据，占 1 个字节，返回 1 个字节的执行状态，0x00 表示执行成功，0x01 表示执行失败。

2.2.2、SPI 接口控制命令(DEF_CMD_SPI_CONTROL)

该命令用于控制 SPI 接口片选引脚输出高低电平以及电平延时时间。

命令包：

CMD	LEN	DATA
命令码	数据长度	后续数据
0xC1	2 个字节 (0x0A、0x00)	10 个字节参数

后续数据，占 10 个字节，依次为：

(1)、1 个字节片选(CS1)引脚控制及后续控制：

位 7：片选 CS1 控制位是否有效，0：无效(忽略位 6)；1：有效(输出高低电平取决于位 6 的值)；

位 6：片选 CS1 控制位，0：控制输出低电平；1：控制输出高电平；

位 5：片选 CS1 在执行完下一条 SPI 读写操作命令后，是否需要自动撤销片选 CS1，0：不自动撤销；1：自动撤销；

位 4-0：保留；

(2)、2 个字节片选(CS1)引脚当前控制延时时间，单位为 uS，低字节在前；0x0000 表示不进行延时；

比如在拉低片选之后，先延时 1000uS 再进行读写操作，则该 2 个字节为 0xE8、0x03；

(3)、2 个字节片选(CS1)引脚后续控制延时时间，单位为 uS，低字节在前；0x0000 表示不进行延时；

比如在执行完读写操作之后，需要自动撤销片选引脚，且撤销前需要先延时 500uS，则该 2 个字节可以设置为：0xF4、0x01；

(4)、1 个字节片选(CS2)引脚控制及后续控制：

位 7：片选 CS2 控制位是否有效，0：无效(忽略位 6)；1：有效(输出高低电平取决于位 6 的值)；

位 6：片选 CS2 控制位，0：控制输出低电平；1：控制输出高电平；

位 5：片选 CS2 在执行完下一条 SPI 读写操作命令后，是否需要自动撤销片选 CS2，0：不自动撤销；1：自动撤销；

位 4-0：保留；

(5)、2 个字节片选(CS2)引脚当前控制延时时间，单位为 uS，低字节在前；0x0000 表示不进行延时；

比如在拉低片选之后，先延时 1000uS 再进行读写操作，则该 2 个字节为 0xE8、0x03；

(6)、2 个字节片选(CS2)引脚后续控制延时时间，单位为 uS，低字节在前；0x0000 表示不进行延时；

比如在执行完读写操作之后，需要自动撤销片选引脚，且撤销前需要先延时 500uS，则该 2 个字节可以设置为：0xF4、0x01；

应答包：无

2.2.3、SPI 接口常规读取写入数据命令(DEF_CMD_SPI_RD_WR)

该命令用于 SPI 接口通用读取写入操作，一般用于简短常规命令操作。

该命令写 N 个字节数据的同时会回读 N 个字节数据。

命令包：

CMD	LEN	DATA
命令码	数据长度	后续数据
0xC2	2 个字节	N 个字节

后续数据，占 N 个字节，依次为：N 个字节要发送的 SPI 数据。N 有效范围：

全速模式：0<N<= DEF_FS_PACK_MAX_LEN；

高速模式：0<N<= DEF_HS_PACK_MAX_LEN；

应答包：

CMD	LEN	DATA
命令码	数据长度	后续数据
0xC2	2 个字节	N 个字节

后续数据，占 N 个字节，依次为：N 个字节回读的 SPI 数据。

2.2.4、SPI 接口批量读取数据命令(DEF_CMD_SPI_BLCK_RD)

该命令用于 SPI 接口批量读取数据，一般用于批量数据的读取操作。

启用该命令后，芯片自动根据要读取的数据长度，从 SPI 接口芯片中读取数据，再将读取的数据，按照最大包大小进行打包上传，直到所有数据读取返回完毕。

命令包：

CMD	LEN	DATA
命令码	数据长度	后续数据
0xC3	2 个字节	4 个字节要读取的数

	(0x04、0x00)	据长度(低字节在前)

后续数据，占 4 个字节，依次为：4 个字节要读取的数据长度，低字节在前。比如要读取 2048 个字节，则 4 个字节依次为：0x00、0x08、0x00、0x00。

应答包：

CMD	LEN	DATA
命令码	数据长度	后续数据
0xC3	2 个字节	N 个字节读取数据

全速模式：0<N<= DEF_FS_PACK_MAX_LEN;

高速模式：0<N<= DEF_HS_PACK_MAX_LEN;

比如当前要读取 2048 字节数据，则根据最大包大小，依次返回读取的数据。

2.2.5、SPI 接口批量写入数据命令(DEF_CMD_SPI_BLCK_WR)

该命令用于 SPI 接口批量写入数据，一般用于批量数据的写入操作。

命令包：

CMD	LEN	DATA
命令码	数据长度	后续数据
0xC4	2 个字节	N 个字节

后续数据，占 N 个字节，依次为：

(1)、N 个字节要写入的数据，芯片接收到该命令后，直接启动 DMA 将 N 个字节数据之间发送到 SPI 接口芯片。发送完毕后，应答计算机。

注：由于一般 FLASH 操作都是按页(256 字节)操作的，建议一次性写入数据最大为 256 字节。

应答包：

CMD	LEN	DATA
命令码	数据长度	后续数据
0xC4	2 个字节	1 个字节执行状态

后续数据，占 1 个字节，返回 1 个字节的执行状态，0x00 表示执行成功，0x01 表示执行失败。

2.2.6、接口参数获取(DEF_CMD_INFO_RD)

该命令用于获取固件版本、SPI、JTAG 接口相关参数等。

命令包：

CMD	LEN	DATA
命令码	数据长度	后续数据
0xCA	2 个字节 (0x01、0x00)	1 个字节参数

后续数据，占 1 个字节，表示获取的参数类型。

0x00：表示获取芯片相关信息；
0x01：表示获取 SPI 接口相关信息；
0x02：表示获取 JTAG 接口相关信息；

应答包：

CMD	LEN	DATA
命令码	数据长度	后续数据
0xCA	2 个字节	N 个字节返回数据

后续数据，占 N 个字节，根据要获取的参数类型，返回数据不同。
(1)、获取的参数类型为 0x00，则返回的数据占 4 个字节，依次为：
(a)、1 个字节的固件版本；
(b)、3 个字节保留；

(2)、获取的参数类型为 0x01，则返回的数据占 26 个字节，依次为：
(a)、18 个字节的 SPI 初始化参数，为 SPI 具体见“附录一”中的结构体 SPI_InitTypeDef；
(b)、2 个字节数据读写之间的延时值，单位为 uS；
(c)、1 个字节 SPI 发送默认字节；
(d)、1 个字节杂项控制；
(e)、4 个字节保留；

(3)、获取的参数类型为 0x02，则返回的数据占 6 个字节，依次为：
(a)、1 个字节的 JTAG 模式；
(b)、1 个字节的 JTAG 速度；
(c)、4 个字节保留；

2.2.7、JTAG 接口初始化命令(DEF_CMD_JTAG_INIT)

该命令用于对 JTAG 接口进行初始化，主要是设置模式及速度。
命令包：

CMD	LEN	DATA
命令码	数据长度	后续数据
0xD0	2 个字节 (0x06、0x00)	6 个字节

后续数据，占 6 个字节，依次为：
(1)、1 个字节：工作模式；
0x00：自定义协议的快速模式；
0x01： bit-bang 模式；
(2)、1 个字节：通信速度；有效值为 0-5，值越大通信速度越快；
速度 0：2.25MHz；
速度 1：4.5MHz；
速度 2：9MHz；
速度 3：18MHz；
速度 4：36MHz；
速度 5：72MHz；

(3)、4 个字节：保留；

应答包：

CMD	LEN	DATA
命令码	数据长度	后续数据
0xD0	2 个字节 (0x01、0x00)	1 个字节

后续数据，占 1 个字节命令执行状态：0：执行成功；1：执行失败。

2.2.8、JTAG 接口引脚位控制命令(DEF_CMD_JTAG_BIT_OP)

该命令主要用于由计算机直接控制芯片的对应引脚，输出高低电平。

命令包：

CMD	LEN	DATA
命令码	数据长度	后续数据
0xD1	2 个字节	N 个字节数据

应答包：无

2.2.9、JTAG 接口引脚位控制并读取命令(DEF_CMD_JTAG_BIT_OP_RD)

该命令主要用于由计算机直接控制芯片的对应引脚，输出高低电平，并回读数据。

命令包：

CMD	LEN	DATA
命令码	数据长度	后续数据
0xD2	2 个字节	N 个字节数据

应答包：

CMD	LEN	DATA
命令码	数据长度	后续数据
0xD2	2 个字节	N 个字节数据

返回 N 个字节从 JTAG 接口读取到的数据。

2.2.10、JTAG 接口数据移位命令(DEF_CMD_JTAG_DATA_SHIFT)

该命令主要用于将数据通过 JTAG 接口移位发送出去。

命令包：

CMD	LEN	DATA
命令码	数据长度	后续数据
0xD3	2 个字节	N 个字节数据

应答包：无

2.2.11、JTAG 接口数据移位并读取命令(DEF_CMD_JTAG_DATA_SHIFT_RD)

该命令主要用于将数据通过 JTAG 接口移位发送出去，并进行回读。

命令包：

CMD	LEN	DATA
命令码	数据长度	后续数据
0xD4	2 个字节	N 个字节数据

应答包：

CMD	LEN	DATA
命令码	数据长度	后续数据
0xD4	2 个字节	N 个字节数据

返回 N 个字节从 JTAG 接口读取到的数据。

附录 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)
```