**USB转JTAG-SPI接口**

**通信协议**

**V1.0**

文档变更记录

|  |  |  |  |
| --- | --- | --- | --- |
| **版本号** | **变更范围** | **变更内容** | **修改人** |
| V1.0 | 文档建立 | 建立文档，初稿 | TECH2 |

# 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(CH32V305/307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搭载V4F内核，加入单精度浮点指令集，扩充硬件堆栈区，具有更高的运算性能。主频支持144MHZ，独立GPIO供电，扩展串口USART/UART数量到8组，电机定时器4组，并提供1组32位的通用定时器。提供USB2.0高速接口(480Mbps)并内置了PHY收发器，以太网MAC升级到千兆并集成了10M-PHY模块。

以下简称CH32V305/307芯片为USB转接芯片。

其结构框图如下：

计算机

或

其他USB主机

JTAG接口设备

(CPU、DSP、FPAG等)

USB转接芯片

(CH32V305/307芯片)

SPI

JTAG

USB

SPI接口设备

(FLASH、MCU、传感器等)

# 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  (0x04、0x00) | 4个字节要读取的数据长度(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、0x00、0x08、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)