# File manage and control chip CH376

English DataSheet
Version: 1
http://wch.cn
http://wch-ic.com

## 1. Introduction

CH376 is used as file manage control chip, used to MCU system read/write file in USB Flash Drive or SD card.

CH376 supports USB-Device Mode and USB-HOST Mode. Set the basic firmware of USB communication protocol in the inner. It is also set firmware of special communication protocol to deal with Mass-Storage devices, communication interface firmware of SD card, FAT16, FAT32 and FAT12 file system manage firmware in the inner. It supports common USB storage devices (contains USB Flash Drive/USB hard disk/USB Flash memory/USB read card) and SD cards (contain SD card with standard capacity and HC-SD card with high capacity, MMC card and TF card compatible with protocol).

CH376 supports three communication interfaces: 8-bit parallel, SPI interface or asynchronism serial interface. DSP/MCU/MPU etc. controller via the any interface of them to control CH376 chip, storage and get file in USB Flash Drive or SD card, or communication with computer.

The USB-DEVICE Mode of CH376 is totally compatible with CH372, the USB-HOST Mode of CH375 is basically compatible with CH375.

The following is the application image of CH376.
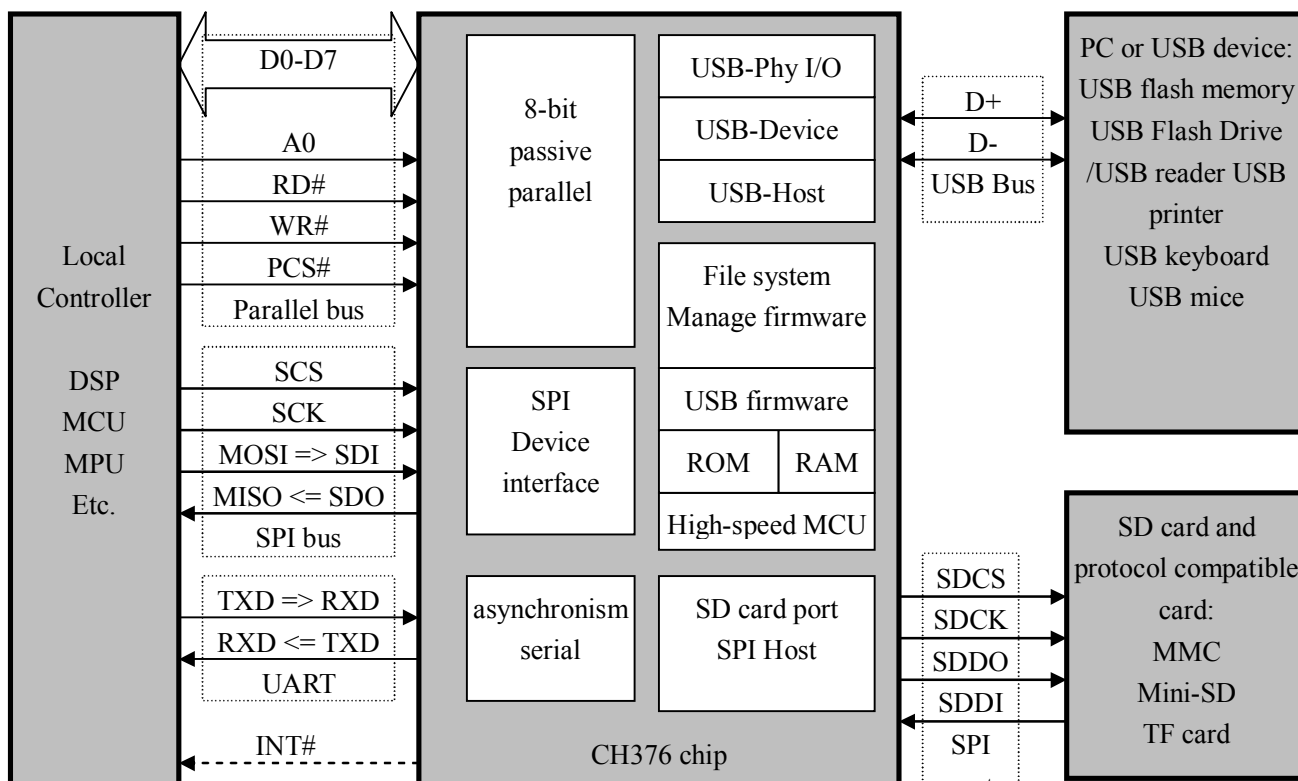


## 2. Features

● Supports 1.5Mbps low-speed and 12Mbps full-speed, compatible with USB V2.0, only needs crystal and

capacitance external.

● USB-HOST Mode and USB-DEVICE Mode, automatically switch Host Mode and Device Mode.

● Supports USB device control transfer, bulk transfer and interrupt transfer.

● Detects USB device plug and unplug automatically and sends message to USB host.

● Supply SPI host interface with 6MHz, supports SD card, MMC card and TF card which compatible with protocol.

● set protocol manager of USB control transfer in the inner to simplify common control transfer.

● Set up firmware of special communication protocol to do with Mass-storage. Supports Bulk-Only transfer protocol and USB storage device of SCSI, UFI, RBC and other equivalent storage device which accommodates the minimum set of command including USB Flash Drive, USB HD, USB flash memory and USB reader.

● Set file system manage firmware of FAT16, FAT32 and FAT12, support the capacity up to 32GB USB Flash Drive and SD Card.

● Supply file manage function: open, create or delete file, enumerate and search file, create the sub-catalog, and support the long file name.

● Supply file read/write function: read/write file in the much long sub-catalog take byte as unit or fan as unit.

● Supply disk manage function: initialize disk, query physical capacity, query spare space, physical sectors read/write.

● Supply 8-bit passive parallel interface with 2MB speed, support parallel data bus connect to MCU.

● Supply SPI device interface with 2MB/24MHz speed, support SPI serial bus connect to MCU.

● Supply asynchronism serial interface up to 3Mbps speed, support serial interface connect to MCU, support automatically adjust communication baud-rate.

● Source power is 5V or 3.3V, and low-power mode is supported.

● USB-DEVICE Mode is totally compatible with CH372 chip, USB-HOST Mode is basically compatible with CH375 chip.

●Adoption SOP-28 and SSOP20 lead-free package, compatible with RoHS, supplies diversion board from SOP28 to DIP28, the pins of SOP28 are compatible with CH375.

## 3. Package
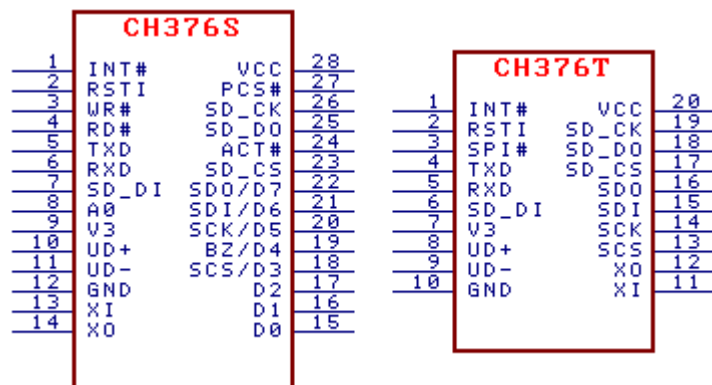


| Package shape | Width of plastic | | Pitch of pin | | Instruction of package | Ordering type |
|---------------|-----------|--------|---------|-------|---------------------------------------|---------------|
| SOP-28        | 7.62mm    | 300mil | 1.27mm  | 50mil | Small outline package of 28-pin       | CH376S        |
| SSOP-20       | 5.30mm    | 209mil | 0.65mm  | 25mil | Shrink small outline package of 20-pin| CH376T        |

## 4. Pins

| CH376S pin No. | CH376T pin No. | Pin Name | Pin Type | Pin Description |
|---|---|---|---|---|
| 28 | 20 | VCC | POWER | Positive power input port, requires an external 0.1uF power decoupling capacitance |
| 12 | 10 | GND | POWER | Public ground, ground connection for USB |
| 9 | 7 | V3 | POWER | Attachment of VCC input external power while 3.3V;connects of 0.01uF decoupling capacitance outside while 5V |
| 13 | 11 | XI | IN | Input of crystal oscillator, attachment of 12MHz crystal outside |
| 14 | 12 | XO | OUT | Opposite output of crystal oscillator, attachment of 12MHz crystal outside |
| 10 | 8 | UD+ | USB signal | USB Data Signal plus |
| 11 | 9 | UD- | USB signal | USB Data Signal minus |
| 23 | 17 | SD_CS | Drain open OUT | SPI interface chip select output of SD, active with low-level, with pull-up resister |
| 26 | 19 | SD_CK | OUT | SPI interface serial clock output of SD |
| 7 | 6 | SD_DI | IN | SPI interface serial data input of SD, with pull-up resister |
| 25 | 18 | SD_DO | OUT | SPI interface serial data output of SD |
| 25 | 18 | RST | OUT | Power on reset and external reset before enter into SD mode, active with high-level |
| 22～15 | No | D7～D0 | Bi-directional tri-state | Parallel 8-bit bi-directional data bus, with pull-up resistor |
| 18 | 13 | SCS | IN | Chip select input of SPI interface, active with low-level, with pull-up resistor |
| 20 | 14 | SCK | IN | Serial clock input of SPI, with pull-up resistor |
| 21 | 15 | SDI | IN | Serial data input of SPI interface, with pull-up resistor |
| 22 | 16 | SD0 | tri-state | Serial data output of SPI interface |
| 19 | No | BZ | OUT | Busy state output of SPI interface, active with high-level |
| 8 | No | A0 | IN | Address input of parallel, distinguish of command port with data port, with pull-up resistor, A0=1, write command or read status; A0=0 read/write data |
| 27 | No | PCS# | IN | Chip select input, active with low-level, with pull-up resistor |

| 4 | No | RD# | IN | Read strobe input of parallel, active with low-level. With pull-up resistor |
|---|---|---|---|---|
| 3 | No | WR# | IN | Write strobe input of parallel, active with low-level. With pull-up resistor |
| No | 3 | SPI# | IN | Interface configure input during internal reset, with pull-up reset |
| 5 | 4 | TXD | IN OUT | Interface configure input during internal reset, with pull-up reset Serial data output of Asynchronism serial interface after resetting |
| 6 | 5 | RXD | IN | Serial data input of Asynchronism serial interface, with pull-up resistor |
| 1 | 1 | INT# | OUT | Interrupt query output, active with low-level, with pull-up resistor |
| 24 | No | ACT# | Drain open OUT | Status output, active with low-level, with pull-up resistor. Inside USB-DEVICE mode, active with low-level under USB-DEVICE; In SD card Mode, SPI communication success status output; USB device connection state output under USB-HOST, active with low-level |
| 2 | 2 | RSTI | IN | Reset input external, active with high level, with pull-down resistor |

## 5. Command

The data in this manual has three types. Binary numbers are followed by a "B". Hexadecimal numbers are followed by an "H". Numbers without annotations are decimals.

The double word data (32-bit) with Litter-Endian means: the first is the lowest byte (bit-7 to bit-0), the following is lower byte (bit-15 to bit-8), the higher byte (bit-23 to bit-16), the end is the highest byte (bit-31 to bit-24).

Data stream means data block constituent by sequence bytes, the min long of data block is 0, and the max is 255.

The data in bracket of the input data and output data with the following table is the byte number of parameter, without bracket means one byte.

The MCU referred in this manual are basically applied to DSP or MCU/MPU/SCM and so on.

The USB Flash Drive referred in this manual contains USB Flash Drive, USB HD, USB Flash Memory, USB reader and so on.

The SD card referred in this manual contains SD, MMC card, HC-SD card (with high capacity), TF card and so on.

The manual mainly supplies commands of file manage control, uses to USB Flash Drive and SD card. Commands of assistant, executing basic affair and controlling transfer can refer to the second manual, please consult CH376DS2.PDF.

The function of CH372 is totally contained by CH376 chip, in this DataSheet, there is no description

about USB-DEVICE, the CH372DS1.PDF is the reference data.

| Code | Command name | Input data | Output data | Functions |
|------|-------------|-----------|-------------|-----------|
| 01H | GET_IC_VER | | Version number | Obtain chip and firmware version number |
| 02H | SET_BAUDRATE | Detach freq coef | (Wait for 1mS) Operation status | Set serial communication baud rate |
| | | Detach freq const | | |
| 03H | ENTER_SLEEP | | | Go to low-power and suspending |
| 05H | RESET_ALL | | (Wait for 35mS) | Execute hardware reset |
| 06H | CHECK_EXIST | Any data | Complementary operation accord bit | Test communication interface and working status |
| 0BH | SET_SD0_INT | Data 16H | | Set interrupt Mode of SD0 in SPI |
| | | Interrupt Mode | | |
| 0CH | GET_FILE_SIZE | Data 68H | File length (4) | Get the current file length |
| 15H | SET_USB_MODE | Mode code | (Wait for 10uS) Operation status | Configure the work mode of USB |
| 22H | GET_STATUS | | Interruption status | Get interruption status and cancel requirement |
| 27H | RD_USB_DATA0 | | Data length | Read data from current interrupt port buffer of USB or receive buffer of host port |
| | | | Data stream (n) | |
| 2CH | WR_USB_DATA | Data length | | Write data to transfer buffer of USB host |
| | | Data stream (n) | | |
| 2DH | WR_REQ_DATA | | Data length | Write requested data block to internal appointed buffer |
| | | Data stream (n) | | |
| 2EH | WR_OFS_DATA | Excursion address | | Write data block to internal buffer with appointed excursion address |
| | | Data length | | |
| | | Data stream (n) | | |
| 2FH | SET_FILE_NAME | Character string (n) | | Set the file name which will be operated |
| 30H | DISK_CONNECT | | Produce interruption | Check the disk connection status |
| 31H | DISK_MOUNT | | Produce interruption | Initialize disk and test disk ready |
| 32H | FILE_OPEN | | Produce interruption | Open file or catalog, enumerate file and catalog |

| 33H | FILE_ENUM_GO | | Produce interruption | Go on to enumerate file and catalog |
|-----|--------------|--|---------------------|------------------------------------|
| 34H | FILE_CREATE | | Produce interruption | Create file |
| 35H | FILE_ERASE | | Produce interruption | Delete file |
| 36H | FILE_CLOSE | Update or not | Produce interruption | Close the open file or catalog |
| 37H | DIR_INF0_READ | Catalog index number | Produce interruption | Read the catalog information of file |
| 38H | DIR_INF0_SAVE | | Produce interruption | Save catalog information of file |
| 39H | BYTE_LOCATE | Excursion byte number (4) | Produce interruption | Move the file pointer take byte as unit |
| 3AH | BYTE_READ | Request byte number (2) | Produce interruption | Read data block from current location take byte as unit |
| 3BH | BYTE_RD_GO | | Produce interruption | Continue byte read |
| 3CH | BYTE_WRITE | Request byte number (2) | Produce interruption | Write data block from current location take unit as unit |
| 3DH | BYTE_WR_GO | | Produce interruption | Continue byte write |
| 3EH | DISK_CAPACITY | | Produce interruption | Check disk physical capacity |
| 3FH | DISK_QUERY | | Produce interruption | Check disk space |
| 40H | DIR_CREATE | | Produce interruption | Create catalog and open or open the existed catalog |
| 4AH | SEG_LOCATE | Excursion fan number (4) | Produce interruption | Move file pointer from current location take fan as unit |
| 4BH | SEC_READ | Request fan number | Produce interruption | Read data block from current location take fan as unit |
| 4CH | SEC_WRITE | Request fan number | Produce interruption | Write data block from current location take fan as unit |
| 50H | DISK_BOC_CMD | | Produce interruption | Execute B0 transfer protocol command to USB Storage |
| 54H | DISK_READ | LBA fan address<br>Sector number | Produce interruption | Read physical fan from USB storage device |

| 55H | DISK_RD_GO | | Produce interruption | Go on reading operation of USB storage device |
|-----|------------|--------|--------|------------------------------|
| 56H | DISK_WRITE | LBA address | Produce interruption | Write physical fan to USB storage device |
|     |            | Sector number | | |
| 57H | DISK_WR_GO | | Produce interruption | Go on writing operation of USB storage device |

If the output data of command is operation state, please consult the following table.

| State code | State name | State explanation |
|------------|-----------|-------------------|
| 51H | CMD_RET_SUCCESS | Operation successful |
| 5FH | CMD_RET_ABORT | Operation failure |

"Produce interruption" in the table needs time to execute operation. CH376 request interruption to MCU after finishing command, MCU can read interrupter status as the operation code of the command. If the interrupt status is USB_INT_SUCCESS, means the operation is successful Many commands have return data (consult the CH376_CMD_DATA configuration in the CH376INC.H), read the return data through CMD_RD_USB_DATA0.

## 5.1. CMD_GET_IC_VER

This command can get chip and firmware version number, and the return byte data is version number. The bit 7 is 0, bit 6 is 1 and bit 5 to bit 0 is version number. If the return byte data is 41H, take the bit 7 and bit 6 out, the actual minimum version number is 01H.

## 5.2. CMD_SET_BAUDRATE

The command is used to set up the serial port communication baud-rate of CH376.CH376 works on serial communication mode, the default communication baud-rate of CH375 is composed of BZ/D4,SCK/D5 and SDI/D6 (consult the 6.4 of the DataSheet) after reset. And these pins are suspend in default means the baud-rate is 9600bps. Through the order adjusts serial communication baud-rate dynamically if the MCU supports upper communication speed. The command needs to input baud-rate detach frequency coefficient and detach frequency const.

Many detach frequency coefficient and detach frequency constant corresponding serial communication baud-rate is given below.

| Detach freq coef | Detach freq const | Serial interface communication baud- rate (bps) | Error |
|------------------|-------------------|--------------------------------------------------|-------|
| 02H | B2H | 9600 | 0.16% |
| 02H | D9H | 19200 | 0.16% |
| 03H | 98H | 57600 | 0.16% |
| 03H | CCH | 115200 | 0.16% |
| 03H | F3H | 460800 | 0.16% |
| 07H | F3H | 921600 | 0.16% |
| 03H | C4H | 100000 | 0% |
| 03H | FAH | 1000000 | 0% |
| 03H | FEH | 2000000 | 0% |
| 02H | Const | Formula: 750000/(256-const) | |
| 03H | Const | Formula: 6000000/(256-const) | |

Usually, the active of set baud-rate costs within 1mS, then the CH376 output operation state according the new-setting communication baud-rate. Consequently, the MCU regulates itself communication baud-rate after giving the order out.

## 5.3. CMD_ENTER_SLEEP

The order make CH376 enter into low power. The clock of CH376 stops oscillating to economize power when keeping low power. Once detection the signal of USB bus signal (USB host start transfer or USB device plug or unplug) or MCU writes new command to CH376 (the command without input data, such as CMD_GET_IC_VER or CMD_ABORT_NAK) the CH376 can exit the low-power state. For SPIN communication mode, if the SCS chip select is active, the CH376 will exit low-power state. So if the MCU transfer the CMD_ENTER_SLEEP command, make the SCS chip select inactive at once.

In general, the time to wake up CH376 from low-power state to work normally is several milliseconds. The CH376 will send USB_INT_WAKE_UP interrupter while totally recovering to work state.

## 5.4. CMD_RESET_ALL

The Command makes the CH376 reset through hardware. Usually, the hardware reset finishes within 35mS. For parallel communication mode, the time is 1mS.

## 5.5. CMD_CHECK_EXIST

The Command is used to check communication interface and work status to examine the CH376.when using the order, one data at random needs to input .The output data is contrary to the input data if the CH376 is working normally. For example, the output data is A8H while the input data is 57H. for parallel communication mode of CH376, without receive any command after reset, the data is 00H from parallel.

## 5.6. CMD_SET_SD0_INT

The Command is used to set SD0 pin interrupt mode of SPI interface. Input 16H at first, then input the new interrupt mode. There are two interrupt mode: 10H means forbid SD0 used as interrupt output, when SCS chip select inactive, it is tri-state output forbid, so it is easy to share SPI bus of SPI with other device; 90H set SD0 is output status, when SCS chip select is inactive, it can be used as interrupt output, equal with INT# pin, MCU can query interrupt request status.

## 5.7. CMD_GET_FILE_SIZE

The Command is used to get file length, byte number. Input 68H for the command, and the output is the file length of current file. The length is double data (32-bit) with 4 byte, and the lower is in front.

If needing to set new file length, please consult the second DataSheet CMD_WRITE-VAR32 set VAR_FILE_SIZE.

## 5.8. CMD_SET_FILE_SIZE

The Command is used to set current file length, byte number. Input 68H at first, then input the new file length. The length is double data (32-bit) with 4 byte, and the lower is in front.

Only modify the file length variable in CH376EMS memory. After execute CMD_FILE_CLOSE and so on command, the file (in USB Storage or SD card) length will update

## 5.9. CMD_SET_USB_MODE

The command can determine the work mode of USB. The mode code may input.

The code of 00H means switch to invalid USB-DEVICE mode (the default mode after power-up or

reset)

The code of 01H means switch to valid USB-DEVICE, peripheral firmware mode. (not support serial connection)

The code of 02H means switch to valid USB-DEVICE, inner firmware mode.

The code of 03H means switch to SD host mode, manage and storage/get file in SD card.

The code of 04H means switch to invalid USB-HOST mode.

The code of 05H means switch to valid USB-HOST, non-generate SOF package.

The code of 06H means switch to valid USB-HOST, produce SOF package automatically.

The code of 07H means switch to valid USB-HOST, and reset USB bus.

The handbook of CH376 explains the USB-DEVICE/SLAVE manner. The USB-DEVICE mode of CH376 is totally compatible with CH372.

Un-automatically detect the USB device connection is defined as invalid, under USB-HOST mode, exterior MCU is needed to detect. Validation refers to examine USB device connection automatically. There is a piece of information to peripheral MCU when the USB device attachment or unlatch. The CH376 timing sends SOF to attachment USB device automatically while switch to mode code of 06H. In general, the mode code of 07H supplies USB bus reset state for attached USB device. USB bus reset won't finish until it changes to other mode. Advice to use mode 5 when there is no USB device, enter mode 7 and then switch mode 6 after insert USN device.

Ordinarily, the time of set USB work mode is within 10uS and outputs operation station after setting.

## 5.10. CMD_GET_STATUS

The order can obtain interrupt state of CH376 and notice CH376 to cancel the interrupt requests. MCU receives interrupt status, analyzes interrupt and deal with interrupt after the CH376 sends interrupt to MCU.

| The byte of interruption state | Sort of interruption |
|---|---|
| 00H~0FH | The interruption state of USB-device refer to the CH372 handbook |
| 10H~1FH | Interruption state in SD card or USB-HOST mode |
| 20H~3FH | The failure operation of USB-HOST used to analyze reasons |
| 40H~4FH | File system notice code in SD card or USB host file mode |
| 80H~BFH | File system error code in SD card or USB host file mode |

The following is common interruption state of SD card or USB-HOST mode.

| State byte | State name | Description of analyzing interruption state |
|---|---|---|
| 14H | USB_INT_SUCCESS | Success of SD card or USB transaction or transfer operation |
| 15H | USB_INT_CONNECT | Detection of USB device attachment |
| 16H | USB_INT_DISCONNECT | Detection of USB device detachment |
| 17H | USB_INT_BUF_OVER | Data error or Buffer overflow |
| 18H | USB_INT_USB_READY | USB device has initialized (appointed USB address) |
| 1DH | USB_INT_DISK_READ | Read operation of storage device |
| 1EH | USB_INT_DISK_WRITE | Write operation of storage device |
| 1FH | USB_INT_DISK_ERR | Failure of USB storage device |

The following is operation failure status of USB-HOST mode used to analyze the reasons induce failure.

| The byte of interrupt status | Name | Description of analyzing interruption status |
|---|---|---|
| Bit 7～bit 6 | Reserved | 00 |
| Bit 5 | Flag | 1,failure flag |
| Bit 4 | Synchronous sign of IN transaction | The data may be invalid when the bit is 0 |
| Bit 3～bit 0 | Return of USB device when failure | 1010,device return NAK |
| | | 1110,device return STALL |
| | | XX00, device return over time, no return |
| | | Other data is PID of device return |

The following is file system notice code and error code of SD card or USB-HOST mode

| State byte | State name | Description of analyzing interruption state |
|---|---|---|
| 41H | ERR_OPEN_DIR | Open directory address is appointed |
| 42H | ERR_MISS_FILE | File doesn't be found which address is appointed, maybe the name is error |
| 43H | ERR_FOUND_NAME | Search suited file name, or open directory according the request, open file in actual |
| 82H | ERR_DISK_DISCON | Disk doesn't connect, maybe the disk has cut down |
| 84H | ERR_LARGE_SECTOR | Fan is too big, only support 512 bytes |
| 92H | ERR_TYPE_ERROR | Disk partition doesn't support, re-prartition by tool |
| A1H | ERR_BPB_ERROR | Disk doesn't format, or parameter is errot, re-formate by WINDOWS with default parameter |
| B1H | ERR_DISK_FULL | File in disk is full, spare space is too small |
| B2H | ERR_FDT_OVER | Many file in directory, no spare directory, clean up the disk, less than 512 in FAT12/FAT16 root directory |
| B4H | ERR_FILE_CLOSE | File is closed, re-open file if need |

## 5.11. CMD_RD_USB_DATA0

The command reads data block from current interrupt endpoint buffer of USB or host endpoint. The length of data block is read at first, i.e. the byte number of following data stream. For file read/write, the data block virtual value is from 0 to 255. The virtual value of data block length is from 0 to 64 for USB bottom transfer.If the length is not Zero, MCU reads following data one by one from CH376.

## 5.12. CMD_WR_HOST _DATA

The command means writing data block to output buffer of USB host endpoint. Also, the data is first written is length of data block, i.e. the byte number of following data stream. The virtual value of data block length is from 0 to 64.When the length isn't zero, MCU writes the following data one by one to CH376.

## 5.13. CMD_WR_REQ _DATA

The command means writing data block (request by CH376) to internal appointed buffer. Also, the data is first written is length of data block, i.e. the byte number of following data stream. For file read/write, the data block virtual value is from 0 to 255. The virtual value of data block length is from 0 to 64 for USB bottom transfer.If the length is not Zero, MCU reads following data one by one from CH376.

## 5.14. CMD_WR_OFS_DATA

The command means writing data block to internal buffer with appointed excursion address. The first input data is excursion address (the beginning address of internal buffer adds excursion address is the start address), and the following data is length of data block, i.e. the byte number of following data stream. the data block virtual value is from 0 to 32, and the sum of excursion address adds data block length is less than 32. If the length is not Zero, MCU reads following data one by one from CH376.

## 5.15. CMD_SET_FILE_NAME

The command used to set file name or directory name. The input data with 0 as end, and the length is less than 14 characters. For the file with many sub-directory, decompose as many sub-directory and one file name, set name by many times and open from the root directory. When open file is error, return to the root directory to open again.

The manner of file name is the same with short file name in DOS system, no need disk symbol and colon. The "/" is equal with "\", "/" is recommended. All the characters are capital letter, number or Chinese, and some special symbol. The length is less than 11 characters, the main file name is no more than 8 characters, and the expend file name is no more than 3 characters. If there is expend file name, use decimal to distinguish. Consult EXAM11.

No character in string (the end character is 0, the following is the same), initialize file system, open no file;

Only one "/" or "\", open root directory;

The first character is "/" or "\", and the following is file name, file in root directory;

File name as character, means file in current directory.

For example, for FILENAME.EXT in root directory, use "/FILENAME.EXT\0" to set, 14 characters in totally, "\0" means 0 by C language, as end character."/" means in root directory. Use "\\" as root directory in C language.

Another, the file has three sub-directory \YEAR2004\MONTH05.NEW\DATE18\ADC.TXT, open as following:

① use "/YEAR2004\0" to set file name (directory name), use CMD_FILE_OPEN to open the first directory;

② use "MONTH05.NEW\0" to set file name (directory name), use CMD_FILE_OPEN to open the second directory;

③ use "DATE18\0" to set file name (directory name), use CMD_FILE_OPEN to open the third directory;

④ use "ADC.TXT\0" to set file name (directory name), use CMD_FILE_OPEN to open the final file;

## 5.16. CMD_DISK_CONNECT

The command can examine disk ready or not, doesn't support SD card. In USB-HOST mode, the command can query the disk connection at any time, after finish the command CH376 request interrupt to MCU, if the operation state is USB_INT_SUCCESS, means the disk or USB device is connected.

## 5.17. CMD_DISK_MOUNT

The command initializes disk and test disk ready or not.. For attached USB storage device or SD card, use this command to initialize, then execute file operation. Execute many initialization for some USB storage device, they can return USB_INT_SUCCESS. During file operation, the command can test disk ready or not at any time.

Execute CMD_DISK_MOUNT at the first time, if the interrupt statue is USB_INT_SUCCESS, use

CMD_RD_USB_DATA0 to get data, the data is always 36 bytes, contain character of USB storage device, and information about manufacture and product.

## 5.18. CMD_FILE_OPEN

The command used to open file or directory, enumerate file and directory.

Open the file, and then can read/write file. Before open file command, set file name which will be open via CMD_SET_FILE_NAME.

If the file has long directory, the path is long, open layer by layer from root directory. Open the first sub-directory, and then the second sub-directory, until open the file. Notice that the open must from root directory, it means that the first character is "/" or "\", and then the first character isn't "/" or "\".

If open the directory successful, return interrupt status is ERR_OPEN_DIR, the file length is invalid, it is 0FFFFFFFFH.

If open the file successful, return interrupt status is USB_INT_SUCCESS, the file length is valid.

If doesn't find the appointed file or directory, return interrupt status is ERR_MISS_FILE.

For example:

Open the file \TODAY1.TXT in root directory, process is as below:

① use string "/TODAY1.TXT" via CMD_SET_FILE_NAME to set file name;

② use CMD_FILE_OPEN to open file.

If open the file has three sub-directorys \YEAR2004\MONTH05.NEW\DATE18\ADC.TXT, process is as below:

① use string "/YEAR2004\0" via CMD_SET_FILE_NAME to set sub-directory name;

② use CMD_FILE_OPEN to open the first sub-directory, after open directory, execute CMD_GET_FILE_SIZE will get invalid file length 0FFFFFFFFH;

③ use string "MONTH05.NEW\0" via CMD_SET_FILE_NAME set sub-directory name;

④ use CMD_FILE_OPEN to open the second sub-directory;

⑤ use string "DATE18\0" via CMD_SET_FILE_NAME set sub-directory name;

⑥ use CMD_FILE_OPEN to open the third sub-directory;

⑦ use string "ADC.TXT\0" via CMD_SET_FILE_NAME set sub-directory name;

⑧ use CMD_FILE_OPEN to open the final file, if open the file, execute CMD_GET_FILE_SIZE can get the file length.

Initialize file system, but don't open file, the process is as below:

① use string "\0" via CMD_SET_FILE_NAME set file name;

② execute CMD_FILE_OPEN, initialize file system (if has initialized, return). Open root directory (such as deal with long file name), process is as following:

① use string "/\0" via CMD_SET_FILE_NAME set file name;

② execute CMD_FILE_OPEN, then open the root directory (after using, CMD_FILE_CLOSE to close).

## 5.19. CMD_FILE_ENUM_GO

The command used to go on enumerating file and directory.

If need to search and query file, process is as following:

① use * to replace total or some character, there is no character after *, use CMD_SET_FILE_NAME to set file name which contain *, such as string "/*\0"means enumerate the total file or directory in root directory. String "USB*\0" means enumerate current directory with "USB" as head file or directory, such as "USB.TXT","USB1234","USB","USBC.H" and so on, but not contain "XUSB","U.SB", "U2SB", "MY.USB" and so on;

② via CMD_FILE_OPEN to enumerate file and directory;

③ CH376 will compare each file name, when find one file accord with requirement, generate one interrupt, interrupt status is USB_INT_DISK_READ, query MCU to read data from Ch376;

④ MCU via CMD_RD_USB_DATA0 to read data, analysis or save, the data is about FAT file directory (consult FAT_DIR_INF0 in CH376INC.H);

⑤ MCU send CMD_FILE_ENUM_GO to notify Ch376 to go on enumerating;

⑥ CH376 will go on to compare file name, if find the required file, turn to ③, or turn to the next;

⑦ CH376 generate one interrupt to MCU, interrupt status is ERR_MISS_FILE, means there is no required file, the enumerate operation has finished.

In the process ④, MCU can analysis the FAT_DIR_INFO to determine, or record the relative information and deal with it after the enumerate operation has finished. MCU can distinguish file or sub-directory via DIR_Attr, via DIR_Name file name unit to compare file name. for example, file expand name DIR_Name[8],[9],[10] to compare with "XLS", so filtrate the EXCEL type file.

## 5.20. CMD_FILE_CREATE

The command used to create file, if the file has existed, delete and then create.

Before create file, via CMD_SET_FILE_NAME to set the file name, the same with CMD_FILE_OPEN command, but doesn't support *. If there is the same file name, delete the file and then create. If don't want to delete file, via CMD_FILE_OPEN to certain file whether exist. The default date and time is 2004/1/1 0:0:0, and the file length is 1. Modify these information via CMD_DIR_INFO_READ and CMD_DIR_INFO_SAVE.

## 5.21. CMD_FILE_ERASE

The command used to delete file, if the file has open, delete directly; or the file will be open at first and then delete. For the sub-directory, it must be open at first.

For common file, the process is as following:

① affirm the former file or directory has closed, or it will be delete, not influenced by the process ②;

② via CMD_SET_FILE_NAME to set the file name which will be delete, don't support * wildcard character;

③ via CMD_FILE_ERASE to open file and delete.

For the sub-directory, consult the following process:

① for sub-directory, delete all the file and the lower sub-directory;

② via CMD_SET_FILE_NAME to set the sub-directory name, don't support * wildcard character;

③ via CMD_FILE_OPEN to open sub-directory;

④ via CMD_FILE_ERASE to delete sub-directory which has been open in the process ②.

## 5.22. CMD_FILE_CLOSE

The command used to close the file or directory which has been open. Need one input data, to notify whether updata file length, 0 means forbid update file length, 1 means automatically update file length.

After read/write file, close file. For root directory, close file is necessary. For the common file read operation, close file is optional. For common file write operation, close file, and select update file length via CH376 automatically.

If via CMD_SEC_LOCATE,CMD_SEC_READ or CMD_SEC_WRITE to read/write file take sector as unit, then the file length automatically update by CH376 is take sector as unit to calculate, the file length is multiple of 512. if wish the file length is not multiple ofsector, via CMD_SET_FILE_SIZE to modify file length before close file, or via CMD_DIR_INFO_READ and CMD_DIR_INFO_SAVE to modify file

information.

If via CMD_BYTE_LOCATE,CMD_BYTE _READ or CMD_BYTE _WRITE to read/write file take byte as unit, then the file length automatically update by CH376 is take byte as unit to calculate, so can get the proper length.

## 5.23. CMD_DIR _INFO_READ

The command used to read file directory information, i.e. FAT_DIR_INFO configuration. Need to input one data to appoint the index number which directory information configuration will be read. The range of index number is from 00H~0FH. 0FFH correspond the current open file. The command only read to EMS memory buffer, then MCU can via CMD_RD_USB_DATA0 to read data.

After open one file, CH376 will get border 16 file directories information from USB storage or SD card to save to EMS memory. MCU can appoint index number 0~15 separate corresponding FAT_DIR_INFO configuration, also appoint index number 0FFH to get current open file FAT_DIR_INFO configuration, to analysis file data, time, length and attribute etc. information.

## 5.24. CMD_DIR _INFO_SAVE

The command used to save file directory information. The command will refresh 16 file directories information to save to USB storage or SD card. The following is modifying file directory information:

① if the file has open, turn to the ②, or via CMD_SET_FILE_NAME and CMD_FILE_OPEN to open file;

② via CMD_DIR_INFO_READ to read the current file or border file FAT_DIR_INFO configuration to buffer;

③ via CMD_RD_USB_DATA0 to read data from buffer, if no need to modify, the process is finished;

④ need modify, via CMD_DIR_INFO_READ to read FAT_DIR_INFO configuration to buffer again;

⑤ via CMD_WR_OFS_DATA to write modified data to buffer with appointed excursion address, such as, write new file data of two bytes to excursion address 18H (DIE_WrtData file data unit in configuration)

⑥ via CMD_DIR_INFO_SAVE to save the modified file directory information to USB storage or SD card.

## 5.25. CMD_BYTE_LOCATE

The command is used to move file pointer byte as unit. This command need to input offset byte number, the offset byte number is double word data with lower byte in front, 4 bytes. If the interrupt status is USB_INT_SUCCESS after command complete, CMD_RD_USB_DATAO to get current file pointer to the absolute linear sector number corresponding to LBA (lower byte in front, 4 bytes, 32-bit double-word data), if come to the end of the file, the value is 0FFFFFFFFH.

File are new or have been re-opened, the current file pointer is 0, move the current file pointer, generally used to read/write data from the appointed location. For example, MCU want to skip the first 158 bytes of the file and then read/write data, use CMD_BYTE_LOCATE command with parameter 158 as offset bytes, execute command, the read/write data will start from the 158 bytes. For the write operation, if the MCU prepare to add data at the end, not attempting to influence the front data, appoint much offset byte, such as 0FFFFFFFFH, move the file pointer to the end to add data.

## 5.26. CMD_BYTE_READ

## 5.27. CMD_BYTE_RD_GO

The command is used read data block from current location in bytes, CMD_BYTE_RD_GO is used to

continue byte read operation. Read successful, CH376 automatically synchronize move file pointer, so the next read/write operation can follow this location. This command need to input byte number request to read, the request is based on lower byte in front (Little-Endian) of two bytes 16-bit data.

A complete byte read is usually start by a CMD_BYTE_READ command, composed by many interrupt notification and many data block read and several CMD_BYTE_RD_GO. The complete byte read steps are as follows:

① open file and confirmed at the appropriate location (file pointer);

② MCU transfer CMD_BYTE_READ and input byte number which is request, begin read operation;

③ CH376 calculate the spare length from the current file pointer to the end of location. If the current file pointer is place the end of file, or the spare length is 0, then finish the read operation and interrupt notification MCU, interrupt status is USB_INT_SUCCESS. Otherwise calculate the bytes number which is allowed to read through request byte number, file remain length, internal buffer, and subtracting the number from request byte number to get remain request byte number, move the file pointer at the same time, then notify MCU, interrupt status is USB_INT_DISK_READ;

④ MCU analysis interrupt status, if it is USB_INT_DISK_READ, read data block through CMD_RD_USB_DATA0 and continue; if it is USB_INT_SUCCESS, turn to step ⑥;

⑤ MCU transfer CMD_BYTE_RD_GO to order CH376 continue read, CH376 automatically turn to step ③;

⑥ file is end or the request read bytes has been read, the entire read operation is finished.

MCU will be interrupted several times to obtain notice of the data block length and sun the several data block length, compare the sum with requested byte number, if the sum is less than the requested byte number, means the pointer has reach to the end.

## 5.28. CMD_BYTE_WRITE

## 5.29. CMD_BYTE_WR_GO

CMD_BYTE_WRITE used to write data block from current location in bytes, CMD_BYTE_WR_GO is used to continue write operate. Write successful, CH376 will automatically synchronize move file pointer, so the next read/write operation can follow this location. This command need to input byte number request to read, the request is based on lower byte in front (Little-Endian) of two bytes 16-bit data. When the requested byte number is 0, only to refresh file size.

A complete byte write is usually start by a CMD_BYTE_WRITE command, composed by many interrupt notification and many data block write and several CMD_BYTE_WR _GO. The complete byte read steps are as follows:

① open file and confirmed at the appropriate location (file pointer);

② MCU transfer CMD_BYTEWRITE and input byte number which is request, begin write operation;

③ CH376 check requested byte number, 0, refresh file size, save file length to USB storage or SD card, output interrupt status USB_INT_SUCCESS after finishing, turn to the step ⑤;

④ CH376 calculate the spare length. 0 then finish the write operation and interrupt notify MCU, interrupt status is USB_INT_SUCCESS. Otherwise calculate the bytes number which is allowed to write through request byte number, file remain length, internal buffer, and subtracting the number from request byte number to get remain request byte number, move the file pointer at the same time, if it is add data, need to update file length, then notify MCU, interrupt status is USB_INT_DISK_WRITE;

⑤ MCU analysis interrupt status, if it is USB_INT_DISK_WRITE, obtain allowed byte number through CMD_WR_REQ_DATA, write data block and continue; if it is USB_INT_SUCCESS, turn to step ⑦;

⑥ MCU transfer CMD_BYTE_WR_GO to order CH376 continue write, CH376 automatically turn to step ④;

⑦ request write byte number has totally been written, the entire write operation is finished.

MCU will be interrupted several times to obtain notice of the data block length and sun the several data block length, compare the sum with requested byte number, if the sum is less than the requested byte number, means the pointer has reach to the end.

If add data directly to the end of the file, or during write operation, automatically moved file pointer has beyond original file location, CH376 will automatically update the file length. The write operation is complete, and in short period of time there is no write operation, MCU should notify CH376 refresh file length of memory to USB storage or SD card in two ways: similar to the above step ② and ③ write 0 length of data; implement CMD_FILE_CLOSE command and allow update length.

## 5.30. CMD_DISK_CAPACITY

The command is used to check disk physical capacity, support USB storage and SD card. If the interrupt status is USB_INT_SUCCESS after execute command, get disk physical capacity via CMD_RD_USB_DATA0, i.e. total sector number, the capacity is double word data with lower byte in front, 4 bytes 32 bits, if multiply 512, get the physical capacity in bytes.

## 5.31. CMD_DISK_QUERY

The command is used to check disk space information, including the remaining space and file system type. If the interrupt status is USB_INT_SUCCESS after executing command, get logical disk sectors number (lower byte in front 4 bytes 32 bit double word data), spare sectors number of current logical disk (lower byte in front 4 bytes 32 bit double word data), FAT file system type of logical disk (consult CH376_CMD_DATA configuration in CH376INC.H) through CMD_RD_USB_DATA0.

## 5.32. CMD_DIR_CREATE

The command is used to create sub-directory and open it. If the sub-directory has existed, open directly, only support the first sub-directory, consult EXAM9 can support many layer sub-directory.

Before create directory command, set directory name via CMD_SET_FILE_NAME, the same format as CMD_FILE_CREATE. If there is a common file with the same name, the interrupt status is ERR_FOUND_NAME; if create sub-directory is successful, or open the existed sub-directory, the interrupt status is USB_INT_SUCCESS. The data, time and other information about new sub-directory is the same with the file create by CMD_FILE_CREATE, the modify method is the same, only the file attribution is ATTR_DIRECTORY, and the file length is 0 (according FAT criterion, file length of sub-directory must be 0).

## 5.33. CMD_SEC_LOCATE

The command is used to move file pointer in sectors, not support SD card. The command need to input the offset sector number, lower byte in front 4 bytes 32 bit double word data. If the interrupt status is USB_INT_SUCCESS after command complete, CMD_RD_USB_DATAO to get current file pointer to the absolute linear sector number corresponding to LBA (lower byte in front, 4 bytes, 32-bit double-word data), if come to the end of the file, the value is 0FFFFFFFFH.

File are new or have been re-opened, the current file pointer is 0, move the current file pointer, generally used to read/write data from the appointed location. For example, MCU want to skip the first 18 sectors of the file and then read/write data, use CMD_SEC_LOCATE command with parameter 18 as offset sectors, execute command, the read/write data will start from the 18 sectors. For the write operation, if the

MCU prepare to add data at the end, not attempting to influence the front data, appoint much offset byte, such as 0FFFFFFFFH, move the file pointer to the end to add data.

## 5.34. CMD_SEC_READ

The command is used to read data block in sectors from current location, not support SD card. Command successful, CH376 will automatically synchronize move file pointer, so the next read/write operation can follow this location. This command need to input one data, to appoint request read sectors, the value is from 0 to 255. If the interrupt status is USB_INT_SUCCESS after command complete, CMD_RD_USB_DATAO to get return data with 8 bytes: the first byte is allowed read sectors, if it is 0, means the file pointer has moved to the end of file; the following 4 bytes is the absolute linear sector number LBA (lower byte in front, 4 bytes, 32-bit double-word data).

A complete byte read is usually get parameter information by CMD_SEC_READ, and start by a CMD_DISK_READ command, composed by many interrupt notifications and many data block read and several CMD_DISK _RD_GO. The complete sector read steps are as follows:

① open file and confirmed at the appropriate location (file pointer);

② MCU transfer CMD_SEC_READ and input sector number which is request, begin read operation;

③ CH376 calculate the parameter and inform the MCU, the interrupt status is USB_INT_SUCCESS;

④ MCU read parameter by CMD_RD_USB_DATAO, the LBA (sector beginning) and allow read sector number, if the allow read sector number is 0, mean the end of the file, turn to the step ⑨;

⑤ MCU transfer CMD_DISK_READ, and input the upper parameter to start read operation;

⑥ Each sector is devised as 8 data blocks with 64 bytes, if the 8 data blocks have been disposal, finish read operation and notice MCU, the interrupt status is USB_INT_SUCCESS; otherwise CH376 read one 64-byte data block from USB storage, and then interrupt notice MCU request read data block, the interrupt status is USB_INT_DISK_READ;

⑦ MCU analysis interrupt status, if it is USB_INT_DISK_READ, read data block through CMD_RD_USB_DATAO and continue; if it is USB_INT_SUCCESS, turn to step ⑨;

⑧ MCU transfer CMD_DISK_RD_GO to order CH376 continue read, CH376 automatically turn to step ⑥;

⑨ The allow read sectors has been read, the entire read operation is finished.

## 5.35. CMD_SEC_WRITE

The command is used to write parameter information of data block in sectors from current location, not support SD card. Command successful, CH376 will automatically synchronize move file pointer, so the next read/write operation can follow this location. This command need to input one data, to appoint request write sectors, the value is from 0 to 255. if the value is 0, only refresh file length. If the interrupt status is USB_INT_SUCCESS after command complete, CMD_RD_USB_DATAO to get return data with 8 bytes: the first byte is allowed write sectors; the following 4 bytes is the absolute linear sector number LBA (lower byte in front, 4 bytes, 32-bit double-word data).

A complete byte read is usually get parameter information by CMD_SEC_WRITE, and start by a CMD_DISK_WRITE command, composed by many interrupt notifications and many data block write and several CMD_DISK _WR _GO. The complete sector read steps are as follows:

① open file and confirmed at the appropriate location (file pointer);

② MCU transfer CMD_SEC_WRITE and input sector number which is request;

③ CH376 check requested sector number, 0, refresh file size, save file length to USB storage, output interrupt status USB_INT_SUCCESS after finishing, otherwise CH376 calculate the parameter and notify MCU, the interrupt status is USB_INT_SUCCESS;

④ MCU read parameter by CMD_RD_USB_DATAO, the LBA (sector beginning) and allow read sector number, if the allow read sector number is 0, mean the end of the file, turn to the step ⑨;

⑤ MCU transfer CMD_DISK_WRITE, and input the upper parameter to start write operation;

⑥ Each sector is devised as 8 data blocks with 64 bytes, if the 8 data blocks have been disposal, finish write operation and notice MCU, the interrupt status is USB_INT_SUCCESS; otherwise interrupt notice MCU request write data block, the interrupt status is USB_INT_DISK_WRITE;

⑦ MCU analysis interrupt status, if it is USB_INT_DISK_WRITE, write one 64-byte data block through CMD_WR _HOST _DATA and notify CH376 continue to write; if it is USB_INT_SUCCESS, turn to step ⑨;

⑧ MCU transfer CMD_DISK_WR_GO to order CH376 continue write, CH376 write the upper data block to USB storage and automatically turn to step ⑥;

⑨ The allow write sectors has been written, the entire write operation is finished.

If add data directly to the end of the file, or during write operation, automatically moved file pointer has beyond original file location, CH376 will automatically update the file length. The write operation is complete, and in short period of time there is no write operation, MCU should notify CH376 refresh file length of memory to USB storage in two ways: similar to the above step ② and ③ write 0 length of data; implement CMD_FILE_CLOSE command and allow update length.

## 5.36. CMD_DISK_BOC_CMD

The command is used to execute BulkOnly transfer protocol command to USB storage. Before this command, MCU must write CBW package to CH376 through CMD_WR_HOST_DATA (consule BULK_ONLY_CBW configuration in CH376INC.H), after finishing the command, CH376 will request interrupt to MCU, if the interrupt status is USB_INT_SUCCESS, the command is successful. For the operation has return data, get return data via CMD_RD_USB_DATAO.

## 5.37. CMD_DISK_BOC_CMD

## 5.38. CMD_DISK_RD_GO

CMD_DISK_READ is used to read physical sector from USB storage, CMD_DISK_RD_GO is used to continue execute read operation from USB storage, not support SD card.

CMD_DISK_READ need two groups parameter: sector start address with 4-byte and sector number with 1-byte, the sector start address is LBA, lower byte in front, 4 bytes, 32-bit double word data. The command need 5 input data: the lowest byte of LBA address, the lower byte of LBA, higher byte of LBA, the highest byte of LBA, sector number. This command can read any data from 1 to 255 in USB storage.

A complete physical sector read is usually start by a CMD_DISK_READ command, composed by many interrupt notification and many data block read and several CMD_DISK_RD_GO. The complete byte read steps are as follows:

① MCU transfer CMD_DISK_READ command and appoint LBA and sector number, start read operation;

② Each sector is devised as 8 data blocks with 64 bytes, if the 8 data blocks have been disposal, finish read operation and notice MCU, the interrupt status is USB_INT_SUCCESS; otherwise CH376 read one 64-byte data block from USB storage and then interrupt notice MCU request read data block, the interrupt status is USB_INT_DISK_READ;

③ MCU analysis interrupt status, if it is USB_INT_DISK_READ, read data block through CMD_RD _USB _DATAO and continue; if it is USB_INT_SUCCESS, turn to step ⑤; if it is USB_INT_DISK_ERR, means the operation is error, turn to the step ⑤, try again if need;

④ MCU transfer CMD_DISK_RD_GO to order CH376 continue read, CH376 automatically turn to step ②;

⑤ The appointed read sectors has been read, the entire read operation is finished.

If MCU transfer DISK_READ command to read one sector, normally, MCU will receive nine interrupt, the front 8 interrupts request MCU to get data, the final interrupt is return operation status. If read 4 sectors, normally, receive 33 interrupts, the front 32 interrupt request MCU to get data. If the read operation is failure in the process, MCU will receive USB_INT_DISK_ERR in advance, so the read operation is finished in advance.

## 5.39. CMD_DISKWRIT

## 5.40. CMD_DISK_WR_GO

CMD_DISK_WRITE is used to write physical sector to USB storage, CMD_DISK_WR_GO is used to continue execute write operation to USB storage, not support SD card.

CMD_DISK_WRITE need two groups parameter: sector start address with 4-byte and sector number with 1-byte, the sector start address is LBA, lower byte in front, 4 bytes, 32-bit double word data. The command need 5 input data: the lowest byte of LBA address, the lower byte of LBA, higher byte of LBA, the highest byte of LBA, sector number. This command can write any data from 1 to 255 in USB storage.

A complete physical sector write is usually start by a CMD_DISK_WRITE command, composed by many interrupt notification and many data block write and several CMD_DISK_WR_GO. The complete steps are as follows:

① MCU transfer CMD_DISK_WRITE command and appoint LBA and sector number, start write operation;

② Each sector is devised as 8 data blocks with 64 bytes, if the 8 data blocks have been disposal, finish write operation and notice MCU, the interrupt status is USB_INT_SUCCESS; otherwise interrupt notice MCU request write data block, the interrupt status is USB_INT_DISK_WRITE;

③ MCU analysis interrupt status, if it is USB_INT_DISK_WRITE, write one 64-byte data block through CMD_WR_HOST_DATA and continue; if it is USB_INT_SUCCESS, turn to step ⑤; if it is USB_INT_DISK_ERR, means the operation is error, turn to the step ⑤, try again if need;

④ MCU transfer CMD_DISK_WR_GO to order CH376 continue write, CH376 write the above data to USB storage and automatically turn to step ②;

⑤ The appointed write sectors has beenwritten, the entire write operation is finished.

If MCU transfer DISK_WRITE command to write one sector, normally, MCU will receive nine interrupts, the front 8 interrupts request MCU to offer data, the final interrupt is return operation status. If write 4 sectors, normally, receive 33 interrupts, the front 32 interrupt request MCU to offer data. If the write operation is failure in the process, MCU will receive USB_INT_DISK_ERR in advance, so the write operation is finished in advance.

# 6. Functions Description

## 6.1 MCU Communication Interface

It supports three kinds of communication interface between CH376 and MCU: 8-bit parallel interface, SPI synchronous serial interface, UART interface. When chip power-on reset, CH376 will sample WR#, RD#, PCS#, A0, RXD, TXD pin state, according to the configuration state of these pins combination selection communication interface, consult to the following table (in the table X denote not care about this bit, 0 denote low-level, 1 denote high-level or un-connection).

| WR# pin | RD# pin | PCS# pin | AO pin | RXD pin | TXD pin | choose communication interface |
|---------|---------|----------|--------|---------|---------|--------------------------------|
| 0 | 0 | 1 | 1 | 1 | 1 | SPI interface |
| 1 | 1 | 1 | 1 | 1 | 1 | UART interface |
| 1 | 1/X | 1/X | X | 1 | 0 | 8-bit parallel port |
| other state | | | | | | CH376 does not work, RST pin output high-level always |

There are two communication interfaces between CH376T and MCU: SPI synchronous serial interface, UART interface. When chip power-on reset, CH376T will sample SPI state, when SPI# is low-level, select SPI interface, SPI# is high-level select UART interface.

INT# pin of CH376 output interrupt request is active with low-level in default, it can be connected to the MCU interrupt input pin or common input pin, MCU can get interrupt request of CH376 by using interrupt mode or query mode. To save pins, MCU can not connect to INT# pin of CH376, but get interrupt by other ways.

## 6.2. Parallel Interface

Parallel signal line includes: 8-bit bidirectional data bus D7~D0, read strobe input pins RD#, write strobe input pin WR#, chip-select input pin PCS# and address input pin A0. PCS# of CH376 driven by the address decoding circuitry, use for choosing device when MCU has more than one peripheral devices. Using passive parallel interface, CH376 can be easily attached to all kinds of 8-bit MCU, DSP, MCU system bus, and can co-exist with a number of peripheral devices.

For the MCU similar to the Intel parallel timing, RD# pin and WR# pin of CH376 can be connected to the MCU read strobe output pin and write strobe output pin respectively. For the MCU similar to the Motorola parallel timing, RD# pin of CH376 should be connected to low-level, and WR # pin connected to read/write direction output pin R/-W of MCU.

The following table shows the truth table of parallel port I/O operations (in the table X denote not care about this bit, Z denote CH376 tri-state disabled).

| PCS# | WR# | RD# | A0 | D7-D0 | The actual operation of CH376 |
|------|-----|-----|----|-------|-------------------------------|
| 1 | X | X | X | X/Z | Not selected CH376, without any action |
| 0 | 1 | 1 | X | X/Z | Although selected, but no operation, without any action |
| 0 | 0 | 1/X | 1 | Input | Write command to CH376 command port |
| 0 | 0 | 1/X | 0 | Input | Write data to CH376 data port |
| 0 | 1 | 0 | 0 | Output | Read data from CH376 data port |
| 0 | 1 | 0 | 1 | output | read interface status from CH376 command port: Bit 7 is the interrupt flag, active with low-level, equivalent INT# pin, Bit 4 is the busy flag, active with high-level, equivalent to BZ pin of SPI interface |

CH376 occupy two address bits, when A0 pin is high-level, select the command port, you can write a new command, or read out interface state; When the A0 pin is low-level, select data port, you can read and write data.

MCU read/write CH376 through 8-bit parallel port, all the operations are formed by a command code, a number of input data and output data, some commands do not need input data, and some commands have no output data. Command operation steps are as follows:

①, MCU write command code to the command port when A0 = 1;

②, If this command has input data, write input data in turn when A0 = 0, one byte one time;

③, If this command has output data, read output data in turn when A0 = 0, one byte one time;

④, Command complete, some commands generate interrupt messages after execute complete, MCU can suspended or turn to step ① to continue executing the next command..

## 6.3. SPI serial interface

SPI synchronous serial interface signal lines include: SPI chip select input pin SCS, serial clock input pins SCK, serial data input pin SDI, serial data output pin SDO and interface busy status output pin BZ. Through SPI serial interface, CH376 can use less connection to connect to a variety of MCU, DSP, SPI serial bus of MCU, or a more long-range point to point connection.

SCS pin of CH376 is driven by SPI chip select output pin or common output pins of MCU, SCK pin is driven by SPI clock output pin SCK of MCU, SDI pin driven by SPI data output pin SDO or MISO of MCU, SDO pin is connect to the SPI data input pin SDI or MISO of MCU. For SPI hardware interface, recommended SPI setting is CPOL = CPHA = 0 or CPOL = CPHA = 1, and the data bit sequence is high in the front MSB first. SPI interface of CH376 also supports MCU common I/O pin simulate SPI interface to communication.

If INT # pin do not connect, you can get interrupt by query SDO pin, the method is let SDO pin exclusive one of MCU input pins, and through CMD_SET_SDO_INT command sets SDO pin do interrupt request output when the SCS chip select valid.

SPI interface of CH376 supports SPI Mode 0 and SPI Mode 3, CH376 always input data from the SPI clock SCK rising edge, and output data when allows output from the SCK falling edge, data bit order is high in the former, counted 8 bits as a byte.

The operation steps of SPI are:

① MCU generate SPI chip select to CH376, active with low-level;

② MCU sends a byte data according to SPI output mode, CH376 always received the first byte after SPI chip select SCS effective as a command code, the follow bytes as data;

③ SCM inquiry BZ pin waiting for SPI interface of CH376 is free, or directly delay TSC time (about 1.5uS);

④ If the write operation, MCU send to CH376 a byte data to be written, waiting for SPI interface is free, MCU continues to send a number of bytes data to be written, CH376 receive in turn, until MCU forbid SPI Chip Select;

⑤ If the read operation, MCU receive a byte of data from CH376, waiting for SPI interface is free, MCU continues to receive a number of bytes data from CH376, until MCU forbid SPI Chip Select;

⑥ MCU forbid SPI Chip Select, to end the current SPI operation.

The following figure is the SPI interface logic timing diagram, the former one is to send command 12H and write data 34H, the latter one is to send command 28H and read data 78H.
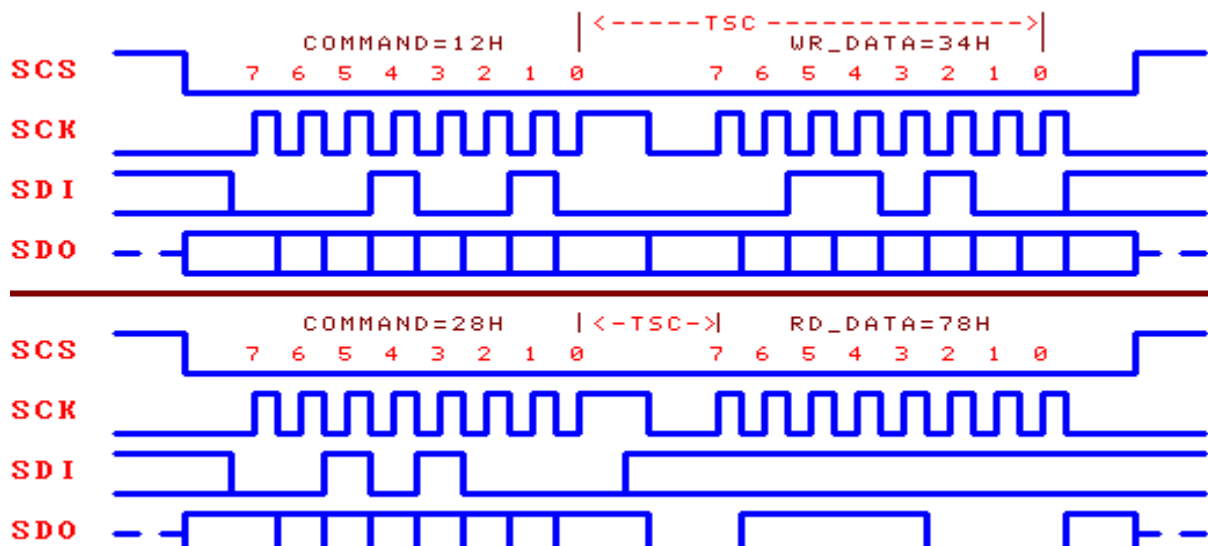
## 6.4. Asynchronous serial port

CH376 serial asynchronous serial data format is not compatible with CH375, and does not support external firmware USB device mode.

Asynchronous serial signal line includes: serial data input pin RXD and serial data output pin TXD. Through the serial interface, CH376 and MCU, DSP can be a more long-distance connection with the least connection.

RXD and TXD of CH376 can be connected to the MCU serial data output pin and serial data input pin respectively.

CH376 serial data format is the standard byte transmission mode, include 1 start bit, 8 data bits, 1 stop bit.

CH376 supports both hardware-set the default serial communication baud rate, also supports     MCU select the appropriate communication baud rate at any time by CMD_SET_BAUDRATE command. Every time after power-on reset, CH376 default serial communication baud rate is set by BZ/D4, SCK/D5, SDI/D6 three pin power level combination, refer to the following table (in the table 0 denote low-level , 1 denote high-level or un-connection).

| SDI/D6 pin | SCK/D5 pin | BZ/D4 pin | default serial communication baud rate after power-on reset |
|---|---|---|---|
| 1 | 1 | 1 | 9600 bps |
| 1 | 1 | 0 | 57600 bps |
| 1 | 0 | 1 | 115200 bps |
| 1 | 0 | 0 | 460800 bps |
| 0 | 1 | 1 | 250000 bps |
| 0 | 1 | 0 | 1000000 bps |
| 0 | 0 | 1 | 2000000 bps |
| 0 | 0 | 0 | 921600 bps |

In order to distinguish between command code and data, CH376 asked MCU through the serial port to send two synchronization code bytes (57H and ABH), and then send command code, follow send data or receiving data. CH376 will check time interval between the two synchronization code, synchronization code and command code. If the time interval is greater than serial input timeout SER_CMD_TIMEOUT (about 32mS), then CH376 will discard the synchronization code and command package. Serial command operation steps are as follows:

① MCU send CH376 the first synchronization code 57H through serial port;

② MCU send the second synchronous code 0ABH to CH376;

③ MCU send command code to CH376;

④ If this command have input data, then send CH376 input data in turn, one byte one time;

⑤ If this command have output data, then receive output data from CH376 in turn, e one byte one time;

⑥ Command complete, some commands generate interrupt message after execute complete and send interrupt state code through serial port, MCU can suspended or turn to step ① to continue executing the next command.

## 6.5. Other Hardware

CH376 integrated USB-SIE and Phy-I/O, CRC data check, USB-Host Controller, USB-Device controller, SD card SPI-Host controller, passive parallel interface, SPI-Slave controller, asynchronous serial port, dual-port SRAM, FIFO, high-speed MCU, firmware, crystal oscillator and PLL frequency multiplier, power-on reset circuit.

ACT# pins of CH376 used as status indication output. In the internal firmware USB device mode, when

USB device is not configured or cancel configuring, this pin output high-level; when USB device configure is complete, this pin output low-level. In USB host mode, when USB device disconnect, this pin output high-level; when USB device is connected, this pin output low-level. In SD card host mode, when SD card SPI communication is successful, this pin output low-level. ACT# pin of CH376 can connect external light-emitting diode LED which serial connect current-limiting resistor, to display the relevant state.

UD+ and UD-pin of CH376 is the USB signal lines, working in USB device mode, it should be directly connect to the USB bus; working in USB host mode, you can connect directly to the USB device. For chip security series insurance resistor or inductor or ESD protection device, then the AC and DC Equivalent series resistance should below 5Ω.

In the inner, CH376 set power-on reset circuit, normally, do not need provide external reset. RSTI pin used as external input asynchronous reset signal; when RSTI pin is high-level, CH376 is reset; when RSTI pin is low after resumption, CH376 will continue to delay reset about 35mS, then enter into normal working condition. In order to supply a reliable reset during power-on period and reduce the external interference, connect a 0.1uF capacitor between RSTI pin and VCC. RST pin (alias SD_DO pin) is active with high-level reset status output pin, can be used to provide power-on reset signal to an external MCU. When the CH376 power-on reset or reset by an external force, as well as during the reset delay, RST pin output high-level; CH376 reset completed and communication interface initialization is complete, RST pin resume to low-level.

CH376 work is required to provide an external 12MHz clock signal. In the inner, CH376 set crystal oscillator and oscillation capacitor. In generally, the clock signal is generated by CH376 built-in oscillator generated by crystal oscillator keeping frequency; the external circuit only need connect a nominal frequency of 12MHz crystal between XI and XO pins. If direct input from an external 12MHz clock signal, it should input from XI pin, and XO pin unconnected.

CH376 supports 3.3V or 5V supply voltage. When working voltage is 5V (when voltages higher than 4V), VCC pin of CH376 inputs external 5V power supply, and V3 pins should add 4700pF to 0.02uF power supply decoupling capacitor. When working voltage is 3.3V (when voltage less than 4V), V3 pin of CH376 should connected to VCC pin, while input external 3.3V power supply, and other circuit voltage which connected with CH376 can not be more than 3.3V.

## 7. Parameter

### 7.1. Absolute maximum rating (Stresses above those listed can cause permanent damage to the device. Exposure to maximum rated conditions can affect device operation and reliability.)

| Name | Parameter note | | Min. | Max. | Units |
|------|---------------|---|------|------|-------|
| TA | Ambient operating temperature | VCC=5V | -40 | 85 | ℃ |
| | | VCC=V3=3.3V | -40 | 85 | |
| | | VCC=V3=3V | -40 | 70 | |
| TS | Storage temperature | | -55 | 125 | ℃ |
| VCC | Voltage source (VCC connects to power, GND to ground) | | -0.5 | 6.0 | V |
| VIO | The voltage of input or output pin | | -0.5 | VCC+0.5 | V |

### 7.2. Electrical parameter (test conditions: TA=25℃, VCC=5V, excluding pin connection of USB bus)
(If the source voltage is 3.3V, multiply 40% of the current parameter)

| Name | Note of parameter | | Min. | Typical | Max. | Units |
|------|------|------|------|------|------|------|
| VCC | Power | V3 doesn't connect to VCC | 4.5 | 5 | 5.3 | V |
| | | V3 connects to VCC,V3=VCC | 3.0 | 3.3 | 3.6 | |
| ICC | Total source current when working | VCC=5V | | 12 | 30 | mA |
| | | VCC=3.3V | | 6 | 15 | |
| ISLP | Source current with low-power, I/O pin in suspend /internal pull up | VCC=5V | | 0.15 | | mA |
| | | VCC=3.3V | | 0.05 | | |
| VIL | Input Voltage LOW | | -0.5 | | 0.7 | V |
| VIH | Input Voltage HIGH | | 2.0 | | VCC+0.5 | V |
| VOL | Output Voltage LOW (draw 4mA current) | | | | 0.5 | V |
| VOH | Output Voltage HIGH (output 4mA current) | | VCC-0.5 | | | V |
| IUP | Input current in input port with internal pull-up resistor | | 30 | 80 | 160 | uA |
| IUP2 | Input current in input port with internal pull-up resistor Drain open pin ACT# and SD_CS | | 100 | 230 | 500 | uA |
| IDN | Input current in input port with internal pull-down resistor | | -30 | -80 | -200 | uA |
| VR | Edge power when power-up reset | | 2.4 | 2.7 | 2.9 | V |

Note: ACT# pin and SD_CS pin low-level draw current is 4mA, high-level output current is 200uA.

During CH376 reset, INT # pin and TXD pin can only provide high-level output current of 80uA.

## 7.3. Time sequence parameter (test conditions: TA=25℃, VCC=5V or VCC=.3V, refer to the accessorial image)
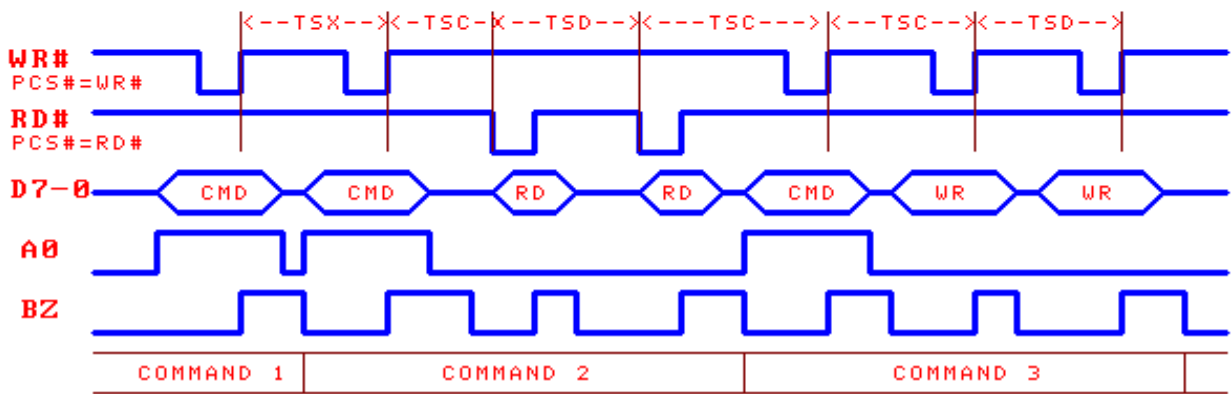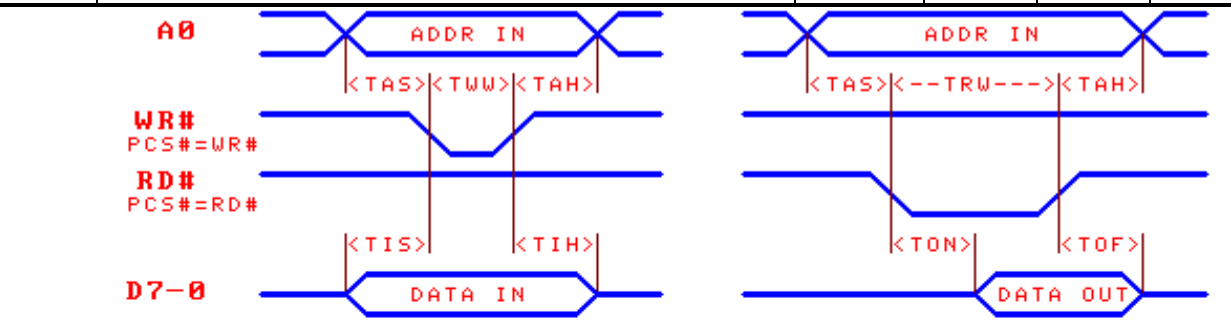
| Name | Explanation of parameter | Min. | Typical | Max. | Units |
|------|------|------|------|------|------|
| FCLK | The clock freq of X1 in mode of USB-HOST | 11.995 | 12.00 | 12.005 | MHz |
| TPR | Reset time of internal power-up | 25 | 35 | 40 | mS |
| TRI | Effective signal width of external reset | 100 | | | nS |
| TRD | Delay time of external reset | 25 | 32 | 35 | mS |
| TWAK | Wake-up time from low-power | 3 | 7 | 12 | mS |
| TE1 | The execute time of CMD_RESET_ALL | | 32 | 35 | mS |
| TE2 | The execute time of CMD_SET_USB_MODE | | 4 | 10 | uS |
| TE3 | The execute time of TEST_CONNECT or SET_ENDP? | | 2 | 3 | uS |
| TE4 | The execute time of CMD_SET_BAUDRATE | 200 | 1000 | 2000 | uS |
| TE0 | The execute time of other commands | | 1.5 | 2 | uS |
| TSX | The interval time between commands | 1.5 | | | uS |
| TSC | The interval time between command and data | 1.5 | | | uS |
| TSD | The interval time between data | 0.6 | | | uS |
| TINT | The time from receiving GET_STATUS to INT# recall interruption | | 1.5 | 2 | uS |

## 7.4. Parallel time sequence parameter (test conditions: TA=25℃,VCC=5V, the parameters in parentheses VCC = 3.3V, refer the following picture)

(RD implies RD# and CS# are active, execute read operation when RD#=CS#=0)

(WR implies WR# and CS# are active, execute write operation when WR#=CS#=0)

| Name | Explanation of parameter | Min. | Typical | Max. | Units |
|------|--------------------------|------|---------|------|-------|
| TWW | Write pulse width | 30(45) | | | nS |
| TRW | Read pulse width | 40(60) | | | nS |
| TAS | Address SET-UP TIME before RD or WR | 4(6) | | | nS |
| TAH | Address hold time after RD or WR | 4(6) | | | nS |
| TIS | Data input set-up time before write strobe WR | 0 | | | nS |
| TIH | Data input hold time after write strobe WR | 4(6) | | | nS |
| TON | Data output valid after Read active | 2 | 12 | 18(30) | nS |
| TOF | Data output invalid after Read inactive | 3 | 16 | 24(40) | nS |



**7.5. SPI time sequence parameter** (test conditions: TA=25℃,VCC=5V, the parameters in parentheses VCC = 3.3V, refer the following picture)
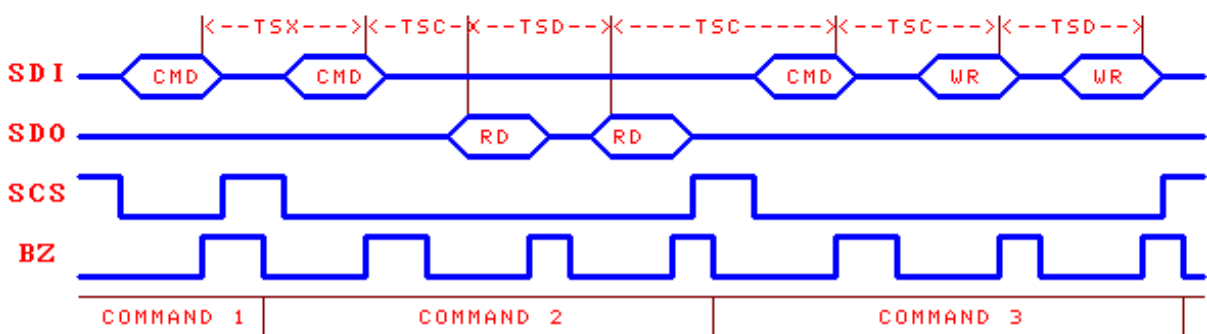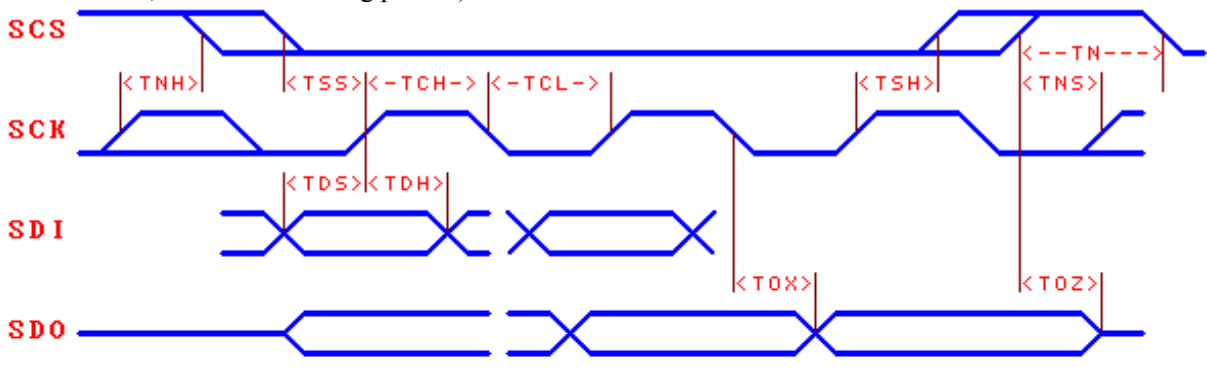
| Name | Explanation of parameter | Min. | Typical | Max. | Units |
|------|--------------------------|------|---------|------|-------|
| TSS | SCS active setup time before SCK rising edge | 20(30) | | | nS |
| TSH | SCS active hold time after SCK rising edge | 20(30) | | | nS |
| TNS | SCS inactive setup time before SCK rising edge | 20(30)) | | | nS |
| TNH | SCS inactive setup time after SCK rising edge | 20(30) | | | nS |
| TN | SCS inactive time (internal time of SPI operation) | 80 (120 | | | nS |
| TCH | High-level time of SCK clock | 14 (18) | | | nS |
| TCL | Low-level time of SCK clock | 18 (24) | | | nS |
| TDS | SDI input setup time before SCK rising edge | 6 (8) | | | nS |
| TDH | SDI input hold time after SCK rising edge | 2 | | | nS |
| TOX | SCK falling edge to SDO output change | 3 | 8(12) | 12(18) | nS |
| TOZ | SCS inactive to SDO output inactive | 4 | | 18(25) | nS |

# 8. Application

## 8.1. USB Flash Drive applications, 5V power supply (the following diagram)
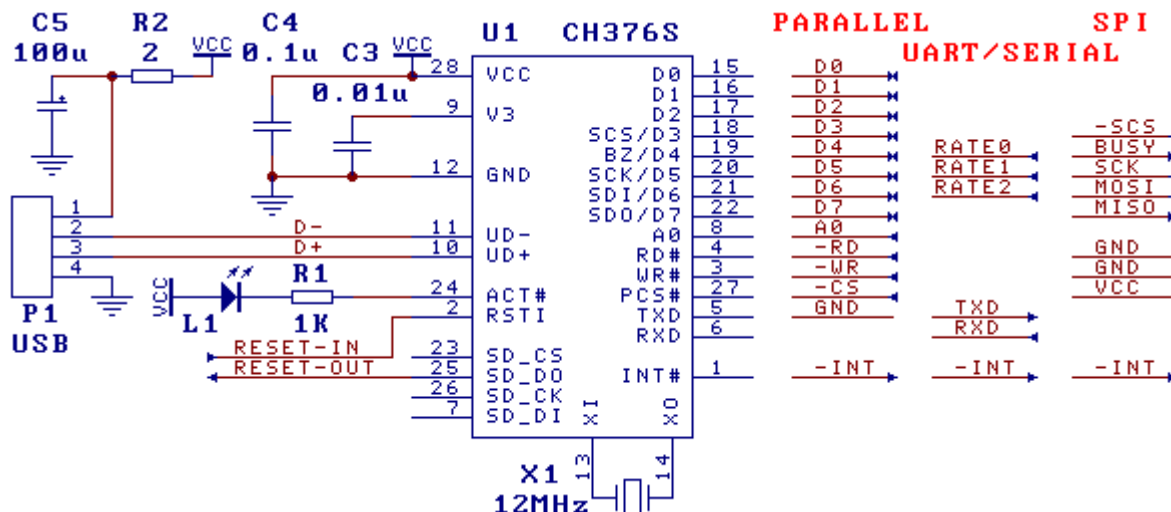


This is the 5V supply voltage of CH376 operation USB Flash Drive application circuit.

If configure CH376 as 8-bit parallel communication mode PARALLEL, then TXD should connect with GND, other pins unconnected.

If configure CH376 as SPI serial communication mode SPI, then RD # and WR # should connect with GND, other pins unconnected.

If configure CH376 as asynchronous serial communications mode UART/SERIAL, then all pins should be unconnected, default serial communication baud rate configure by SDI/D6, SCK/D5, BZ/D4 three-pin. If you need to modify CH376 serial communication baud rate dynamically, it is proposed control RSTI pin of CH376 by MCU I/O pins, in order to reset CH376 to restore to default communication baud rate if necessary. As RSTI pin with built-in pull-down resistors, so MCU such as MCS51 the quasi-bidirectional I/O pin drivers may need to plus a pull-up resistor about a few KΩ.

As INT# pin, and TXD pin can only provide weak high-level output current during CH376 reset, for more long-distance connections, in order to avoid MCU mis-operation caused by INT# or TXD disturbance during CH376 reset, plus 2KΩ ~ 5KΩ pull-up resistor at INT# pin or TXD pin in order to hold a more stable high-level. When CH376 reset complete, INT# pin and TXD pin will be able to provide 4mA high-level

output current or 4mA low-level draw current.

To save pins, MCU can not connect INT # pin in CH376 chip, get interrupt message method is as follows:

① 8-bit parallel mode by inquiry CH376 status port (namely command port) to get interface status, bit-7 is the interrupt flag PARA_STATE_INTB, activewith low-level, is equivalent to query INT# pin, bit-7 is 0 indicate the interrupt request;

② SPI interface mode by inquiry SDO pin informed interrupted (after power-on or reset the first through CMD_SET_SDO_INT command to set SDO pin when SCS chip select invalid interrupt request output concurrently), SDO is low indicate the interrupt request;

③ Serial mode CH376 generate interrupt message (INT# turns to low-level) at the same time, will be sent interrupt status code through the serial port directly, MCU receive interrupt status code indicate the interrupt request.

R2 used to limit current of USB host provide to external USB devices, if necessary, series with a current-limiting role fast electronic switch, USB power supply voltage must be 5V.

The capacitance C3 eliminates the coupling of inner power of CH376.The capacity of C3 is 4700pF to 0.02uF.It is made of monolithic or high frequency ceramic. The C4 and C5 are used to decoupling of external power. The C4 is 0.1uF and made of monolithic or high frequency ceramic. The crystal X is used as clock oscillating circuit.The USB-HOST mode needs exact frequency. The frequency of X1 is 12MHz±0.4‰.

When designing the PCB, pay much attention to some notes: decoupling capacitance C3 and C4 must keep near to connection pin of CH376; makes sure D+ and D- are parallel and supply ground net or pour copper beside them to decrease the disturbance from outside signal; the relevant signal between X1 and X2 must be kept as short as possible. In order to lessen the high frequency clock disturbance, play ground net or pour copper to the relative equipment.

## 8.2. SD card and USB Flash Drive applications, 3.3V power supply (the following diagram)
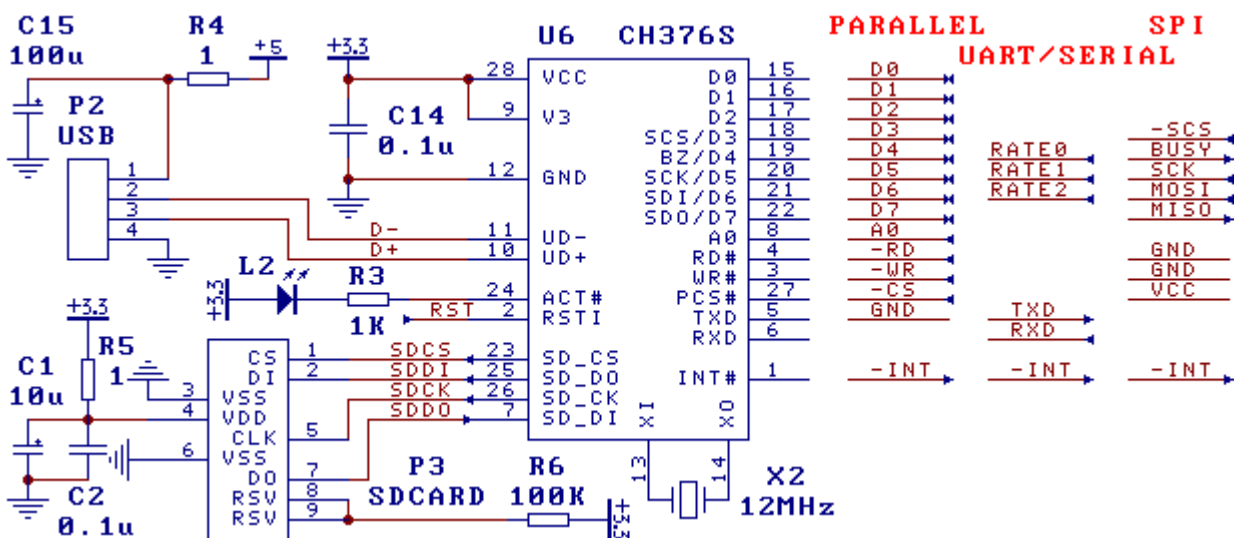


This is 3.3V or 3V supply voltage CH376 operations USB Flash Drive and SD card application circuit.

P3 is simplify SD card slot, SD card plug status pin can directly connected to the MCU I/O or interrupt input pin.

Communication Interface configuration and 5V voltage applications are the same, refer to Section 8.1.

R4 used to limit current of USB host provide to external USB devices, if necessary, series with a current-limiting role fast electronic switch, USB power supply voltage must be 5V.

CH376 power supply voltage of 3.3V, in figure V3 pin and VCC pin will be shorted together to 3.3V

input voltage.

Capacitors C14 and C15 for external power supply decoupling; C14 is the capacity of 0.1μF and made of monolithic or high frequency ceramic.

## 8.3. Application foundation

USB Flash Drive (or SD card, the same below) provides a number of physical sectors used for data storage, the size of each sector is usually 512 bytes. Due to a computer usually organizations the physical sector of USB Flash Drive to FAT file system, in order to facilitate MCU through USB Flash Drive or SD card and the exchange of data between computers, MCU should also be adopted under the FAT specification documents in the form to save and get data in USB Flash Drive.

A USB Flash Drive can have a number of files, each file is a set of data collection, distinguished and identified by the file name. The actual file data is stored may not be continuous, but through a set of "pointer" link on a number of blocks (is the allocation of units or clusters), which can be increased as necessary to keep file size to accommodate more data. Directory (folder) in order to facilitate the classification management, administrators can specify the number of file, for example, the file of2004 to a directory (folder).

In FAT file system, the disk clusters as the basic unit of capacity to be allocated, while the clusters size is always a multiple of sectors, so the file space is always a multiple of clusters, also a multiple of sectors. Although the file space is occupied by a multiple of clusters or sectors, but in practical application, the length of valid data saved in the file is not necessarily a multiple of sectors, so FAT file system in the file directory information FAT_DIR_INFO specifically record the length of valid data in the current file, the number of bytes of valid data, which is known as file size, file length is always less than or equal to the space occupied by the file.

After write data to the file, if original data is covered, then the file size may not change, when more than the original file length, becomes additional data, then the file size should be changed (increased). If you append data to files, there is no modify the file size in file directory information, then FAT file system will consider the data over the length file is invalid, normally, the computer can not read out the data over the file length, although the data actually exists.

If the data is less or the data is not continuous, you can append the data immediately after each update file size in the file directory information, but if large volumes of data and need to write data continuously, update file directory information immediately will reduce efficiency, and modify file directory information frequently will also shorten flash memory life of USB Flash Drive (because flash memory can only erase a limited time), so in this case, it should be written continuous multiple sets of data then updated the file size in file directory Information once, or wait until you close the file and then update the file size, CMD_FILE_CLOSE command can refresh the file size in memory to the USB Flash Drive file directory information.

While CH376 maximum support 1GB single file, but in order to improve efficiency, the proposed length of a single file no more than 100MB, usually a few KB to several MB range is normal, when more data can be divided into multiple directories, multiple files to store.

In generally, MCU or embedded system processing USB Flash Drive file systems need to achieve the following diagram of 4 levels on the left, on the right is the level of the internal structure of USB Flash Drive. As CH376 is not only a universal USB-HOST hardware interface chip, but also built-in USB underlying transport-related firmware program, Bulk-Only protocol firmware program, FAT file system management firmware program, contains the following diagram of 4 levels on the left (marked as gray), so the actual MCU programs only need to send document management and file read and write commands.
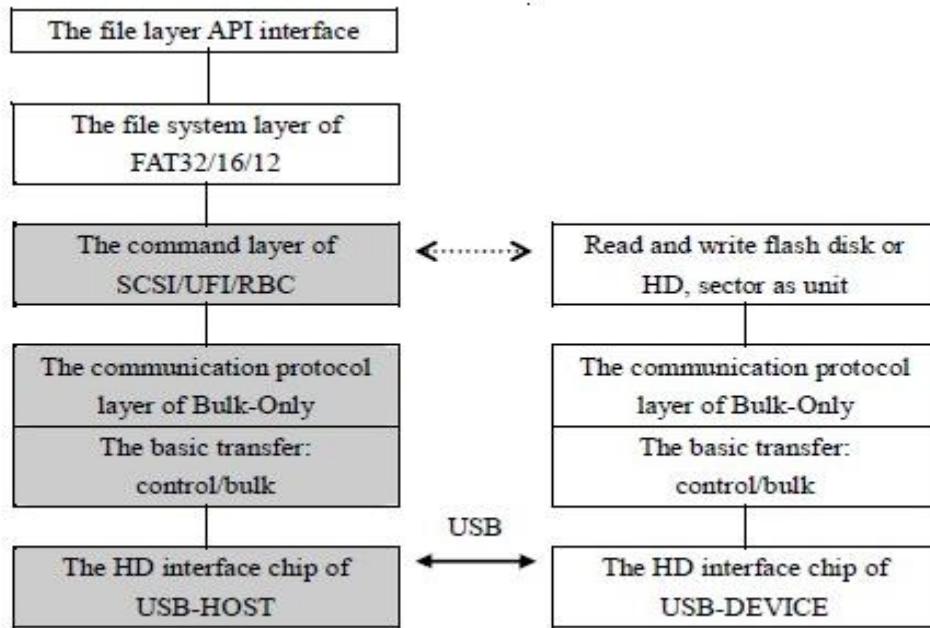
## 8.4. Speediness Application Reference Step

Please refer to example program, tranfer the package has multiple commands after the subroutine. The following steps use the original command code for reference only.

8.4.1. Initialization, for any one file operations a necessary step before

① CMD_SET_USB_MODE command, into the USB-HOST work mode or SD card host work mode (mode 3)

② Waiting for connecting USB Flash Drive or SD card, USB Flash Drive can be checked by CH376 automatically and generate an interrupt notification, or by the MCU send CMD_DISK_CONNECT command to CH376 for regularly check, SD cards must to be checked by MCU itself.

③ CMD_DISK_MOUNT command, initialize USB Flash Drive or SD card, and test the disk ready or not, you can try again up to 5 times after failure

④The above steps need implementation only once, unless USB Flash Drive or SD card to reconnect after disconnect, that must return to step ②

8.4.2. read file in turn

① CMD_SET_FILE_NAME command + CMD_FILE_OPEN command, open file

② many times CMD_BYTE_READ command + CMD_RD_USB_DATA0 command + CMD_BYTE_RD_GO command, read data

③ CMD_FILE_CLOSE command, close file, optional operation

8.4.3. rewrite file in turn(overwrite original data, shift additional data if more than the length of the original file)

① CMD_SET_FILE_NAME command + CMD_FILE_OPEN command, open file

② many times CMD_BYTE_WRITE command + CMD_WR_REQ_DATA command + CMD_BYTE_WR_GO command, write data

③ CMD_FILE_CLOSE command, parameter is 1, close file and allow automatic update file size

8.4.4. Addition data to existing files

① CMD_SET_FILE_NAME command + CMD_FILE_OPEN command, open file

② CMD_BYTE_LOCATE command, parameter is 0FFFFFFFFH, move file pointer to the end of file

③ many times CMD_BYTE_WRITE command + CMD_WR_REQ_DATA command + CMD_BYTE_WR_GO command, write data

④ CMD_FILE_CLOSE command, parameter is 1, close file and allow automatic update file size


8.4.5. Create file and write data

① CMD_SET_FILE_NAME command + CMD_FILE_CREATE command, create file

② many times CMD_BYTE_WRITE command + CMD_WR_REQ_DATA command + CMD_BYTE_WR_GO command, write data

③ CMD_FILE_CLOSE command, parameter is 1, close file and allow automatic update file size


8.4.6. Rewrite file after read file

① CMD_SET_FILE_NAME command + CMD_FILE_OPEN command, open file

② many times CMD_BYTE_READ command + CMD_RD_USB_DATA0 command + CMD_BYTE_RD_GO command, read data

③ CMD_BYTE_LOCATE command, parameter is 0, move file point to the head of file

④ many times CMD_BYTE_WRITE command + CMD_WR_REQ_DATA command + CMD_BYTE_WR_GO command, write data

⑤ CMD_FILE_CLOSE command, parameter is 1, close file and allow automatic update file size


8.4.7. Addition data if file already exist, create file and write data if file does not exist

① CMD_SET_FILE_NAME command + CMD_FILE_OPEN command, open file, if file is not exist returned ERR_MISS_FILE, then turn to step ③

② CMD_BYTE_LOCATE command, parameter is 0FFFFFFFFH, move file pointer to the end of the file, then turn to step ④

③ CMD_FILE_CREATE command, create file

④ many times CMD_BYTE_WRITE command + CMD_WR_REQ_DATA command + CMD_BYTE_WR_GO command, write data

⑤ CMD_FILE_CLOSE command, parameter is 1, close file and allow automatic update file size

8.4.8. Modify file name, file date/time, file length and other file directory information, please refer to the explain of example EXAM10

① CMD_SET_FILE_NAME command + CMD_FILE_OPEN command, open file

② CMD_DIR_INFO_READ command, parameter is 0FFH, read file directory information into memory

③ Read out original file directory information with CMD_RD_USB_DATA0 command

④ CMD_DIR_INFO_READ command, parameter is 0FFH, read file directory information into memory

⑤ Write a new file directory information with CMD_WR_OFS_DATA command

⑥ CMD_DIR_INFO_SAVE command, save file directory information

⑦ CMD_FILE_CLOSE command, parameter is 0, close file and forbid automatic update file size, optional operation


8.4.9. Create subdirectory (folder), please refer to the explain of example EXAM9

① CMD_SET_FILE_NAME command + CMD_DIR_CREATE command, create subdirectory (folder)

② CMD_FILE_CLOSE command, parameter is 0, close file and forbid automatic update file size

8.4.10. Operation lower-case file name and long file name, please refer to the explain of example EXAM11

8.4.11. Search and enumeration file name, enumeration all files in the disk, please refer to the explain of example EXAM13

8.4.12. Switch master and slave, communication with PC, read and write files of USB Flash Drive or SD card, please refer to the explain of example EXAM0

## 8.5. USB device applications
Please refer to the datasheet of CH372DS1.PDF and its application information.