CH376 Programming Guide

(U disk and SD card)

Chinese Manual

http://wch.cn

Version History

version	date	description
V1.0	January 2011	First amendment

table of Contents

1 Overview

CH376 is a file management control chip, the microcontroller for the file system read U disk or SD card. And can operate using CH376 USB keyboard and mouse, printer and other USB devices. This document mainly describes how the client software to communicate with the CH376 firmware, file U disk and SD card operation.

CH376 supports three communication interfaces: 8-bit parallel, SPI interface or an asynchronous serial port;

2, the hardware abstraction layer

2.1 8-bit parallel port

Parallel signal line comprises: bit bidirectional data bus D7 ~ D0, read strobe input pin RD #, write strobe input pin WR #, chip select input pin PCS #, INT # interrupt output pin, and the address input pins similar A0 parallel timing Intel microcontroller, CH376 chip RD # WR # pin and the pin can be connected to the microcontroller output pins are read strobe and write strobe output pin. Similar parallel timing Motorola microcontroller, by CH376 chip RD # pin should be connected to the low level and WR # pin is connected to the reader microcontroller output pins direction R / -W.

Parallel read and write sequence is as follows:

```
/ * Write command to the command port CH376 * / void
        xWriteCH376Cmd (UINT8 mCmd) / * write command to the CH376 * / {

    / * (* (Volatile unsigned char xdata *) 0xBDF1) = mCmd;
    * / / * Written via an external 51-chip parallel bus interface operation to the command CH376 * /
    CH376_DATA_DAT_OUT (mCmd); / * parallel to the data output of CH376 * / CH376_DATA_DIR_OUT
    (); / * set the direction parallel to the output * / CH376_A0 = 1 ; CH376_CS = 0;


    CH376_WR = 0; / * output valid write control signal, the write port command of CH376 * / // CH376_CS = 0; / * this operation meaningless, only
    for the delay, write pulse width is greater than required CH376 40nS * / CH376_WR = 1; / * output inactive control signal, to complete the
    operation of CH376 * / CH376_CS = 1; CH376_A0 = 0;


    CH376_DATA_DIR_IN (); / * output prohibited data * / / * mDelay0_5uS (); mDelay0_5uS ();
   mDelay0_5uS (); * / / * read and write cycles to ensure that the delay is greater than 1.5us 1.5uS, or a
   state query instead of * /}


/ * Write data to data port CH376 * / void
        xWriteCH376Data (UINT8 mData) / * Write data to CH376 * / {

    / * (* (Volatile unsigned char xdata *) 0xBCF0) = mData; * / / * the microcontroller 51 by an outer
                                 Parallel bus interface unit operation of writing data to CH376 * /
    CH376_DATA_DAT_OUT (mData); / * output data to the parallel port of CH376 * /
    CH376_DATA_DIR_OUT ();                      / * Set output direction parallel * /
    CH376_A0 = 0;
    CH376_CS = 0;
```

```
        CH376_WR = 0; / * output valid write control signal, a write data port of CH376 * / // CH376_CS = 0; / * this operation meaningless, only for
        the delay, write pulse width is greater than required CH376 40nS * / CH376_WR = 1; / * output inactive control signal, to complete the
        operation of CH376 * / CH376_CS = 1;

        CH376_DATA_DIR_IN (); / * output prohibited data * / //
        mDelay0_5uS ();                          / * Read and write cycles to ensure that greater than 0.6uS * /
    }

    / * CH376 data read from the data port * /
    UINT8 xReadCH376Data (void) / * read data from CH376 * / {

        UINT8        mData;
        / * MData = (* (volatile unsigned char xdata *) 0xBCF0); * / / * by 51 microperipherals
                                             Parallel bus interface unit operation to read data from CH376 * /
        // mDelay0_5uS ();                     / * Read and write cycles to ensure that greater than 0.6uS * /
        CH376_DATA_DIR_IN (); / * set the direction parallel input * / CH376_A0
        = 0; CH376_CS = 0;

        CH376_RD = 0; / * output valid read control signal, a read data port of CH376 * / CH376_CS = 0; / * this operation meaningless,
        only for the delay, write pulse width is greater than required CH376 40nS * / mData = CH376_DATA_DAT_IN ( ); / * CH376 input
        data from the parallel port * / CH376_RD = 1;
                                                  / * Output inactive control signal, to complete the operation of CH376 * /
        CH376_CS = 1; return
      (mData);}


    / * Read from the command port state * CH376 /
    UINT8          xReadCH376Status (void) / * reads from CH376 state only in parallel mode * / {

        UINT8 mData;
        / * MData = (* (volatile unsigned char xdata *) 0xBDF1); * / / * by 51 microperipherals
                                             Parallel bus interface unit operation status is read from CH376 * /
        CH376_DATA_DIR_IN (); / * set the direction parallel input * / CH376_A0
        = 1; CH376_CS = 0;

        CH376_RD = 0; / * output valid read control signal, reads the status port of CH376 * / CH376_CS = 0; / * this operation
        meaningless, only for the delay, write pulse width is greater than required CH376 40ns * / mData = CH376_DATA_DAT_IN ( ); / *
        CH376 input data from the parallel port * / CH376_RD = 1;
                                                  / * Output inactive control signal, to complete the operation of CH376 * /
        CH376_CS = 1; CH376_A0
      = 0; return (mData);}
```

2.2. 4-wire serial interface (SPI)

SPI synchronous serial interface signal lines include: SPI chip select input pin SCS, the serial clock input pin SCK, the serial data input pin SDI, serial data output pin SDO, INT # interrupt output pin, and an interface is busy the BZ state output pin (can be omitted).

1, CH376 supports SPI Mode 0 and Mode 3 (CH376 rising edge of SCK sample data, the output data of the falling edge of SCK);

2, CH376 support the highest clock speed SPI 24MHZ;

3, CH376 operating in SPI mode, the command processing by the SCS as the first byte after the high to low; period after the end of a command should be processed microcontroller SCS is asserted high;

Here is the microcontroller 51 analog SPI timing IO port:

```
void Spi376OutByte (UINT8 d) / * SPI outputs 8-bit data * / {

    / * If the SPI interface hardware, the data should be written to the SPI data register first, and then wait for the query to the SPI status register SPI transfer
    complete byte * / UINT8
                i;
    for (i = 0; i <8; i ++) {

        CH376_SPI_SCK = 0;
        if (d & 0x80) CH376_SPI_SDI = 1; else
        CH376_SPI_SDI = 0; d << = 1;
                                        / * Bit is MSB first data * /
        CH376_SPI_SCK = 1; / * CH376 sampling edge of the clock input * /}}



UINT8       Spi376InByte (void) / * SPI input 8-bit data * / {

    / * If the hardware SPI interface, the query should be the first to wait for the SPI status register SPI byte transfer, then read data from
    the SPI data register * / UINT8 i, d; d = 0;


    for (i = 0; i <8; i ++) {

        CH376_SPI_SCK = 0; / * CH376 falling edge of the output clock * / d << =
        1;                                  / * Bit is MSB first data * /
        if (CH376_SPI_SDO) d ++;
      CH376_SPI_SCK = 1;}

    return (d);}


#define         xEndCH376Cmd ()
{
    CH376_SPI_SCS = 1;
}
/ * SPI chip select is invalid, ending command CH376, SPI interface mode only * /

void        xWriteCH376Cmd (UINT8 mCmd) / * write command to the CH376 * / {

    CH376_SPI_SCS = 1; SPI sheet is not prohibited by xEndCH376Cmd / * prior to prevent selected from * / mDelay0_5uS ();

    / * For bidirectional I / O pin SPI interface simulation, it must ensure that set SPI_SCS, SPI_SCK, SPI_SDI output direction,
    SPI_SDO direction input * / CH376_SPI_SCS = 0;
                                        / * The SPI chip select is active * /
    Spi376OutByte (mCmd); / * issue command code * / mDelay0_5uS (); mDelay0_5uS (); mDelay0_5uS (); / * read and
  write cycles to ensure that the delay is greater than 1.5us 1.5uS, or a state in place of the above query row * /}


void        xWriteCH376Data (UINT8 mData) / * Write data to CH376 * / {

    Spi376OutByte (mData); //
    mDelay0_5uS ();                     / * Read and write cycles to ensure that greater than 0.6uS * /
} UINT8
        xReadCH376Data (void) / * read data from CH376 * / {

    // mDelay0_5uS ();                  / * Read and write cycles to ensure that greater than 0.6uS * /
    return (Spi376InByte ());}
```

2.3. 2-wire serial interface (UART)

Asynchronous serial signal line comprising: a serial data input pin and a serial data output pin RXD and TXD interrupt output pin INT #.

1, CH376 RXD and TXD of the chip may be connected to the serial data output pin serial data input pin and the microcontroller, respectively. 2, serial data format is standard CH376 byte transmission mode, including a start bit, 8 data bits, 1 stop bit.

3, CH376 default baud rate of 9600, to support the microcontroller at any time by selecting the appropriate communication baud rate CMD_SET_BAUDRATE command. After each power-on reset, CH376 default serial communication baud rate of BZ / D4, SCK / D5, SDI / D6 three level combination setting pins (see Section CH376DS1 6.4)

4, CH376 transmission command is a two-byte sync code starts (57H, ABH, command code)

Here is an example of single-chip serial timing 51:

```
void        xWriteCH376Cmd (UINT8 mCmd) / * write command to the CH376 * / {

        TI = 0;
        SBUF = 0x57; / * 1st serial preamble start operation * / while (TI == 0);
        TI = 0;
        SBUF = 0xAB; / * second serial preamble start operation * / while (TI ==
        0); TI = 0;

        SBUF = mCmd; / * serial output * / while (TI
      == 0);}


    void        xWriteCH376Data (UINT8 mData) / * Write data to CH376 * / {

        TI = 0;
        SBUF = mData; / * serial output * / while (TI
      == 0);}


    UINT8 xReadCH376Data (void) / * read data from CH376 * / {

        UINT32 i;
        for (i = 0; i <500000; i ++) / * Timeout Count prevented * / {

            if (RI)                               / * * Received serial /
            {
                RI = 0;
                Return (SBUF); / * serial input * /}}


        return (0);                     / * The case should not have happened * /
    }
```

2.4. CH376 interrupt pin (INT #) is used

The interrupt request INT # pin of CH376 default output is active low, the microcontroller may be connected to the interrupt input pin or the normal input pin, the microcontroller can interrupt or manner to obtain CH376 interrupt request.

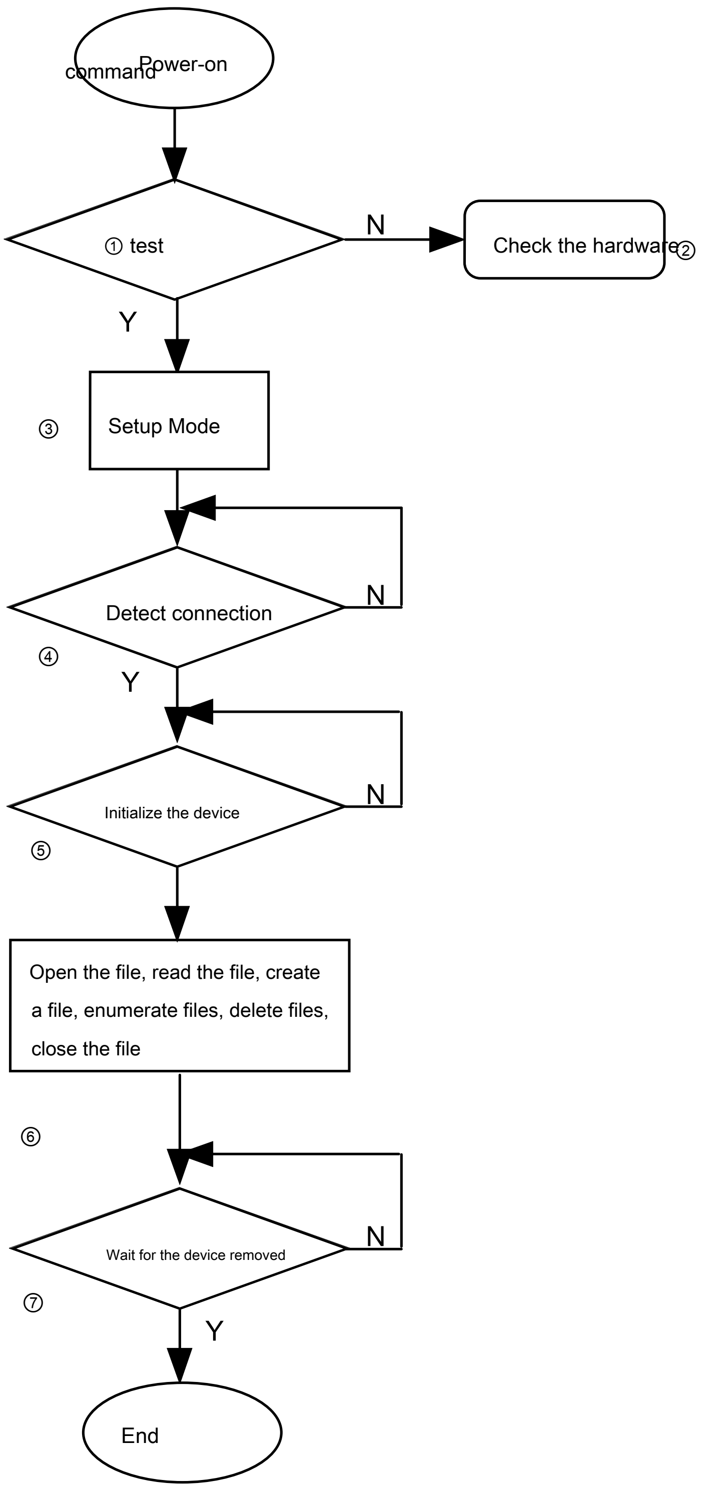
To save pins, may not be connected to the microcontroller INT # pin of CH376, interrupted by other means known. Parallel mode: () function reads the most significant bit by xReadCH376Status parallel port, this bit is equivalent to the state of the interrupt pin, the highest bit is low, it indicates that an interrupt is generated;

SPI Interface: CH376 be provided by sending a command SET_SDO_INT (0BH) also serves as the SDO pin INT # pin is used when the chip select (SCS) is invalid; not recommended for use when using the SPI interface hardware;

Serial manner: When CH376 generate an interrupt, it will be transmitted through TXD of CH376 a status code to the microcontroller indicating interrupt occurs, the microcontroller may receive the query status code;

3, the application layer system

The operation of the main flowchart of the chip:


command
Power-on
① test
N
Check the hardware②
Y
③
Setup Mode
Detect connection
N
④
Y
Initialize the device
N
⑤
Open the file, read the file, create a file, enumerate files, delete files, close the file
⑥
Wait for the device removed
N
⑦
Y
End

Note:

①, power-on delay 50MS;

②, send CMD_CHECK_EXIST (06H) command, sending 55H data, the right to return 0AAH. 0AAH non-data, and check the MCU and CH376 CH376 hardwired Reset and crystal vibrating;

③, transmission CMD_SET_USB_MODE (15H) command, subsequent data mode is the mode .U disk 6, SD card mode 3. Returns status CMD_RET_SUCESS (51H), selective reading of this state;

④, transmission CMD_DISK_CONNECT (30H) command, an interrupt return interrupt status USB_INT_CONNECT (15H), the connection device, SD card detecting device are connected by the first pin 10 SD SD card connector showing low high no link;

⑤, transmission CMD_DISK_MOUNT (31H) command, an interrupt return interrupt status USB_INT_SUCESS (14H), initialization is completed. If it fails, it needs to be repeated 5 times, and needs 200MS delay before each retry;

⑥, each step in detail with reference to the flowchart;

⑦, transmission CMD_DISK_CONNECT (30H) command, generates an interrupt, the interrupt status returns USB_INT_DISCONNECT (16H), the device is removed. If the return interrupt status is USB_INT_CONNECT (15H), continue to wait;

Note: This guide is only for host mode do CH376, CH372 device mode refer to the instruction manual;

3.1 chip initialization

Before chip CH376 initialized CMD_CHECK_EXIST needs to send commands for checking the operating state, and a communication interface. Check the CH376 chip is working properly. When CH376 chip work. CH376 chip can to initialize. MCU needs to send a command to initialize CMD_SET_USB_MODE of CH376. The need to enter a command requires the data, the data pattern code:

Mode code switches to 03H to the SD card host mode, managing and accessing the SD card file; 06H mode code to switch to the USB host mode when enabled, automatically generated SOF packet; on USB devices Please refer to the manual of CH372, fully compatible with CH376 and CH372 USB device mode chip. Typically, the USB mode is provided within 10uS of time to finish, and the output state returns to the state 51H under normal circumstances. Actual operation, as long as the command is checked by CMD_CHECK_EXIST set mode command may not need to read the state.

3.2. Query device connection

After the mode setting is completed, it detects devices connected. SD card and connecting U plate detecting follows: SD card

connection detection mode:

SD card connector pin 10 is used to detect whether the SD card is inserted. When the SD card is not inserted, SD card pin 10 voltage is high, the SD card into the low level.

SD card connector to 11 pin is used to detect whether the SD card is write-protected. Is not write-protected high-level, low level is write-protected.

U-sense connections:

U disk detection command CMD_DISK_CONNECT connection needs to be sent. This command is sent, an interrupt is generated to MCU. If the interrupt status is USB_INT_SUCCESS. Then that U disk connection.

3.3. Initialization equipment

When the MCU is detected U disk or SD card is connected. The need for U disk, SD card initialization, send CMD_DISK_MOUNT. If this command returns the interrupt status for the USB_INT_SUCCESS, description of the U disk or SD card initialization is complete. For some U disk or SD card, you may need to be repeated several times to send the command to complete the initialization of U disk, SD card. 200MS delay about each addition before retrying.

3.4. Open File

3.4.1 Procedure Open file or directory as the current directory or root

directory:

1, transmission CMD10_SET_FILE_NAME (2FH) command to set the file name; 2, transmits the file

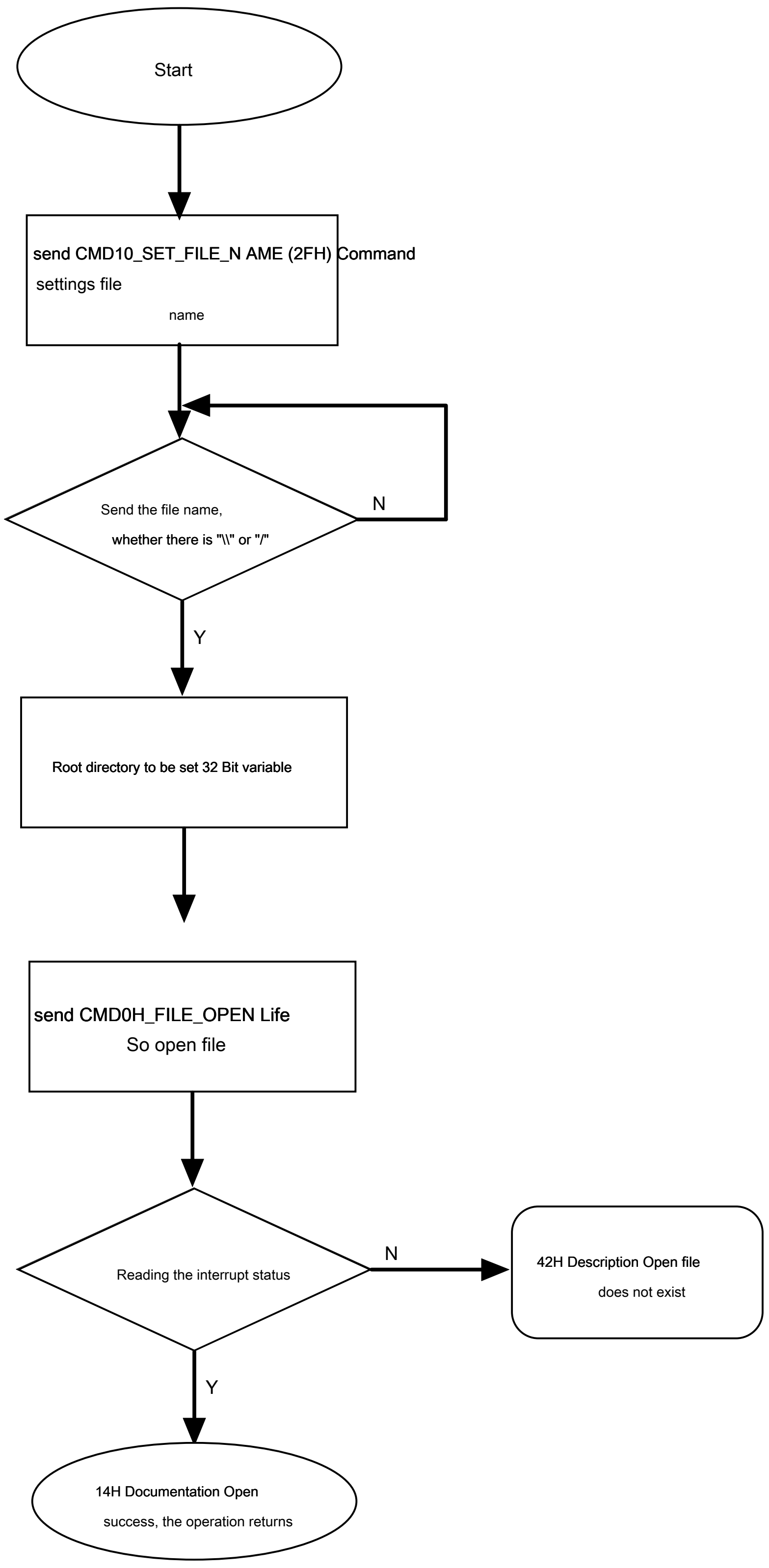
name, the file name to the end of the number 0;

3, when the file path in the "\\" or "/", the sending end of the file name;

4, if the root directory file, is provided inside of CH376 needs to send 32-bit variable. Transmission CMD50_WRITE_VAR32 (0DH), then write data VAR_CURRENT_CLUST (64H), and then write 32-bit data of 0;

5, Open File command to send CMD0H_FILE_OPEN (32H), open the file, if the state is to interrupt USB_INT_SUCCESS (14H), the file is opened successfully, if the interrupt status is ERR_MISS_FILE (42H) Description Open the file does not exist;

Flowchart is as follows:


Start
send CMD10_SET_FILE_N AME (2FH) Command settings file name
Send the file name, whether there is "\\" or "/"
N
Y
Root directory to be set 32 Bit variable
send CMD0H_FILE_OPEN Life So open file
Reading the interrupt status
N
42H Description Open file does not exist
Y
14H Documentation Open success, the operation returns

3.4.2 The open file operation step in multi-level

directory as follows:

1, first find the identifier of the subordinate directory "\\" or "/";

2, find the current name of a file, if the root directory, open the file, you need to add "\\" or "/"; 3, the cycle is called the "root directory or open a file or directory under the current directory" until the file is opened .

Open the file, for example:

Open the file full path "\\ ABC \\ 123 \\ WCH.TXT". So first open "\\ ABC" folder, and then open the "123" folder, then open the "WCH.TXT" file.

3.5. Create a file

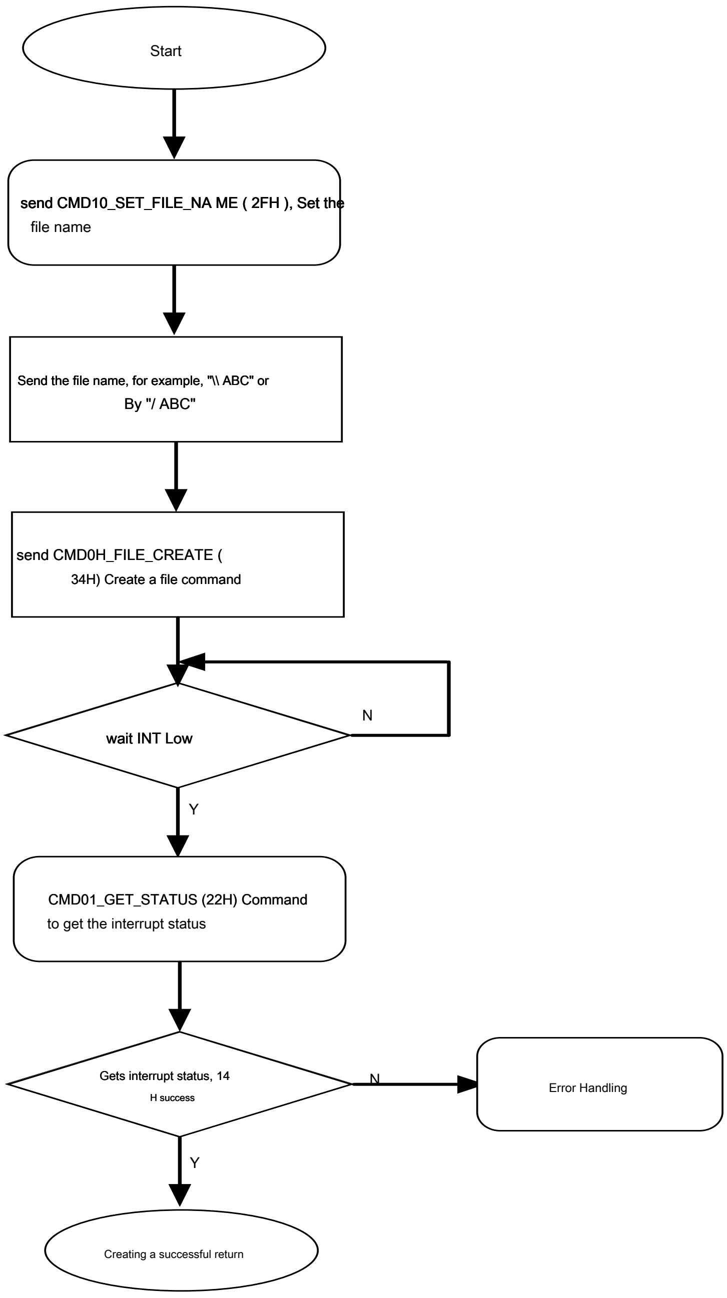
3.5.1 steps to create a file in the root directory or the current directory

as follows:

1, transmission CMD10_SET_FILE_NAME (2FH), set the file name; 2, then sends the file (root directory file names are created need to add "\\" or "/"); 3, after completion of the transmission, transmission CMD0H_FILE_CREATE (34H), creating file;

4, wait CH376 chip interrupt, when the interrupt pin is low, sending CMD01_GET_STATUS (22H) command to get the interrupt status, if the interrupt status is USB_INT_SUCCESS (14H), the instructions to create the file successfully;

Flowchart is as follows:


Start
send CMD10_SET_FILE_NA ME (2FH), Set the file name
Send the file name, for example, "\\ ABC" or By "/ ABC"
send CMD0H_FILE_CREATE (34H) Create a file command
wait INT Low
N
Y
CMD01_GET_STATUS (22H) Command to get the interrupt status
Gets interrupt status, 14 H success
N
Error Handling
Y
Creating a successful return

3.5.2 steps to create a file under the multi-level

directory as follows:

1, first find the last level file name. For example, create a file "123.TXT" in the "ABC" folder above, the full path of the file is created for the "//ABC//123.TXT" Well, finally a file called "123.TXT". You do not need to "\\";

2, the cycle is called the "root directory or open a file or directory under the current directory"; 3, calls

"the root directory of the current file or create" Create a file;

3.6. Byte-write file

Steps are as follows:

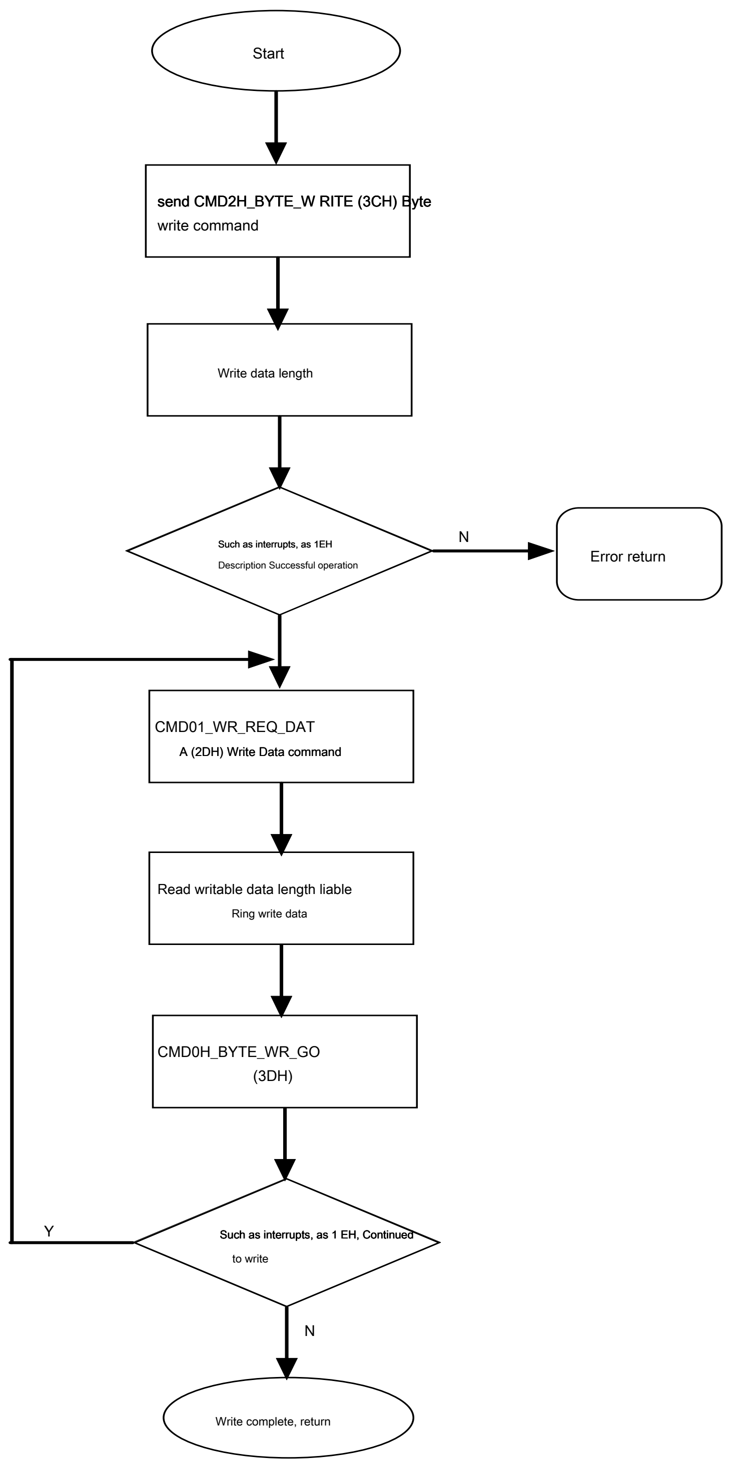
1, transmission CMD2H_BYTE_WRITE (3CH) Write Byte command;

2, followed by the subsequent need to write the write data length of byte write. Supports up to 65535 bytes of digital bytes; 3,

waiting for an interrupt, if the interrupt status is USB_INT_DISK_WRITE (1EH);

4, transmission CMD01_WR_REQ_DATA (2DH) commands, read data can be the length to CH376. Then the data write cycle; 5, after the data is written, transmitting CMD0H_BYTE_WR_GO (3DH), waiting for an interrupt, if the interrupt status is USB_INT_DISK_WRITE (1EH) Step 4 is to recall the data write cycle. If other interrupt status, then write complete;

Flowchart is as follows:


Start
send CMD2H_BYTE_W RITE (3CH) Byte write command
Write data length
Such as interrupts, as 1EH Description Successful operation
N
Error return
CMD01_WR_REQ_DAT A (2DH) Write Data command
Read writable data length liable Ring write data
CMD0H_BYTE_WR_GO (3DH)
Such as interrupts, as 1 EH, Continued to write
Y
N
Write complete, return

3.7. Byte read file

Steps are as follows:

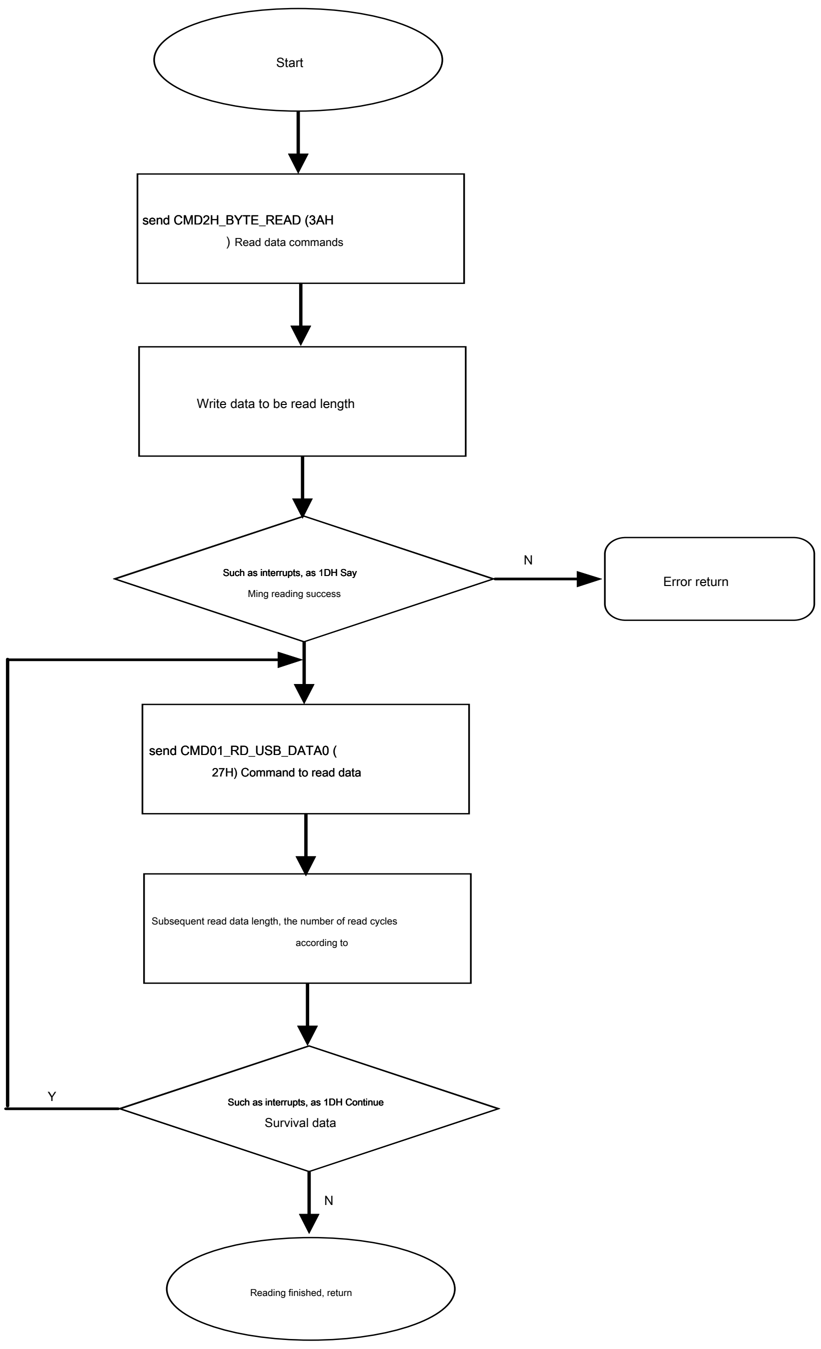
1, transmission CMD2H_BYTE_READ (3AH) bytes of read data command; 2, need to write the

subsequent data transmission length. Write data length is at most 65535;

3, waiting for an interrupt. If the interrupt status is USB_INT_DISK_READ (1DH). Send CMD01_RD_USB_DATA0 (27H) command. Then the length of subsequent data read. If the data length is non-zero, the data read cycle;

4, the read data after completing transmission CMD0H_BYTE_RD_GO (3BH) command. If the interrupt status is USB_INT_DISK_READ (1DH), skip to Step 3 to continue reading. If other interrupt state, the read data is completed;

Flowchart is as follows:


Start
send CMD2H_BYTE_READ (3AH) Read data commands
Write data to be read length
Such as interrupts, as 1DH Say Ming reading success
N
Error return
send CMD01_RD_USB_DATA0 (27H) Command to read data
Subsequent read data length, the number of read cycles according to
Y
Such as interrupts, as 1DH Continue Survival data
N
Reading finished, return

3.8 The sector read files way

Steps are as follows:

1, transmission CMD14_READ_VAR32 (0CH) command, then transmits VAR_FILE_SIZE (68H) data. 4 bytes of data is read again, after taking three bytes of valid data;

2, transmission CMD14_READ_VAR32 (0CH) command, then transmits VAR_FILE_SIZE (68H) data. 4 bytes of data is read again, after taking three bytes of valid data;

3, if the three-byte data acquired in step 2 Step 2 greater than or equal +510 acquiring three-byte data, then the transmission CMD20_WRITE_VAR8 (0BH) command. 6BH and then transmits data 0FFH data, otherwise go to step 5;

4, transmission CMD1H_SEC_READ (4BH) command, and then sends a read data sector number. Wait for interrupt. Proceeds to step 4 if the transmission CMD20_WRITE_VAR8 (0BH) command, and a data transmitting step 6B taken to bit 3 ① The data;

5, if the interrupt status is USB_INT_SUCCESS (14H). Send CMD01_RD_USB_DATA0 (27H) command. Read 5 bytes stored bit 1 ① Read data. This number of sectors the data bit is readable. Otherwise quit. Read data error

6, in the reading 4 bytes, this 4-byte absolute logical sector number. If the read to read the number of sectors read bit 0 is ended;

7, transmission CMD5H_DISK_READ (54H) USB memory read sector command. Step 6 is then sent to the four read absolute logical sector number. Step 6 retransmission number acquired acquired sector;

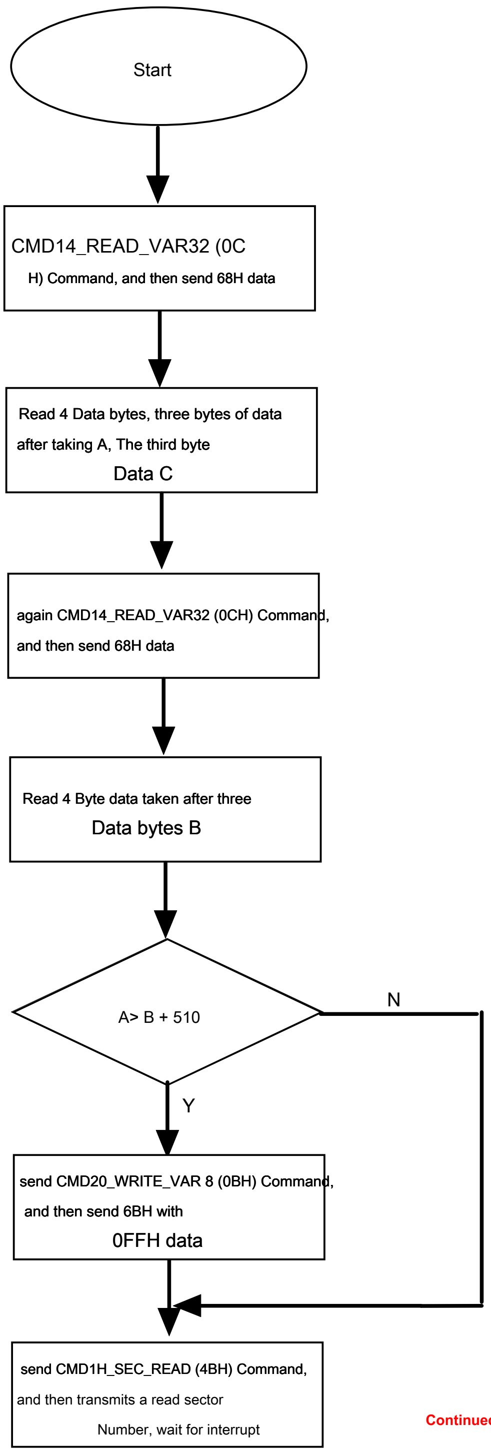
8, to wait for an interrupt, if the interrupt status is USB_INT_DISK_READ (1DH), sending CMD01_RD_USB_DATA0 (27H), re-read data length. Loop reads data according to the data length. After the reading is completed CMD0H_BYTE_RD_GO (3BH) command, if the interrupt status is USB_INT_DISK_READ (1DH), the recycling step 8, otherwise quit reading. If the interrupt status is USB_INT_SUCCESS (14H), data is read successfully;

9, if data is finished, waiting for an interrupt, the interrupt status of 14H, read completion instructions; 10, it is

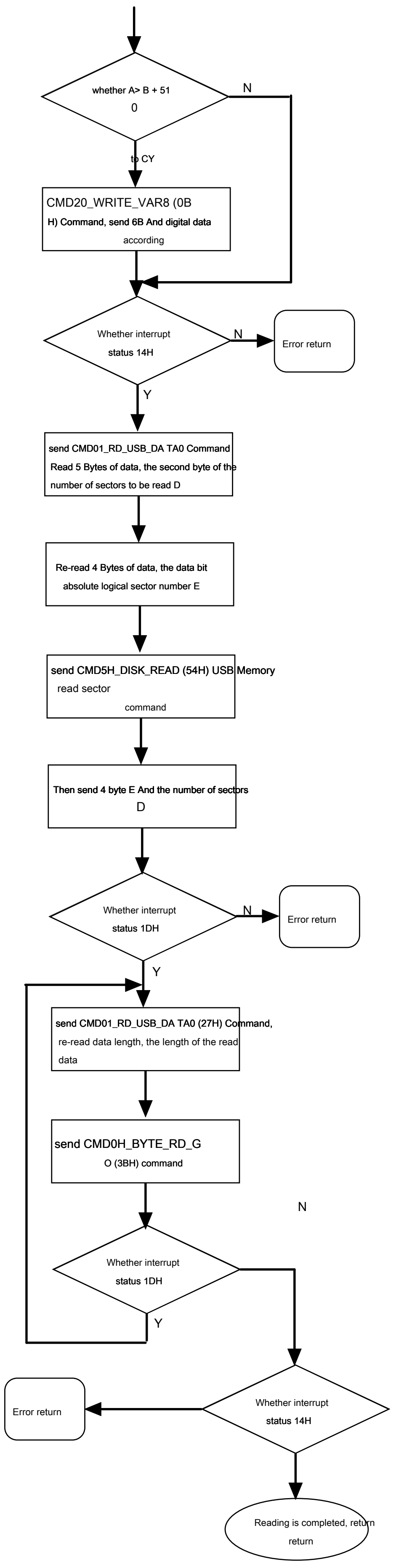
determined if the sector is finished, if it is not finished, then go to step 1;

Note: ① read data starts reading data from the zero position, followed by the first and second of the last to third.

Flowchart is as follows:


Start
CMD14_READ_VAR32 (0C
H) Command, and then send 68H data
Read 4 Data bytes, three bytes of data
after taking A, The third byte
Data C
again CMD14_READ_VAR32 (0CH) Command,
and then send 68H data
Read 4 Byte data taken after three
Data bytes B
A> B + 510
N
Y
send CMD20_WRITE_VAR 8 (0BH) Command,
and then send 6BH with
0FFH data
send CMD1H_SEC_READ (4BH) Command,
and then transmits a read sector
Number, wait for interrupt

Continued on next page


whether A> B + 51
0
N
to CY
CMD20_WRITE_VAR8 (0B
H) Command, send 6B And digital data
according
Whether interrupt
status 14H
N
Error return
Y
send CMD01_RD_USB_DA TA0 Command
Read 5 Bytes of data, the second byte of the
number of sectors to be read D
Re-read 4 Bytes of data, the data bit
absolute logical sector number E
send CMD5H_DISK_READ (54H) USB Memory
read sector
command
Then send 4 byte E And the number of sectors
D
Whether interrupt
status 1DH
N
Error return
Y
send CMD01_RD_USB_DA TA0 (27H) Command,
re-read data length, the length of the read
data
send CMD0H_BYTE_RD_G
O (3BH) command
N
Whether interrupt
status 1DH
Y
Error return
Whether interrupt
status 14H
Reading is completed, return
return

3.9. Sector way to write files

Steps are as follows:

1, transmission CMD1H_SEC_WRITE (4CH), and then sends the write sector number;

2, waiting for an interrupt, the interrupt status if a non-USB_INT_SUCCESS (14H), returned. Otherwise, proceed to Step 3; 3, transmission CMD01_RD_USB_DATA0 (27H) command. Read 5 bytes, the first ① The number of sectors to be written. 4-byte write and then read the absolute logical sector number data. If the number of sectors to be written return 0;

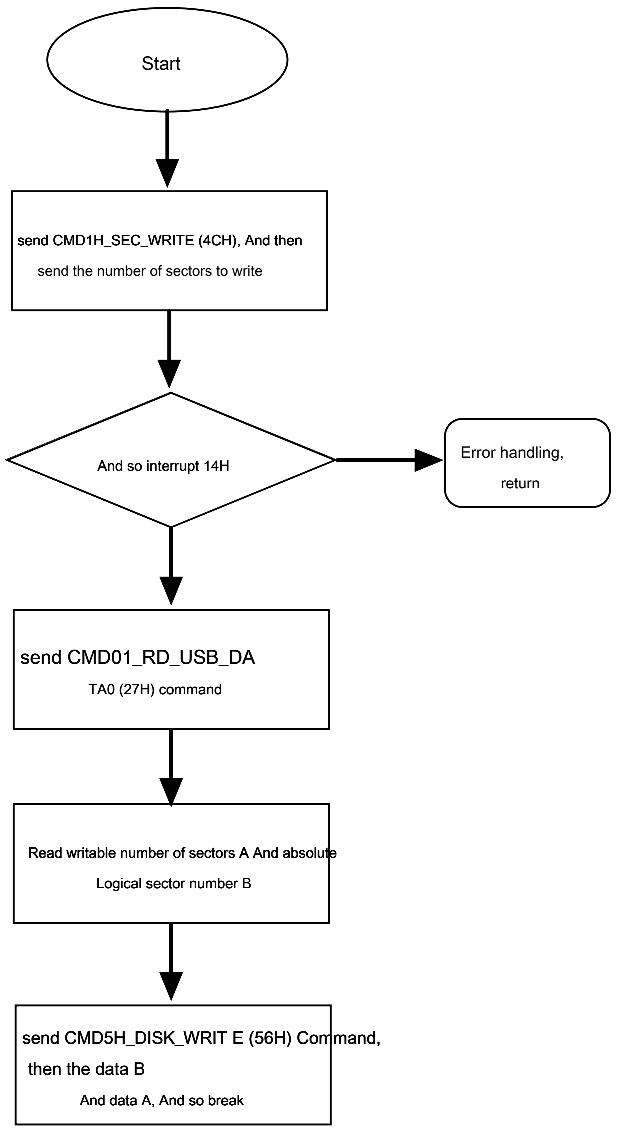
4, transmission CMD5H_DISK_WRITE (56H) command. Step 3 then sends the absolute logical sector number is acquired. Retransmission number of sectors acquired in step 2;

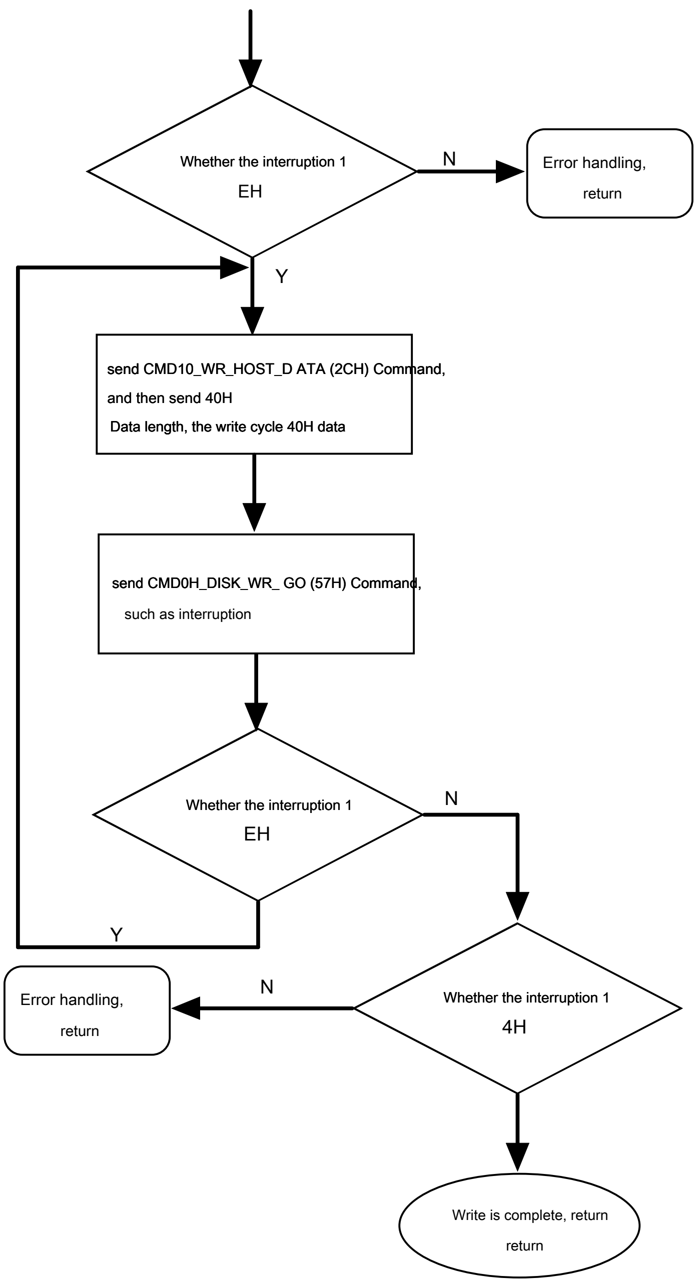
5, waiting for an interrupt, if the interrupt status is USB_INT_DISK_WRITE (1EH), then send CMD10_WR_HOST_DATA (2CH). 40H written next subsequent data length data. 40H data write recirculation. Otherwise;

6, after writing the data, and then transmits CMD0H_DISK_WR_GO (57H) data, skip to step 5;

7, after the data written, waiting for an interrupt. If the interrupt status is USB_INT_SUCCESS (14H), then the write success, or failure;

Note: ① read data starts reading data from the zero position, followed by the first and second of the last to third. Flowchart is as follows:


Start
send CMD1H_SEC_WRITE (4CH), And then send the number of sectors to write
And so interrupt 14H
Error handling, return
send CMD01_RD_USB_DATA0 (27H) command
Read writable number of sectors A And absolute Logical sector number B
send CMD5H_DISK_WRIT E (56H) Command, then the data B And data A, And so break


Whether the interruption 1 EH
N
Error handling, return
Y
send CMD10_WR_HOST_D ATA (2CH) Command, and then send 40H Data length, the write cycle 40H data
send CMD0H_DISK_WR_ GO (57H) Command, such as interruption
Whether the interruption 1 EH
N
Y
Error handling, return
N
Whether the interruption 1 4H
Write is complete, return return

3.10 Close the file

Steps are as follows:

1, transmits a command CMD1H_FILE_CLOSE (36H), then send a byte of data, the data is not the update file indicates a length of 0 and 1 for automatically update the file length;

2, waiting for an interrupt, if the interrupt status is USB_INT_SUCCESS (14H) represents close the file successfully, or fail;

3.11. Create a directory

. 3.11.1 steps to create a directory in the root

directory:

1, CMD_SET_FILE_NAME (2FH) command to set the name of the directory you want to create the directory name ends with the digits 0; 2, set the current directory to create a cluster number.

Send written internal CH376 32-bit variable command CMD50_WRITE_VAR32 (0DH); send to modify the variable name VAR_CURRENT_CLUST (64H);

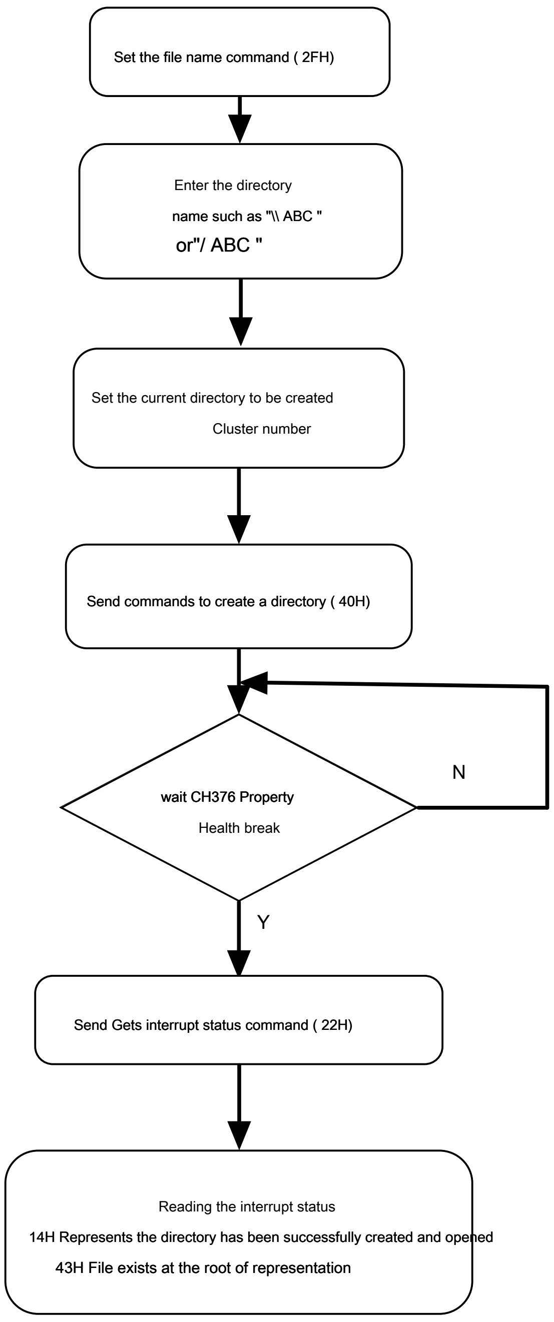
Transmitting the current 32-bit cluster number, LSB first, the root directory is 0; 3, creation of the directory command transmission DIR_CREATE (40H); 4, wait for an interrupt CH376;

5, the transmission interrupt status obtaining command GET_STATUS (22H); 6, reading the interrupt status;

7, if the interrupt status is ERR_FOUND_NAME (43H) indicating the presence of the same name files in the directory;

If the interrupt status is USB_INT_SUCCESS (14H) Contents successfully created and has been opened;

Flowchart is as follows:


Set the file name command (2FH)
Enter the directory name such as "\\ ABC " or"/ ABC "
Set the current directory to be created Cluster number
Send commands to create a directory (40H)
wait CH376 Property Health break
N
Y
Send Gets interrupt status command (22H)
Reading the interrupt status
14H Represents the directory has been successfully created and opened
43H File exists at the root of representation

Note:

1, directory name can not be more than 8 bytes and all the characters must be capital letters, numbers, Chinese characters and some special characters;

2, please refer to the relevant routines CH376EVT \ EXAM \ EXAM9

3.11.2 create a subdirectory in a multi-stage procedure the

following directory:

1, open the directory levels until the last parent directory (the directory you want to create parent directory); 2, read the

starting cluster number of the parent directory (the directory you want to create the parent directory);

32-bit internal transmit the read file system by CH376 system variables command CMD14_READ_VAR32 (0CH); transmitting

variable name VAR_START_CLUSTER (60H) to be read; read starting cluster number (32) of the parent directory, the low byte

first;

3, new directory, with reference to the operation flow 3.11.1 (parent directory has been opened) in the current directory

Note: "/ ABC" \\ ABC or below the root directory represents

"ABC" represents the current directory

4, modify the "Create Directory" to save the "parent directory" starting cluster number;

Send command byte move the file pointer BYTE_LOCATE (39H); transmitting

pointer offset 52 bytes

The writing starting cluster number of the parent directory in a 16-bit high byte and low byte first;

transmission command byte move the file pointer BYTE_LOCATE (39H); offset pointer four bytes

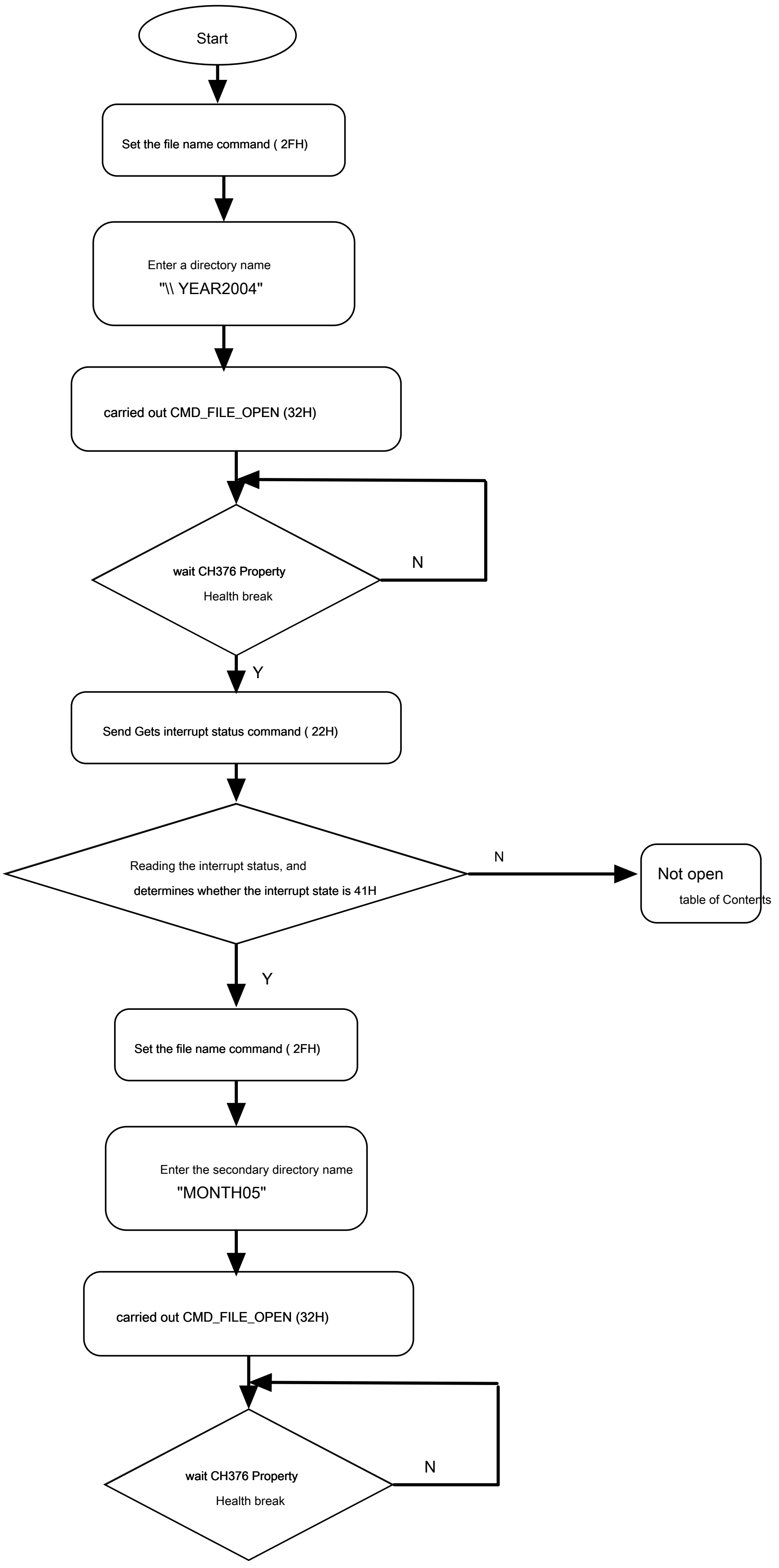
transmitted

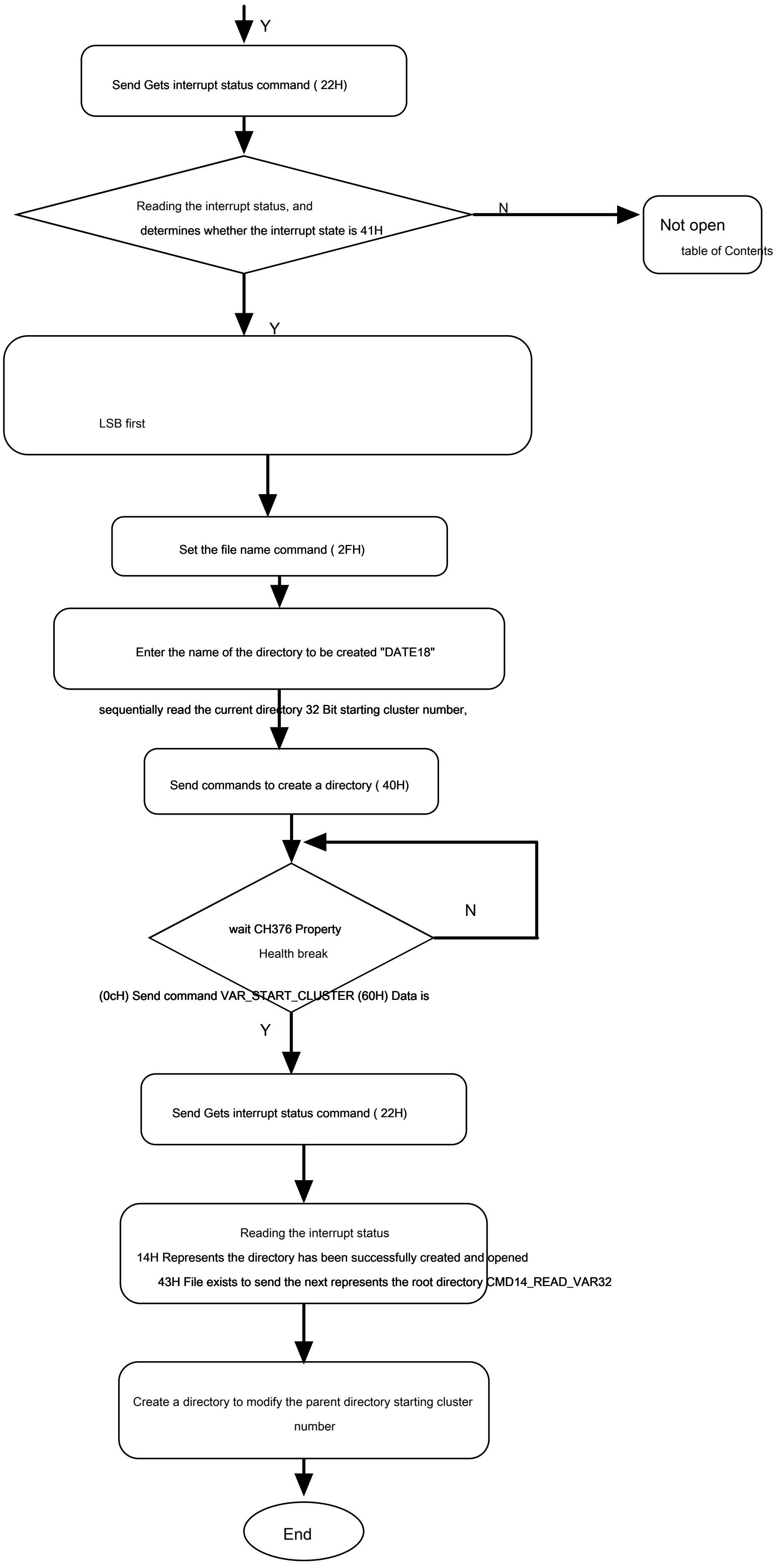
In the low byte write 16-bit starting cluster number of the parent directory, the low byte first; in byte move

the file pointer command BYTE_LOCATE (39H); transmitting pointer offset byte 0 (the file pointer move to

the beginning of the file)

A flow chart is as follows: demo created DATE18 directory \\ YEAR2004 \\ MONTH05 \\ below


Start
Set the file name command (2FH)
Enter a directory name
"\\ YEAR2004"
carried out CMD_FILE_OPEN (32H)
wait CH376 Property
Health break
N
Y
Send Gets interrupt status command (22H)
Reading the interrupt status, and
determines whether the interrupt state is 41H
N
Not open
table of Contents
Y
Set the file name command (2FH)
Enter the secondary directory name
"MONTH05"
carried out CMD_FILE_OPEN (32H)
wait CH376 Property
Health break
N


Y
Send Gets interrupt status command (22H)
Reading the interrupt status, and
determines whether the interrupt state is 41H
N
Not open
table of Contents
Y
LSB first
Set the file name command (2FH)
Enter the name of the directory to be created "DATE18"
sequentially read the current directory 32 Bit starting cluster number,
Send commands to create a directory (40H)
N
wait CH376 Property
Health break
(0cH) Send command VAR_START_CLUSTER (60H) Data is
Y
Send Gets interrupt status command (22H)
Reading the interrupt status
14H Represents the directory has been successfully created and opened
43H File exists to send the next represents the root directory CMD14_READ_VAR32
Create a directory to modify the parent directory starting cluster
number
End

3.12 Modify file attributes

3.12.1. Modify the file creation date and time

1, open the file you want to modify, if the file is already open, you do not need to re-open 2, read the directory information of the file;

The index command is sent to read the file directory information CMD1H_DIR_INFO_READ (37H) to send the file directory information indicates that the current number 0FFH open file; 3, waiting for an interrupt CH376

4, the transmission interrupt status obtaining command GET_STATUS (22H); 5, reading the interrupt status, 14H read the file directory information indicates success 6, converted into 16-bit time data.

/ * File time UINT16 * /

/ * Time = (Hour << 11) + (Minute << 5) + time data file (Second >> 1) * / / * is generated every second designated * /

#define MAKE_FILE_TIME (h, m, s) ((h << 11) + (m << 5) + (s >> 1)) / * file date UINT16 * /

/ * Date = ((Year-1980) << 9) + (Month << 5) + Day * / / * generate the specified date file date data * /

#define MAKE_FILE_DATE (y, m, d) (((y-1980) << 9) + (m << 5) + d) Unsigned short iCreateDate = MAKE_FILE_DATE (2004, 6, 8); Unsigned short iCreateTime = MAKE_FILE_TIME (15 , 39, 20); 7, the creation date written into the internal buffer CH376

Buf [0] = (unsigned char) iCreateDate; Buf [1] = (unsigned char) (iCreateDate >> 8)

Send the specified command buffer address to CH376 internal write data CMD20_WR_OFS_DATA (2EH); transmission offset address 16; write data length subsequent 2; the date of the write transducer Buf [0]; the date of the write transducer Buf [1]; 8, will create a buffer time into the interior of CH376

Buf [0] = (unsigned char) iCreateTime; Buf [1] = (unsigned char) (iCreateTime >> 8)

Send the specified command buffer address to CH376 internal write data CMD20_WR_OFS_DATA (2EH); transmission offset address 14; write data length subsequent 2; Buf time after the write transducer [0]; Buf time after the write transducer [1];

9, a command is sent to save the file directory information CMD0H_DIR_INFO_SAVE (38H); 10, an interrupt is generated to wait CH376

11, send receives interrupt status command GET_STATUS (22H); 12, reads the interrupt status, 14H represents the file directory information saved successfully.

3.12.2. Modify file modification date and time

1, open the file you want to modify, if the file is already open, you do not need to re-open 2, read the directory information of the file;

Send command to read the file directory information CMD1H_DIR_INFO_READ (37H);

Send the file directory information 0FFH index number represents the currently open file; 3, waiting for

an interrupt CH376

4, the transmission interrupt status obtaining command GET_STATUS (22H); 5, reading the

interrupt status, 14H read the file directory information indicates success 6, converted into

16-bit time data.

/ * File time UINT16 * /

/ * Time = (Hour << 11) + (Minute << 5) + time data file (Second >> 1) * / / * is generated

every second designated * /

#define MAKE_FILE_TIME (h, m, s) ((H << 11) + (m << 5) + (s >> 1))

/ * File date UINT16 * /

/ * Date = ((Year-1980) << 9) + (Month << 5) + Day * / / * generate the

specified date file date data * /

#define MAKE_FILE_DATE (y, m, d) (((Y-1980) << 9) + (m << 5) + d)

Unsigned short iModifyDate = MAKE_FILE_DATE (2008, 2, 28); Unsigned short iModifyTime =

MAKE_FILE_TIME (12, 36, 40); 7, the modified date written into the internal buffer CH376

Buf [0] = (unsigned char) iModifyDate; Buf [1] = (unsigned char)

(iModifyDate >> 8)

Send the specified command buffer address to CH376 internal write data CMD20_WR_OFS_DATA (2EH); transmission offset address 24;

write data length subsequent 2; the date of the write transducer Buf [0]; the date of the write transducer Buf [1]; 8, the modification time written

to the internal buffer CH376

Buf [0] = (unsigned char) iModifyTime; Buf [1] = (unsigned char)

(iModifyTime >> 8)

Send the specified command buffer address to CH376 internal write data CMD20_WR_OFS_DATA (2EH); transmission offset address 22;

write data length subsequent 2; Buf time after the write transducer [0]; Buf time after the write transducer [1];

9, a command is sent to save the file directory information CMD0H_DIR_INFO_SAVE (38H); 10, an interrupt is

generated to wait CH376

11, send receives interrupt status command GET_STATUS (22H); 12, reads the interrupt

status, 14H represents the file directory information saved successfully.

/ * FAT file directory information of the data area * /

typedef struct _FAT_DIR_INFO {

UINT8 DIR_Name [11]; / * 00H, file name, a total of 11 bytes, at less than fill space * / UINT8

DIR_Attr; / * 0BH, file attributes, described later with reference to * / UINT8

DIR_NTRes; / * 0CH * / UINT8

DIR_CrtTimeTenth; / * time 0DH, documents created in units of 0.1 seconds count * /

UINT16 DIR_CrtTime; / * 0EH time, file created * / UINT16 DIR_CrtDate; / * date 10H, file created * /

UINT16 DIR_LstAccDate; / * 12H, date of last access operation * / UINT16

DIR_FstClusHI; / * 14H * /

UINT16 DIR_WrtTime; / * 16H, file modification time, with reference to a preceding macroblock MAKE_FILE_TIME * / UINT16

DIR_WrtDate; / * 18H, file modification date, with reference to a preceding macroblock MAKE_FILE_DATE * / UINT16

DIR_FstClusLO; / * 1AH * / UINT32 DIR_FileSize; / * 1CH , file length * /}

FAT_DIR_INFO, * P_FAT_DIR_INFO; / * 20H * /

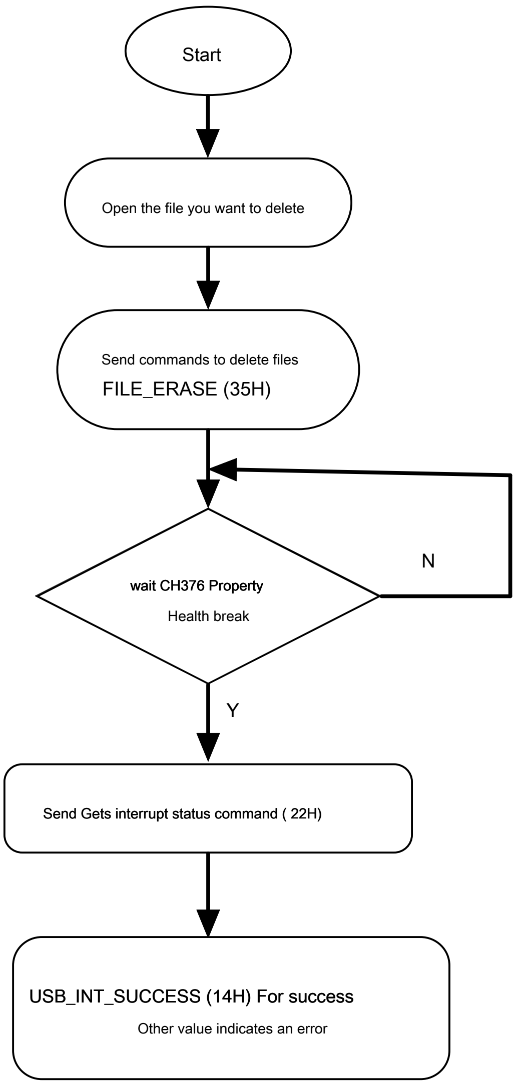
table 1

3.13. Delete File

Steps:

1, open the file you want to delete, for already open files, you can delete 2, sends a command to delete a file FILE_ERASE (35H); 3, waiting for an interrupt CH376

4, transmits an acquisition interrupt status command GET_STATUS (22H); 5, reading the interrupt status, 14H means to delete the file successfully operating procedures:


Start
Open the file you want to delete
Send commands to delete files
FILE_ERASE (35H)
N
wait CH376 Property
Health break
Y
Send Gets interrupt status command (22H)
USB_INT_SUCCESS (14H) For success
Other value indicates an error

Note: To delete a directory and delete files as a method, but if you want to delete a directory, you must put all the files in the directory to delete, and then delete the directory;

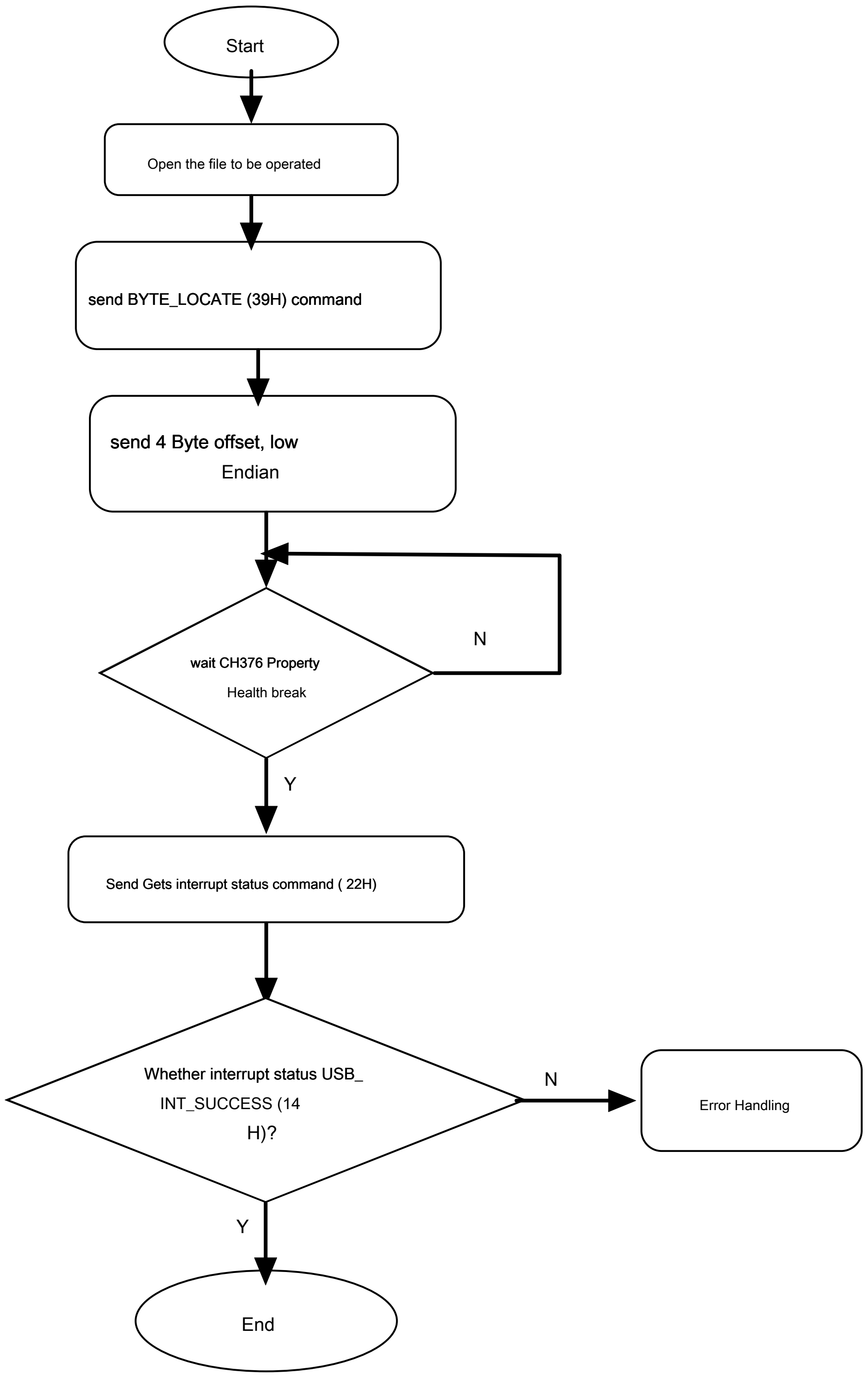
3.14. In byte move the file pointer

Steps:

1, first open the object file, for a file has been opened, it can operate directly; 2, transmitted in bytes of the file pointer movement command BYTE_LOCATE (39H); 3, 4-byte write offset, low word the front section;

0 represents the beginning of the file offset; offset FFFFFFFFH end of file; other values corresponding to the specific location of the file 4, waiting for an interrupt CH376

5, the transmission interrupt status obtaining command GET_STATUS (22H); 6, reading the interrupt status, 14H indicates success; flow:


Start
Open the file to be operated
send BYTE_LOCATE (39H) command
send 4 Byte offset, low Endian
wait CH376 Property Health break
N
Y
Send Gets interrupt status command (22H)
Whether interrupt status USB_ INT_SUCCESS (14 H)?
N
Error Handling
Y
End

NOTE: Move the pointer offset must not exceed the length of the file;

In byte move the file pointer may not be used with a sector manner moves the file pointer;

.

3.15. Manner to move the file pointer sector

Steps:

1, first open the object file, for a file has been opened, it can operate directly; 2, a sector transmitting way to move the file pointer command SEC_LOCATE (4AH); 3, 4-byte transmission sector offset, low endian;

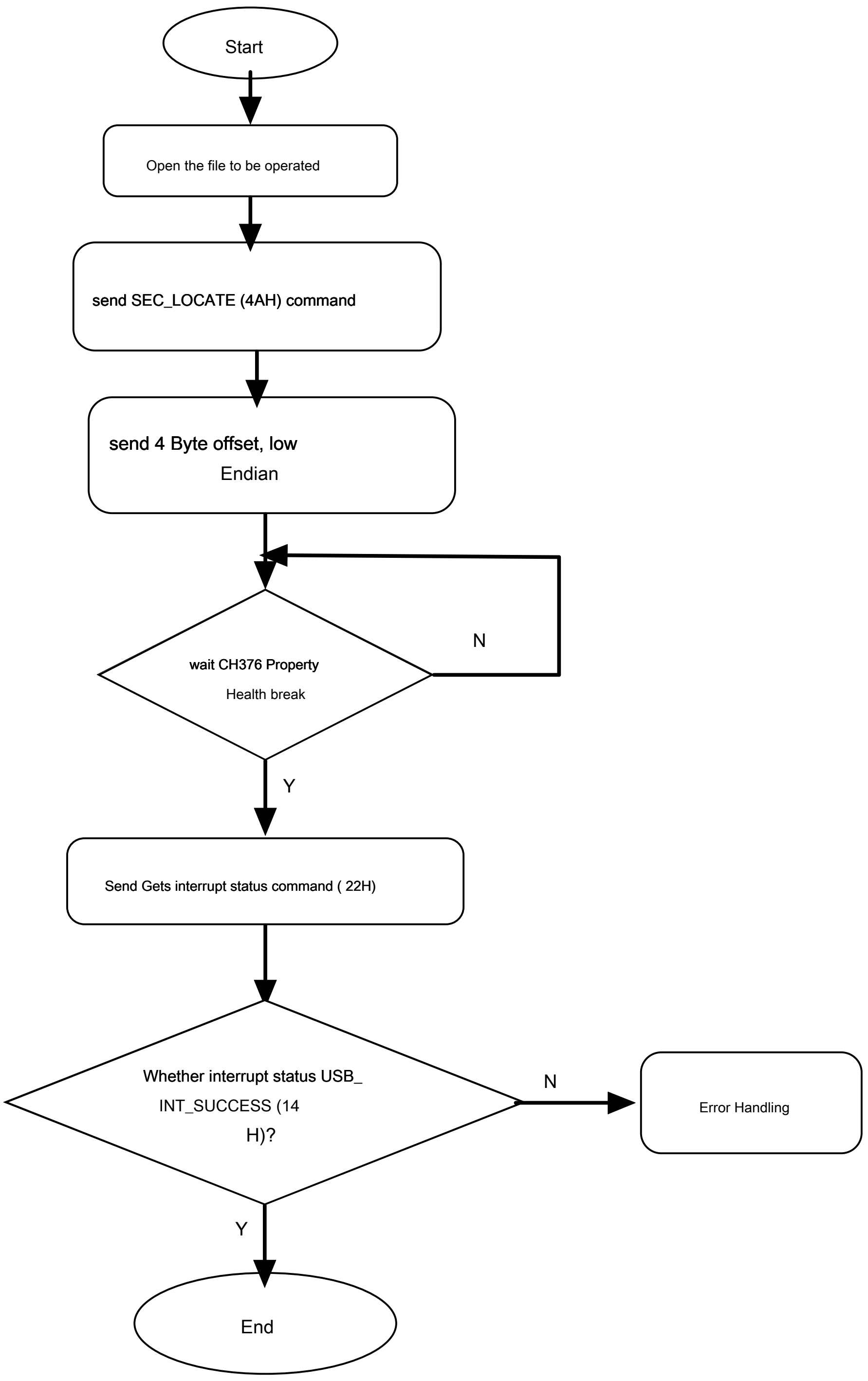
Sector offset of 0 indicates the file pointer to the beginning of the file; FFFFFFFFH shows a sector offset pointer to the end of the file;

Other values of the specific location of the

corresponding file; 4, wait for an interrupt CH376

5, the transmission interrupt status obtaining command GET_STATUS (22H); 6,

reading the interrupt status, 14H indicates success; flow:


Start
Open the file to be operated
send SEC_LOCATE (4AH) command
send 4 Byte offset, low Endian
wait CH376 Property Health break
N
Y
Send Gets interrupt status command (22H)
Whether interrupt status USB_ INT_SUCCESS (14 H)?
N
Error Handling
Y
End

Note: a sector manner move the file pointer, the operation of the SD card does not support;

File pointer offset can not exceed the length of the file;

"In a manner moves the file pointer sector" can not be used in conjunction with "in byte move the file pointer"

3.16. Enumerate files

Steps:

1. Open the need to enumerate files directory, open the directory of the method and the same method to open the file;

"\\" represents the root directory; "\\ ABC" means to open

the ABC directory;

2, SET_FILE_NAME (2FH) command sets need to enumerate and search file names;

Send SET_FILE_NAME (2FH) command to set the file name; send the file

name, can use wildcards "*" "*" indicates that enumerates all the files in the

current directory the following;

"USB *" represents enumerate the current directory of all file names that begin with USB; wildcard "*"

followed not with the characters;

3, sending CMD_FILE_OPEN (32H) command to begin to enumerate files and directories;

4, CH376 compare each file name each time a file is found to meet the requirements, the microcontroller will output an interrupt; 5, waiting for an interrupt

CH376

6, transmits an acquisition interrupt status command GET_STATUS (22H);

7, reading the interrupt status, if the interrupt status is USB_INT_DISK_READ (1DH) enumerations to matching file, and requests the microcontroller reads the file information; if no match to the enumeration file, go to step 13;

8, the microcontroller transmits RD_USB_DATA0 (27H) command reads the directory information of the file;

Transmission RD_USB_DATA0 (27H) command; read

data length of the subsequent return CH376

CH376 subsequent read return transactions; the data format to match the file directory structure of information with reference to

Table 1; 9, the microcontroller continues to send enumeration file directory command CMD_FILE_ENUM_GO (33H); 10, wait for an

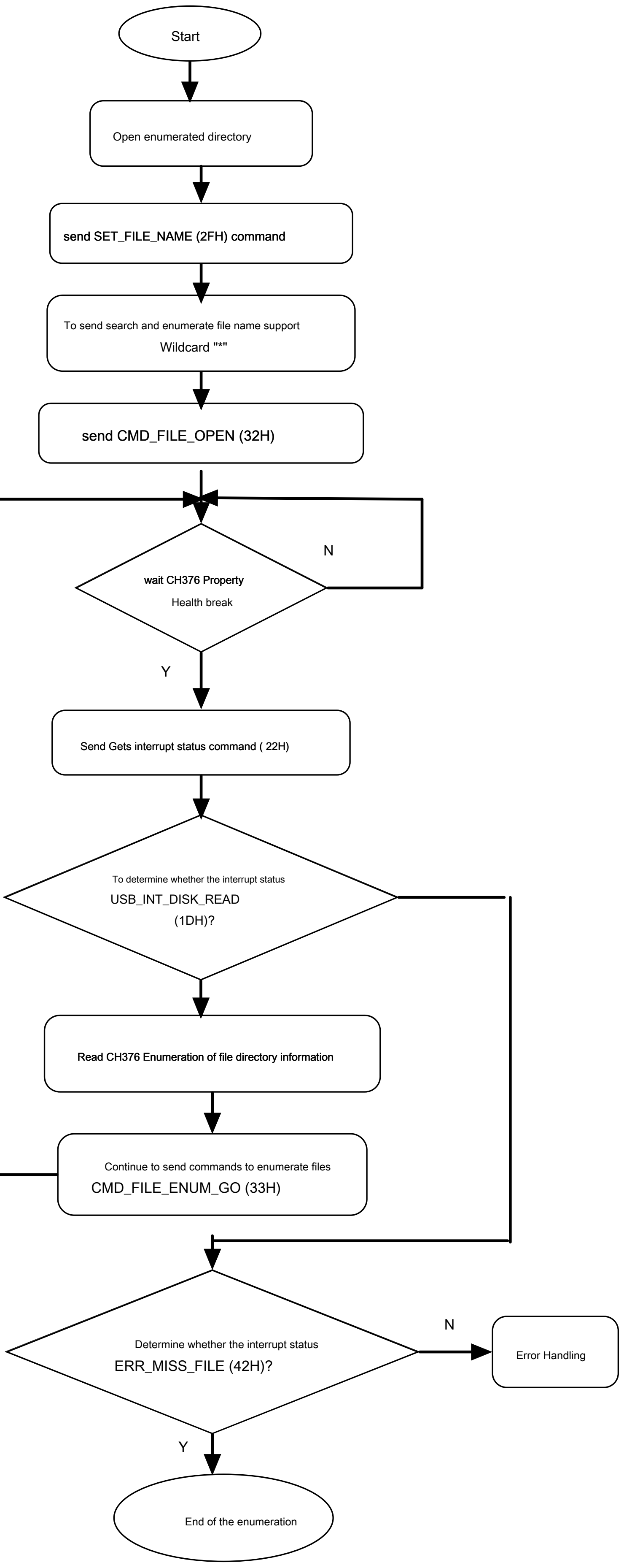
interrupt CH376

11, send receives interrupt status command GET_STATUS (22H);

12, reading the interrupt status, if the interrupt status is USB_INT_DISK_READ (1DH) enumerations to matching file, and requests the microcontroller reads the file information; go to step 8; if not down;

13, CH376 output of the microcontroller interrupt, the interrupt status ERR_MISS_FILE (42H), find no explanation more in line with the requirements of the file, the end of the entire enumeration.

Operating procedures:


Start
Open enumerated directory
send SET_FILE_NAME (2FH) command
To send search and enumerate file name support Wildcard "*"
send CMD_FILE_OPEN (32H)
wait CH376 Property Health break
N
Y
Send Gets interrupt status command (22H)
To determine whether the interrupt status USB_INT_DISK_READ (1DH)?
Read CH376 Enumeration of file directory information
Continue to send commands to enumerate files CMD_FILE_ENUM_GO (33H)
Determine whether the interrupt status ERR_MISS_FILE (42H)?
N
Error Handling
Y
End of the enumeration

Note: CH376EVT \ EXAM \ EXAM13 demonstrates how quickly enumerate the overall file;

3.17. Description of operation long file names

1, to create a long file name, must convert into long file names in UNICODE (little-endian format data); 2, a short file name assigned for long file names (conform capital 8 + 3 format, a digital , Chinese characters and some special characters);

3, for the long file name assigned short file name in the same directory must be a unique correspondence relationship; 4, if the long file name to the file read and write, the file name should be set corresponding to the long file name short file names, long file name has no effect when the file operation.

5, if you want to open a long file name, you must first find the corresponding short file name, and then open the file through its short file name. Delete long file names the same way;

6. long file names can be found with the corresponding short filename by the method of enumeration; 7,

can obtain the long name of the file by the short filename;

Since the operation is more complex long file names, recommends that customers refer to the official program examples provided, CH376EVT \ EXAM \ EXAM11 This example demonstrates how to create a long file name, and how to obtain the corresponding long file names through its short file name;