CH376 编程指南

（U盘和SD卡）

中文手册

http://wch.cn

版本历史

版本	日期	描述
V1.0	2011 年 1 月	第一次修订

目录

1、概述

CH376 是文件管理控制芯片，用于单片机系统读写 U 盘或者 SD 卡中的文件。同时可以使用 CH376 操作 USB 键盘鼠标，以及打印机等各种 USB 设备。该文档主要是介绍 客户端软件如何与 CH376 的固件程序通讯，实现对 U 盘以及 SD 卡的文件操作。

CH376 支持三种通讯接口：8 位并口，SPI 接口或者异步串口；

2、硬件抽象层

2.1. 8 位并口

并口信号线包括：位双向数据总线 D7～D0、读选通输入引脚 RD#、写选通输入引脚 WR#、片选输入引脚 PCS#、中断输出引脚 INT#以及地址输入引脚 A0 对于类似 Intel 并口时序的单片机，CH376 芯片的 RD#引脚和 WR#引脚可以分别连接到单片机的读选通输出引脚和写选通输出引脚。对于类似 Motorola 并口时序的单片机，CH376 芯片的 RD#引脚应该接低电平，并且 WR#引脚连接到单片机的读写方向输出引脚 R/-W。

并口读写时序如下：

```
/* 往 CH376 命令端口写入命令 */
void    xWriteCH376Cmd( UINT8 mCmd )  /* 向 CH376 写命令 */
{
    /*  (*(volatile unsigned char xdata *)0xBDF1) = mCmd ;
    */ /* 通过 51 单片外部并行总线接口操作向 CH376 写入命令 */
    CH376_DATA_DAT_OUT( mCmd );   /* 向 CH376 的并口输出数据 */
    CH376_DATA_DIR_OUT( );   /* 设置并口方向为输出 */
    CH376_A0 = 1;
    CH376_CS = 0;
    CH376_WR = 0;   /* 输出有效写控制信号， 写 CH376 芯片的命令端口 */
    //CH376_CS = 0;   /* 该操作无意义，仅作延时，CH376 要求读写脉冲宽度大于 40nS */
    CH376_WR = 1;   /* 输出无效的控制信号， 完成操作 CH376 芯片 */
    CH376_CS = 1;
    CH376_A0 = 0;
    CH376_DATA_DIR_IN( );   /* 禁止数据输出 */
    /*mDelay0_5uS( );  mDelay0_5uS( );  mDelay0_5uS( ); */
    /* 延时 1.5uS 确保读写周期大于 1.5uS，或者用状态查询代替 */
}

/* 往 CH376 数据端口写入数据 */
void    xWriteCH376Data( UINT8 mData )  /* 向 CH376 写数据 */
{
    /*  (*(volatile unsigned char xdata *)0xBCF0) = mData ;  */ /* 通过 51 单片机外
                            部并行总线接口操作向 CH376 写入数据 */
    CH376_DATA_DAT_OUT( mData );   /* 向 CH376 的并口输出数据 */
    CH376_DATA_DIR_OUT( );         /* 设置并口方向为输出 */
    CH376_A0 = 0;
    CH376_CS = 0;
```

```
    CH376_WR = 0;    /* 输出有效写控制信号， 写 CH376 芯片的数据端口 */
    //CH376_CS = 0;    /* 该操作无意义，仅作延时，CH376 要求读写脉冲宽度大于 40nS */
    CH376_WR = 1;    /* 输出无效的控制信号， 完成操作 CH376 芯片 */
    CH376_CS = 1;
    CH376_DATA_DIR_IN( );    /* 禁止数据输出 */
    //mDelay0_5uS( );          /* 确保读写周期大于 0.6uS */
}

/* 从 CH376 数据端口读取数据 */
UINT8  xReadCH376Data( void )  /* 从 CH376 读数据 */
{
    UINT8    mData;
    /*  mData  = (*(volatile unsigned char xdata *)0xBCF0) ;  */ /* 通过 51 单片机外
                              部并行总线接口操作从 CH376 读取数据 */
    //mDelay0_5uS( );        /* 确保读写周期大于 0.6uS */
    CH376_DATA_DIR_IN( );    /* 设置并口方向为输入 */
    CH376_A0 = 0;
    CH376_CS = 0;
    CH376_RD = 0;    /* 输出有效读控制信号， 读 CH376 芯片的数据端口 */
    CH376_CS = 0;    /* 该操作无意义，仅作延时，CH376 要求读写脉冲宽度大于 40nS*/
    mData = CH376_DATA_DAT_IN( ); /* 从 CH376 的并口输入数据 */
    CH376_RD = 1;                 /* 输出无效的控制信号， 完成操作 CH376 芯片 */
    CH376_CS = 1;
    return( mData );
}

/* 从 CH376 命令端口读取状态 */
UINT8    xReadCH376Status( void )  /* 从 CH376 读状态，仅用于并口方式 */
{
    UINT8   mData;
    /*  mData  = (*(volatile unsigned char xdata *)0xBDF1) ;  */ /* 通过 51 单片机外
                              部并行总线接口操作从 CH376 读取状态 */
    CH376_DATA_DIR_IN( );    /* 设置并口方向为输入 */
    CH376_A0 = 1;
    CH376_CS = 0;
    CH376_RD = 0;    /* 输出有效读控制信号， 读 CH376 芯片的状态端口 */
    CH376_CS = 0;    /*该操作无意义，仅作延时，CH376 要求读写脉冲宽度大于 40ns*/
    mData = CH376_DATA_DAT_IN( ); /* 从 CH376 的并口输入数据 */
    CH376_RD = 1;                 /* 输出无效的控制信号， 完成操作 CH376 芯片 */
    CH376_CS = 1;
    CH376_A0 = 0;
    return( mData );
}
```

2.2. 4 线串口(SPI)

SPI 同步串行接口信号线包括：SPI 片选输入引脚 SCS、串行时钟输入引脚 SCK、串行数据输入引脚 SDI、串行数据输出引脚 SDO、中断输出引脚 INT#以及接口忙状态输出引脚 BZ(可省略)。

1、CH376 支持 SPI 模式 0 和模式 3(CH376 在 SCK 的上升沿采样数据，在 SCK 的下降沿输出数据)；

2、CH376 支持 SPI 的时钟速度最高 24MHZ；

3、CH376 工作在 SPI 模式时，将 SCS 由高变为低之后的第一个字节作为命令处理；在一个命令周期处理结束之后单片机应该将 SCS 置为高电平；

以下是使用 51 单片机 IO 口模拟 SPI 时序：

```
void   Spi376OutByte( UINT8 d )  /* SPI 输出 8 个位数据 */
{
    /* 如果是硬件 SPI 接口，应该是先将数据写入 SPI 数据寄存器，然后查询 SPI 状态寄存器以等待 SPI 字节传输完成 */
    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;              /* 数据位是高位在前 */
        CH376_SPI_SCK = 1;    /* CH376 在时钟上升沿采样输入 */
    }
}

UINT8    Spi376InByte( void )  /* SPI 输入 8 个位数据 */
{
    /* 如果是硬件 SPI 接口，应该是先查询 SPI 状态寄存器以等待 SPI 字节传输完成，然后从 SPI 数据寄存器读出数据 */
    UINT8   i,  d;
    d = 0;
    for ( i = 0;  i < 8;  i ++ )
    {
        CH376_SPI_SCK = 0;    /* CH376 在时钟下降沿输出 */
        d <<= 1;              /* 数据位是高位在前 */
        if ( CH376_SPI_SDO ) d ++;
        CH376_SPI_SCK = 1;
    }
    return( d );
}

#define    xEndCH376Cmd( )
{
    CH376_SPI_SCS = 1;
```

```
}
/* SPI 片选无效，结束 CH376 命令，仅用于 SPI 接口方式 */

void   xWriteCH376Cmd( UINT8 mCmd )  /* 向 CH376 写命令 */
{
    CH376_SPI_SCS = 1;   /* 防止之前未通过 xEndCH376Cmd 禁止 SPI 片选 */
    mDelay0_5uS( );
    /* 对于双向 I/O 引脚模拟 SPI 接口，那么必须确保已经设置 SPI_SCS，SPI_SCK，SPI_SDI
    为输出方向，SPI_SDO 为输入方向 */
    CH376_SPI_SCS = 0;       /* SPI 片选有效 */
    Spi376OutByte( mCmd );   /* 发出命令码 */
    mDelay0_5uS( );  mDelay0_5uS( );  mDelay0_5uS( );
    /* 延时 1.5uS 确保读写周期大于 1.5uS，或者用上面一行的状态查询代替*/
}

void    xWriteCH376Data( UINT8 mData )  /* 向 CH376 写数据 */
{
    Spi376OutByte( mData );
    //mDelay0_5uS( );        /* 确保读写周期大于 0.6uS */
}
UINT8   xReadCH376Data( void )  /* 从 CH376 读数据 */
{
    //mDelay0_5uS( );        /* 确保读写周期大于 0.6uS */
    return( Spi376InByte( ) );
}
```

2.3. 2 线串口(UART)

异步串口信号线包括：串行数据输入引脚 RXD 和串行数据输出引脚 TXD 以及中断输出引脚 INT#。

1、CH376 芯片的 RXD 和 TXD 可以分别连接到单片机的串行数据输出引脚和串行数据输入引脚。

2、CH376 的串行数据格式是标准的字节传输模式，包括 1 个起始位、8 个数据位、1 个停止位。

3、CH376 默认的波特率为 9600，支持单片机随时通过 CMD_SET_BAUDRATE 命令选择合适的通讯波特率。每次上电复位后，CH376 默认的串行通讯波特率由 BZ/D4，SCK/D5，SDI/D6 三个引脚的电平组合设定(详见 CH376DS1 6.4 节)

4、CH376 发送命令是以两个字节的同步码开始(57H， ABH ，命令码)

以下是使用 51 单片机的串口时序为例：

```
void   xWriteCH376Cmd( UINT8 mCmd )  /* 向 CH376 写命令 */
{
    TI = 0;
    SBUF = 0x57;   /* 启动操作的第 1 个串口同步码 */
    while ( TI == 0 );
    TI = 0;
```

```
    SBUF = 0xAB;    /* 启动操作的第 2 个串口同步码 */
    while ( TI == 0 );
    TI = 0;
    SBUF = mCmd;    /* 串口输出 */
    while ( TI == 0 );
}

void   xWriteCH376Data( UINT8 mData )  /* 向 CH376 写数据 */
{
    TI = 0;
    SBUF = mData;    /* 串口输出 */
    while ( TI == 0 );
}

UINT8   xReadCH376Data( void )  /* 从 CH376 读数据 */
{
    UINT32  i;
    for ( i = 0;  i < 500000;  i ++ ) /* 计数防止超时 */
    {
        if ( RI )               /* 串口接收到 */
        {
            RI = 0;
            Return( SBUF );    /* 串口输入 */
        }
    }
    return( 0 );          /* 不应该发生的情况 */
}
```

2.4. CH376 中断引脚(INT#) 的使用

CH376 芯片 INT#引脚输出的中断请求默认是低电平有效，可以连接到单片机的中断输入引脚或普通输入引脚，单片机可以使用中断方式或查询方式获知 CH376 的中断请求。

为了节约引脚，单片机可以不连接 CH376 的 INT#引脚，而通过其它方式获知中断。

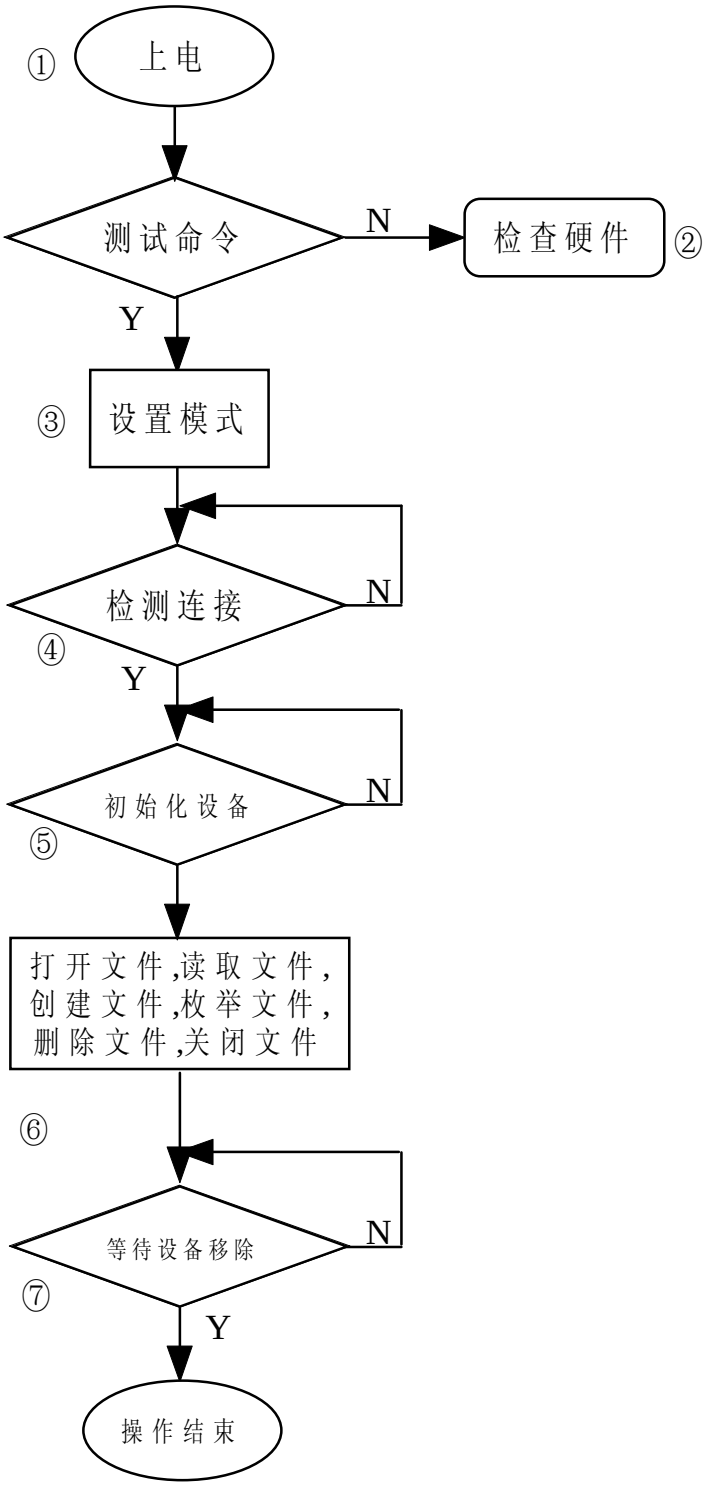
并口方式： 可以通过 xReadCH376Status( )函数读取并行端口的最高位，该位等效 于中断引脚的状态，最高位为低，表示有中断产生；

SPI 接口方式：可以通过发送 SET_SDO_INT(0BH)命令 设置 CH376 的 SDO 引脚在片选(SCS)无效时 兼做 INT#引脚使用；在使用硬件 SPI 接口时不推荐使用；

串口方式：当 CH376 产生中断的时候，会通过 CH376 的 TXD 发送一个状态码给单片机，表示有中断产生，单片机可以查询接收该状态码；

3、系统应用层

芯片操作主流程图下：



注：

①、上电延时 50MS；

②、发送 CMD_CHECK_EXIST(06H)命令，发送 55H 数据，正确返回 0AAH。非 0AAH 数据，检查 MCU 和 CH376 硬件连线以及 CH376 复位和晶振是否起振；

③、发送 CMD_SET_USB_MODE(15H)命令，后续数据为模式.U 盘为模式 6，SD 卡为模式 3。返回状态为 CMD_RET_SUCESS(51H)，此状态选择性读取；

④、发送 CMD_DISK_CONNECT(30H)命令，产生中断返回中断状态 USB_INT_CONNECT(15H)，设备连接，SD 卡检测设备连接通过 SD 卡座上第 10 脚低表示 SD 连接，高表示未连接；

⑤、发送 CMD_DISK_MOUNT(31H)命令，产生中断返回中断状态 USB_INT_SUCESS(14H)，初始化完成。如果失败，需要重复 5 次，每次重试之前需要加 200MS 延时；

⑥、参考每步骤详细流程图；

⑦、发送 CMD_DISK_CONNECT(30H)命令，产生中断，返回中断状态 USB_INT_DISCONNECT(16H)，设备移除。如果返回中断状态为 USB_INT_CONNECT(15H)，则继续等待；

注：本指南只针对 CH376 做主机模式，设备模式请参考 CH372 说明手册；

3.1. 芯片初始化

CH376 芯片进行初始化之前，需要发送 CMD_CHECK_EXIST 命令用于检查通信接口和工作状态。检查 CH376 芯片是否正常工作。当 CH376 芯片正常工作。即可对 CH376 芯片进行初始化。MCU 需要发送 CMD_SET_USB_MODE 命令初始化 CH376 芯片。该命令需要需要输入一个数据，该数据是模式代码：

模式代码为 03H 时切换到 SD 卡主机模式，管理和存取 SD 卡中的文件；

模式代码为 06H 时切换到已启用的 USB 主机方式，自动产生 SOF 包；

关于 USB 设备方式请参考 CH372 手册，CH376 的 USB 设备方式与 CH372 芯片完全兼容。

通常情况下，设置 USB 模式在 10uS 时间之内完成，完成后输出状态，正常情况下返回状态为 51H。实际操作过程中，只要 CMD_CHECK_EXIST 命令检查通过，设置模式命令可以不需要读取状态。

3.2. 查询设备连接

模式设置完成之后，需要检测设备连接。SD 卡和 U 盘检测连接方式如下：

SD 卡检测连接方式：

SD 卡座第 10 引脚用来检测 SD 卡是否插入。SD 卡未插入时，SD 卡座第 10 脚电压为高电平，SD 卡插入后为低电平。

SD 卡座的第 11 引脚用来检测 SD 卡是否写保护。高电平为未写保护，低电平为写保护。

U 盘检测连接方式：

U 盘检测连接需要发送 CMD_DISK_CONNECT 命令。此命令发送完成，产生中断给 MCU。如果中断状态为 USB_INT_SUCCESS。那么说明有 U 盘连接。

3.3. 初始化设备

当MCU检测到U盘或者SD卡连接后。需要对U盘、SD卡进行初始化，发送CMD_DISK_MOUNT。此命令如果返回中断状态为USB_INT_SUCCESS，说明对U盘或者SD卡初始化完成。对某些U盘或者SD卡，可能需要重复多次发送此命令才能完成对U盘、SD卡的初始化。每次重试之前加200MS左右延时。

3.4. 打开文件

3.4.1. 根目录或者当前目录下打开文件或者目录

操作步骤如下：

1、发送CMD10_SET_FILE_NAME(2FH)命令设置文件名；

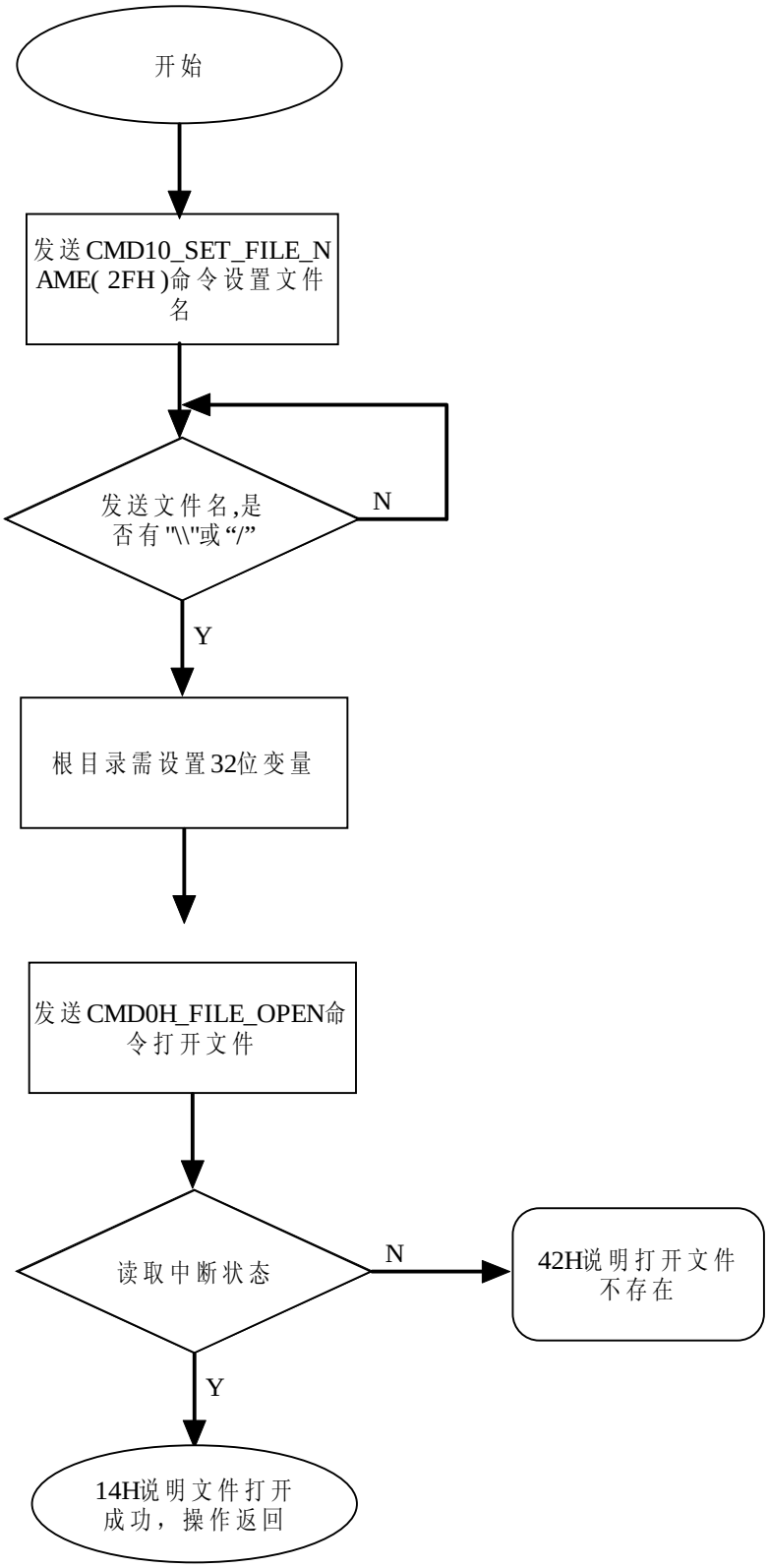
2、发送文件名，文件名以数字0结束；

3、当文件路径里有“\\”或者“/”时，结束发送文件名；

4、如果为根目录文件，需要发送设置CH376芯片内部32位变量。发送CMD50_WRITE_VAR32(0DH)，接着写数据VAR_CURRENT_CLUST(64H)，接着写32位0数据；

5、发送打开文件命令CMD0H_FILE_OPEN(32H)，打开文件，如果为中断中断状态为USB_INT_SUCCESS(14H)，文件打开成功，如果中断状态为ERR_MISS_FILE(42H)说明打开的文件不存在；

流程图如下：



3.4.2. 打开多级目录下的文件

操作步骤如下：

1、首先找到下级目录的标示符“\\”或者“/”；

2、找到当前文件的文件名，如果为根目录，打开文件时，需要加上“\\”或者“/”；

3、循环调用“根目录或者当前目录下打开文件或者目录”直到文件被打开。

打开文件举例：

打开文件完整路径为“\\ABC\\123\\WCH.TXT”。那么首先打开“\\ABC” 文件夹，接着在打开“123”文件夹，最后在打开“WCH.TXT”文件。

3.5. 创建文件

3.5.1. 根目录或者当前目录下创建文件

操作步骤如下：

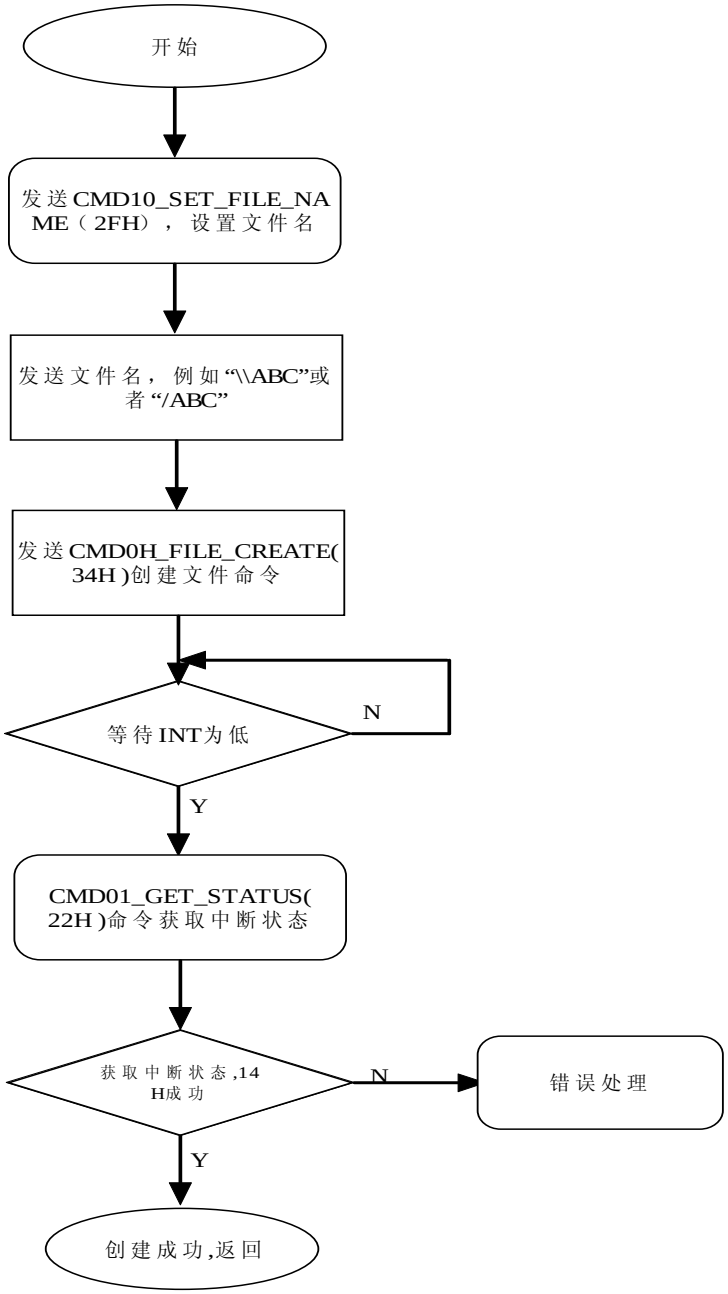
1、发送CMD10_SET_FILE_NAME（ 2FH ），设置文件名；

2、接着发送文件名(根目录创建文件名需要加“\\”或者“/”)；

3、上述发送完成后，发送CMD0H_FILE_CREATE（ 34H ），创建文件；

4、等待CH376芯片中断，当中断引脚为低电平，发送CMD01_GET_STATUS（ 22H ）命令，获取中断状态，如果中断中断状态为USB_INT_SUCCESS（ 14H ）,说明创建文件成功；

流程图如下：



3.5.2. 多级目录下创建文件

操作步骤如下：

1、先找到最后一级文件的文件名。例如在“ABC”文件夹上面创建一个文件“123.TXT”，则创建文件的完整路径为“//ABC//123.TXT”那么，最后一级的文件名为“123.TXT”。不需要“\\”；

2、循环调用“根目录或者当前目录下打开文件或者目录”；

3、调用“根目录或者当前目录下创建文件”创建文件；

3.6. 字节方式写文件

操作步骤如下：

1、发送 CMD2H_BYTE_WRITE(3CH)字节写命令；

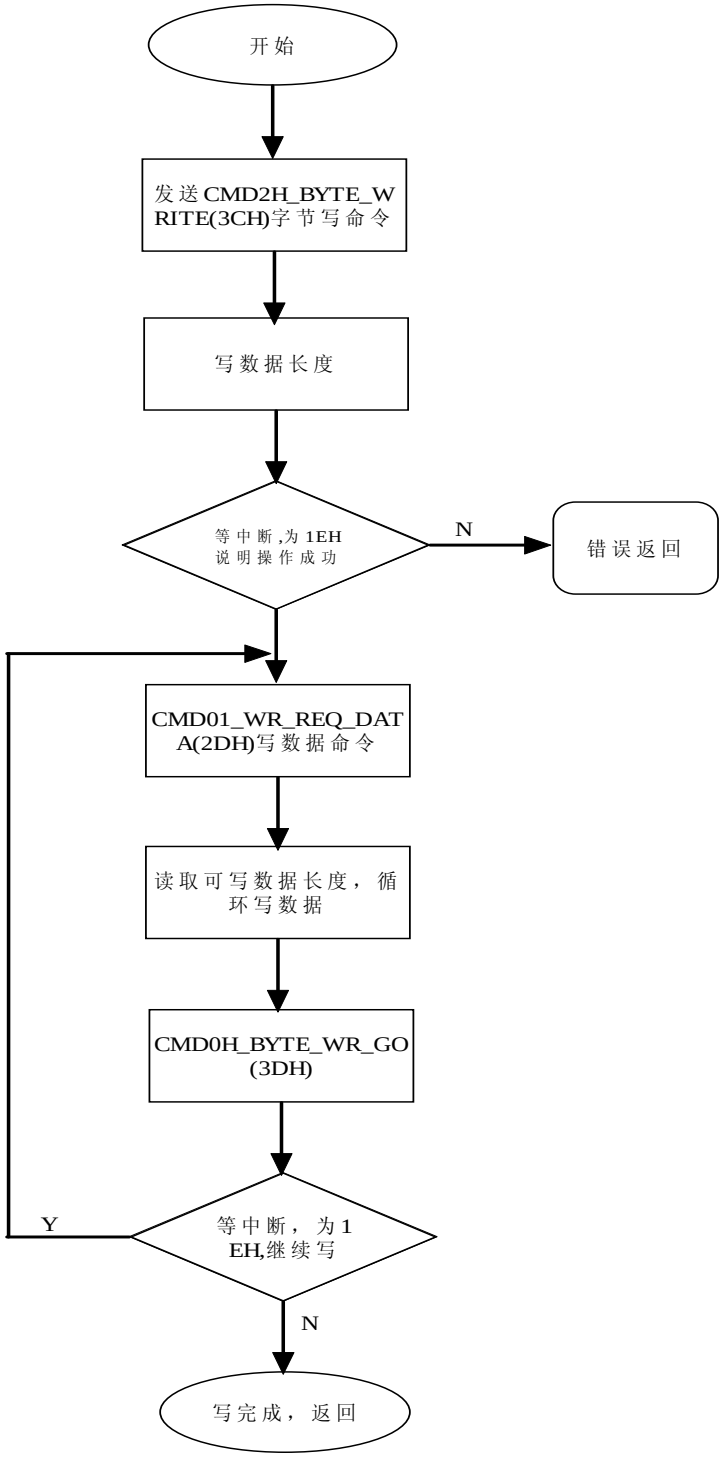
2、接着写需要写后续字节写数据长度。最多支持字节数位 65535 个字节；

3、等待中断，如果中断状态为 USB_INT_DISK_WRITE(1EH)；

4、发送 CMD01_WR_REQ_DATA(2DH)命令，读取可以往 CH376 数据长度。接着循环写数据；

5、写完数据之后，发送 CMD0H_BYTE_WR_GO(3DH)，等待中断，如果中断状态为 USB_INT_DISK_WRITE(1EH)则重新调用步骤 4 循环写数据。如果其他中断状态，则写完成；

流程图如下：



3.7. 字节方式读文件

操作步骤如下：

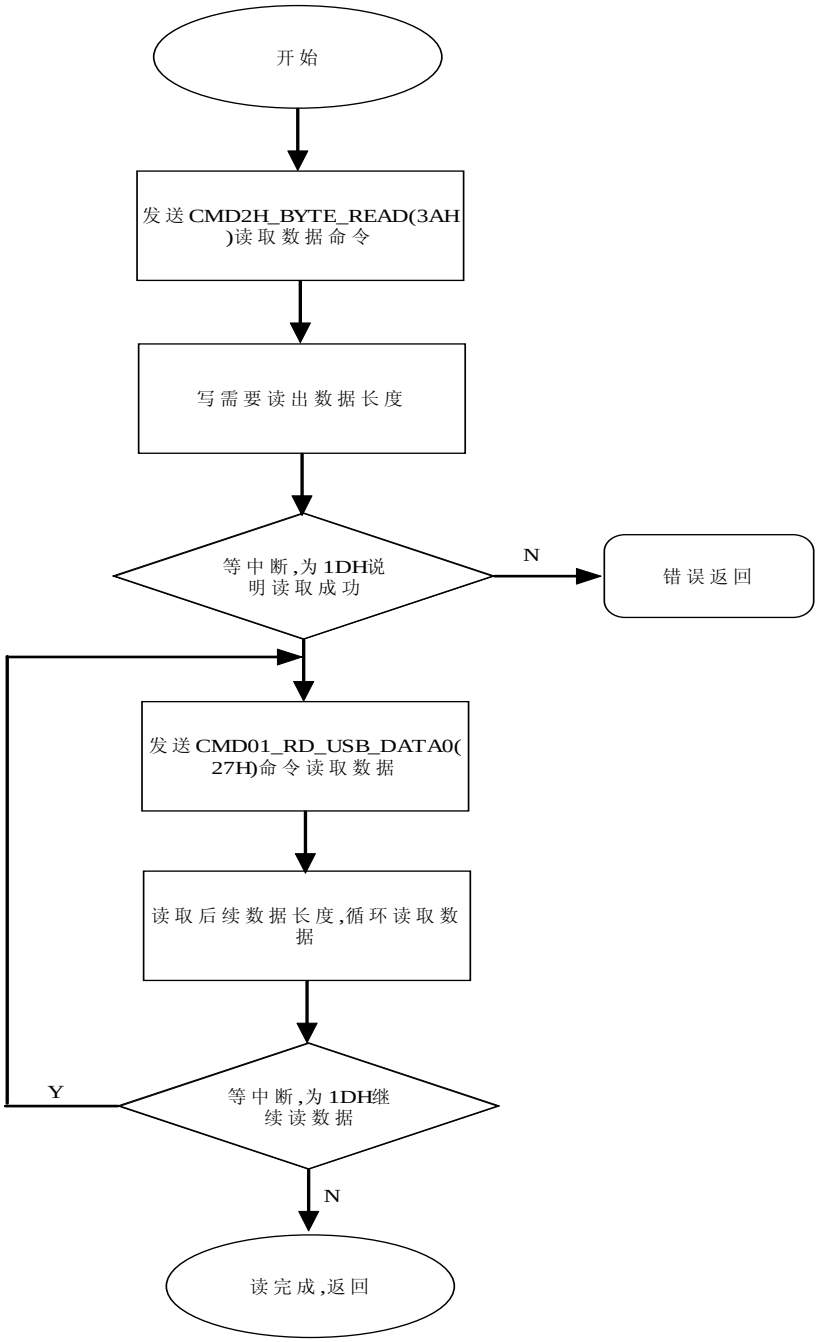
1、发送 CMD2H_BYTE_READ(3AH)字节读数据命令；

2、发送需要写后续数据长度。最多写数据长度为 65535；

3、等待中断。如果中断状态为 USB_INT_DISK_READ(1DH)。则发送 CMD01_RD_USB_DATA0(27H)命令。接着读后续数据长度。如果长度为非 0 数据，循环读取数据；

4、读取完数据之后发送 CMD0H_BYTE_RD_GO(3BH)命令。如果中断状态为 USB_INT_DISK_READ(1DH)，则跳到步骤 3 继续读。如果为其他中断状态，则读取数据完成；

流程图如下：



3.8. 扇区方式读文件

操作步骤如下：

1、发送 CMD14_READ_VAR32(0CH)命令，接着发送 VAR_FILE_SIZE(68H)数据。再读取 4 个字节数据，取后三个字节为有效数据；

2、发送 CMD14_READ_VAR32(0CH)命令，接着发送 VAR_FILE_SIZE(68H)数据。再读取 4 个字节数据，取后三个字节为有效数据；

3、如果步骤 2 获取三个字节数据大于等于步骤 2 获取三个字节数据+510，则发送 CMD20_WRITE_VAR8(0BH)命令。接着发送 6BH 数据和 0FFH 数据，否则跳到步骤 5；

4、发送 CMD1H_SEC_READ(4BH)命令，再发送读取扇区数数据。等待中断。如果进入步骤 4 发送 CMD20_WRITE_VAR8(0BH)命令，发送 6B 数据和 1 步骤取到的第 3 位①的数据；

5、如果中断状态为 USB_INT_SUCCESS(14H)。则发送 CMD01_RD_USB_DATA0(27H)命令。读取 5 个字节，保存第 1 位①读取到的数据。此数据位为可读取的扇区数。否则退出。读取数据错误

6、在读取 4 个字节，此 4 个字节为绝对逻辑扇区号。如果读取到的可读取扇区数位 0 则结束读取；

7、发送 CMD5H_DISK_READ(54H) USB 存储器读扇区命令。接着发送步骤 6 读取到的 4 位绝对逻辑扇区号。再发送步骤 6 获取到的获取到的扇区数；

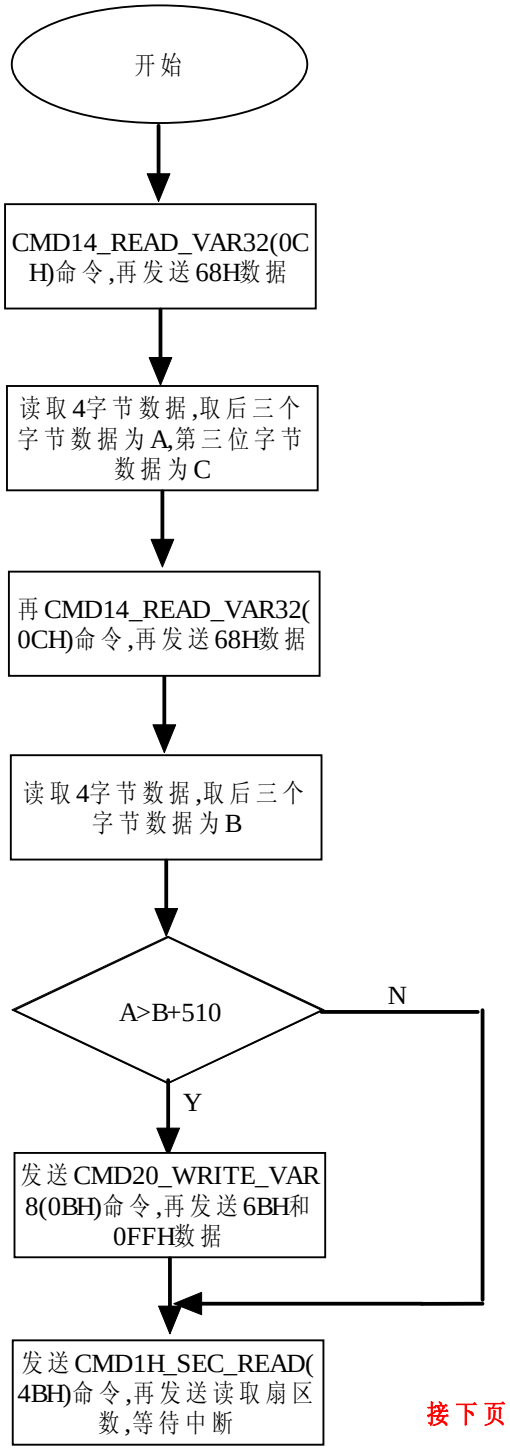
8、等待中断，如果中断状态为 USB_INT_DISK_READ(1DH)，则发送 CMD01_RD_USB_DATA0(27H)，再读取数据长度。根据数据长度循环读取数据。读取完成后 CMD0H_BYTE_RD_GO(3BH)命令，如果中断状态为 USB_INT_DISK_READ(1DH)，则循环步骤 8，否则退出读取。如果中断状态为 USB_INT_SUCCESS(14H)，则读取数据成功；

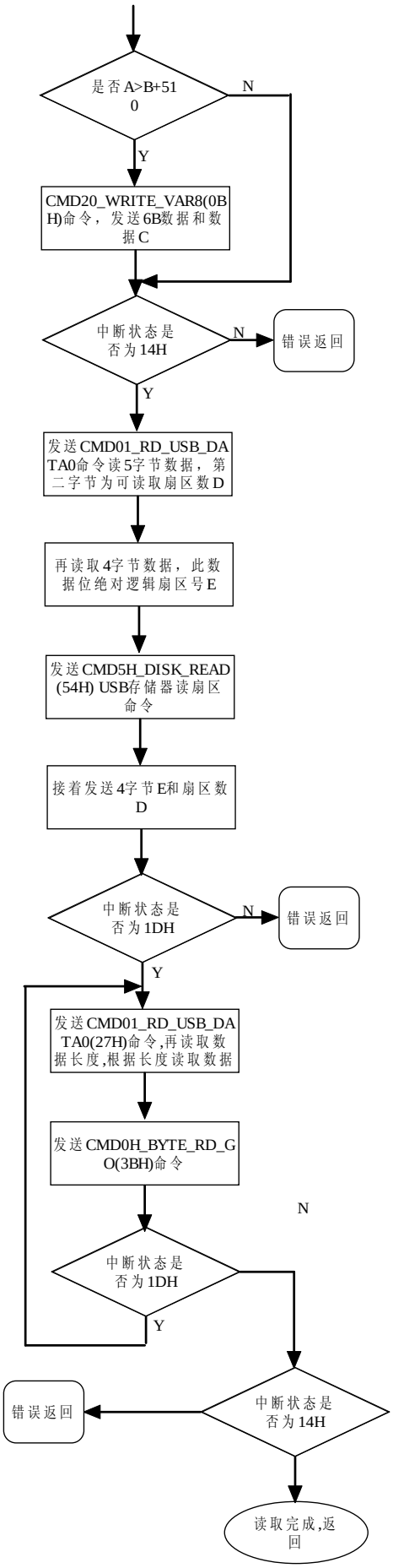
9、如果写完数据后，等待中断，中断状态为 14H，说明读取完成；

10、判断扇区是否写完，如果未写完，则跳到步骤 1；

注：① 读取数据，从第零位数据开始读取，之后为第一位，然后为第二位，最后为第三位。

流程图如下：


开始
CMD14_READ_VAR32(0CH)命令,再发送68H数据
读取4字节数据,取后三个字节数据为A,第三位字节数据为C
再CMD14_READ_VAR32(0CH)命令,再发送68H数据
读取4字节数据,取后三个字节数据为B
A>B+510
N
Y
发送CMD20_WRITE_VAR8(0BH)命令,再发送6BH和0FFH数据
发送CMD1H_SEC_READ(4BH)命令,再发送读取扇区数,等待中断
接下页


是否A>B+510
N
Y
CMD20_WRITE_VAR8(0BH)命令，发送6B数据和数据C
中断状态是否为14H
N
错误返回
Y
发送CMD01_RD_USB_DATA0命令读5字节数据，第二字节为可读取扇区数D
再读取4字节数据，此数据位绝对逻辑扇区号E
发送CMD5H_DISK_READ(54H) USB存储器读扇区命令
接着发送4字节E和扇区数D
中断状态是否为1DH
N
错误返回
Y
发送CMD01_RD_USB_DATA0(27H)命令,再读取数据长度,根据长度读取数据
发送CMD0H_BYTE_RD_GO(3BH)命令
N
中断状态是否为1DH
Y
中断状态是否为14H
错误返回
读取完成,返回

3.9. 扇区方式写文件

操作步骤如下：

1、发送 CMD1H_SEC_WRITE(4CH)，再发送写扇区数；

2、等待中断，如果中断状态为非 USB_INT_SUCCESS(14H)，返回。否则进入步骤 3；

3、发送 CMD01_RD_USB_DATA0(27H)命令。读取 5 个字节，第一位①为可写的扇区数。再读取 4 字节为写数据绝对逻辑扇区号。如果可写扇区数为 0 则返回；

4、发送 CMD5H_DISK_WRITE(56H)命令。接着发送步骤 3 获取到的绝对逻辑扇区号。再发送步骤 2 获取到的扇区数；

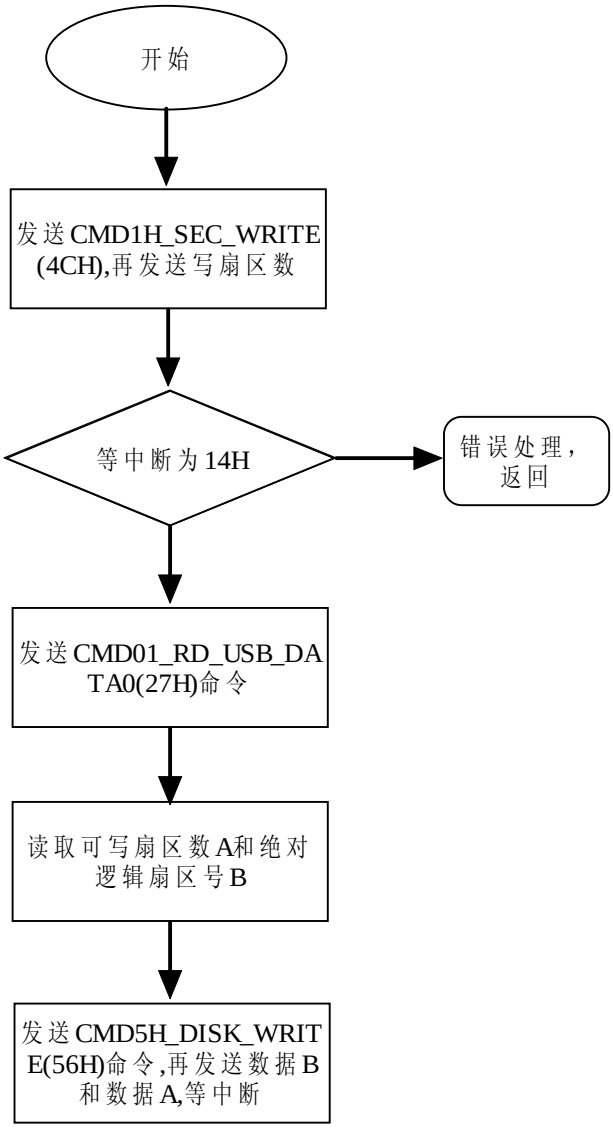
5、等待中断，如果中断状态为 USB_INT_DISK_WRITE(1EH)，则发送 CMD10_WR_HOST_DATA(2CH)。接着写入后续数据长度 40H 数据。再循环写入 40H 数据。否则返回；

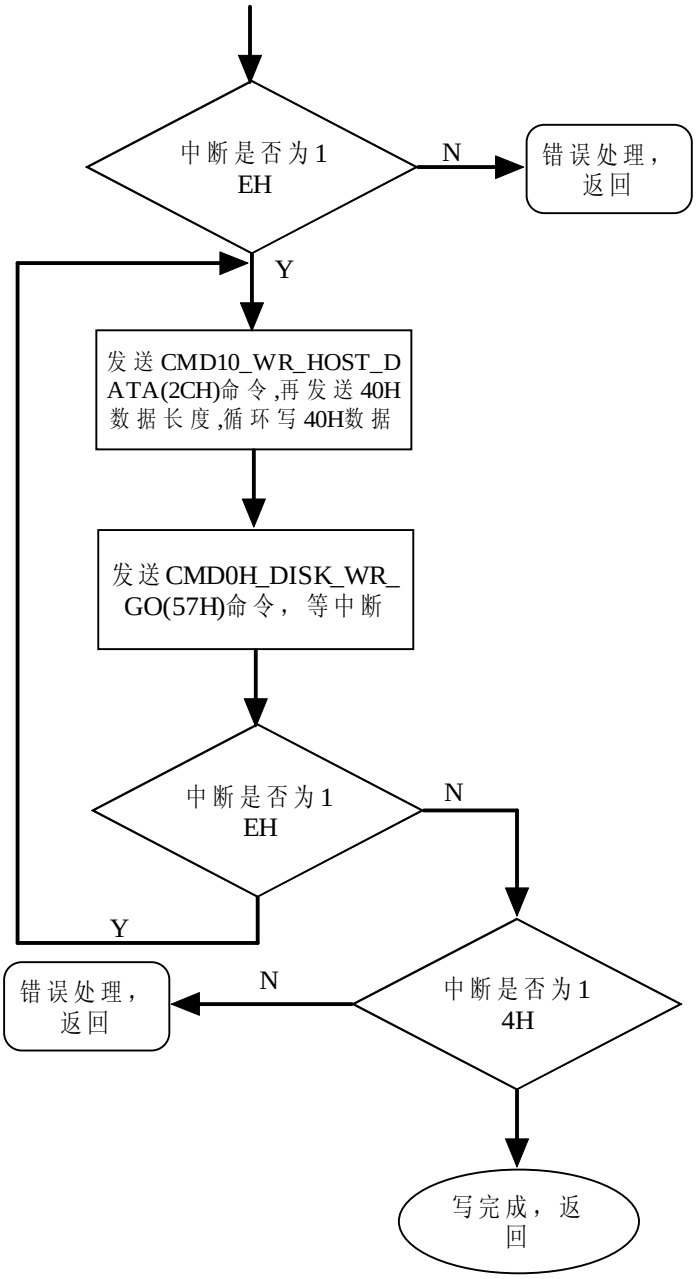
6、写完数据后，再发送 CMD0H_DISK_WR_GO(57H)数据，跳到步骤 5；

7、数据写完之后，等待中断。如果中断状态为 USB_INT_SUCCESS(14H)，则说明写成功，否则失败；

注：① 读取数据，从第零位数据开始读取，之后为第一位，然后为第二位，最后为第三位。

流程图如下：





3.10. 关闭文件

操作步骤如下：

1、发送命令 CMD1H_FILE_CLOSE(36H)，再发一个字节数据，此数据为 0 表示不更新文件长度，为 1 表示自动更新文件长度；

2、等待中断，如果中断状态为 USB_INT_SUCCESS(14H)表示关闭文件成功，否则失败；

3.11. 创建目录

3.11.1. 在根目录下创建目录

操作步骤：

1、使用 CMD_SET_FILE_NAME(2FH)命令设置要创建的目录名，目录名以数字 0 结束；

2、设置要创建目录的当前簇号.

发送写 CH376 内部 32 位变量命令 CMD50_WRITE_VAR32(0DH)；

发送要修改的变量名 VAR_CURRENT_CLUST(64H)；

发送 32 位的当前簇号，低字节在前，根目录下为 0；

3、发送创建目录的命令 DIR_CREATE(40H)；

4、等待 CH376 产生中断；

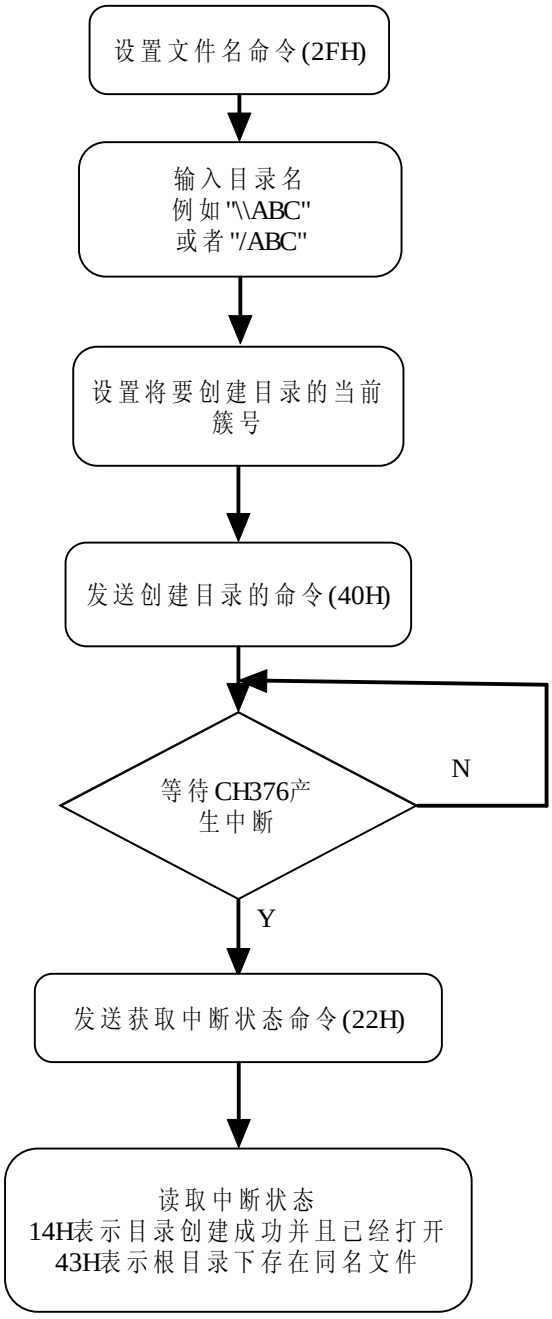
5、发送获取中断状态命令 GET_STATUS(22H)；

6、读取中断状态；

7、如果中断状态为 ERR_FOUND_NAME(43H) 说明该目录下存在同名文件；

如果中断状态为 USB_INT_SUCCESS(14H) 说明目录创建成功，并且已经打开；

流程图如下：



注：

1、目录名不可以超过 8 个字节，所有字符必须是 英文大写字母，数字，中文字符以及某些特殊字符；

2、相关例程请参考 CH376EVT \EXAM\EXAM9

3.11.2. 在多级目录下面创建子目录

操作步骤：

1、依次打开各级目录，直到最后一级目录(要创建目录的上级目录)；

2、读取上级目录的起始簇号(要创建目录的上级目录)；

发送读 CH376 内部 32 位文件系统统变量命令 CMD14_READ_VAR32 (0CH)；

发送要读取的变量名 VAR_START_CLUSTER (6OH)；

读取上级目录的起始簇号(32 位)，低字节在前；

3、在当前目录下新建目录，操作流程参考 3.11.1(上级目录已经打开)

注： “/ABC”或者\\ABC 表示在根目录下面

“ABC” 表示在当前目录下面

4、修改”创建目录”中保存的”上级目录”的起始簇号；

发送字节方式移动文件指针命令 BYTE_LOCATE(39H)；

发送指针的偏移量 52 个字节

以字节方式写入上级目录起始簇号的高 16 位，低字节在前；

发送字节方式移动文件指针命令 BYTE_LOCATE(39H)；

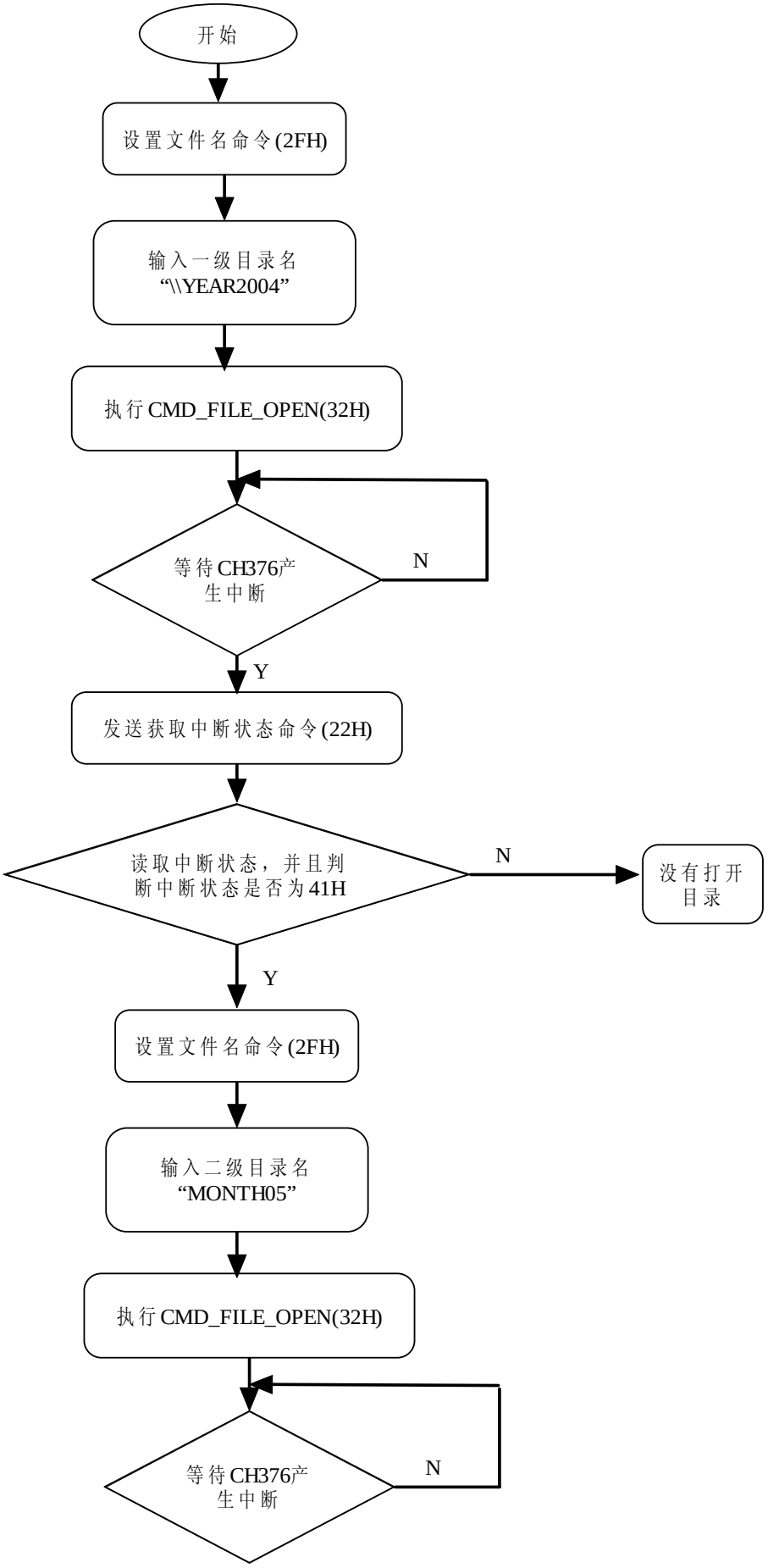
发送指针的偏移量 4 个字节

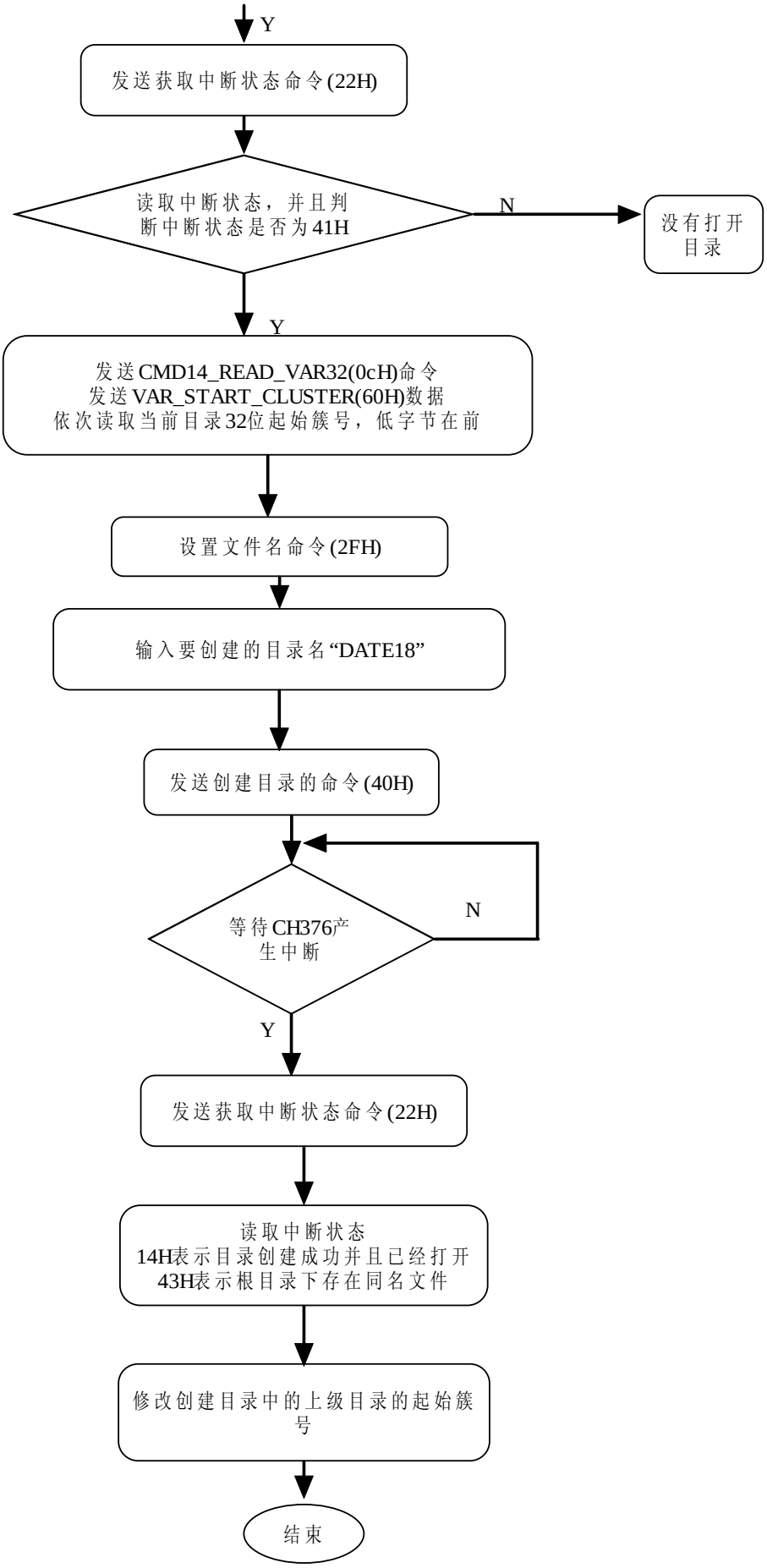
以字节方式写入上级目录起始簇号的低 16 位，低字节在前；

以字节方式移动文件指针命令 BYTE_LOCATE(39H)；

发送指针的偏移量 0 个字节(将文件指针移动到文件开头)

流程图如下： 演示在\\YEAR2004\\MONTH05\\下面创建 DATE18 目录


开始
设置文件名命令(2FH)
输入一级目录名
“\\YEAR2004”
执行CMD_FILE_OPEN(32H)
等待CH376产生中断
N
Y
发送获取中断状态命令(22H)
读取中断状态，并且判断中断状态是否为41H
N
没有打开目录
Y
设置文件名命令(2FH)
输入二级目录名
“MONTH05”
执行CMD_FILE_OPEN(32H)
等待CH376产生中断
N


Y
发送获取中断状态命令(22H)
读取中断状态，并且判断中断状态是否为41H
N
没有打开目录
Y
发送CMD14_READ_VAR32(0cH)命令
发送VAR_START_CLUSTER(60H)数据
依次读取当前目录32位起始簇号，低字节在前
设置文件名命令(2FH)
输入要创建的目录名“DATE18”
发送创建目录的命令(40H)
等待CH376产生中断
N
Y
发送获取中断状态命令(22H)
读取中断状态
14H表示目录创建成功并且已经打开
43H表示根目录下存在同名文件
修改创建目录中的上级目录的起始簇号
结束

3.12. 修改文件属性

3.12.1. 修改文件创建日期和时间

1、先打开要修改的文件，如果文件已经打开，则不需要重新打开

2、读取该文件的目录信息；

发送读取文件目录信息的命令 CMD1H_DIR_INFO_READ(37H)

发送该文件目录信息的索引号 0FFH 表示当前打开的文件；

3、等待 CH376 产生中断

4、发送获取中断状态命令 GET_STATUS(22H)；

5、读取中断状态， 14H 表示读取文件目录信息成功.

6、将时间转换成 16 位的数据

```
/* 文件时间 UINT16 */
/* Time = (Hour<<11) + (Minute<<5) + (Second>>1) */
/* 生成指定时分秒的文件时间数据 */
#define MAKE_FILE_TIME( h， m， s ) ( (h<<11) + (m<<5) + (s>>1) )
/* 文件日期 UINT16 */
/* Date = ((Year-1980)<<9) + (Month<<5) + Day */
/* 生成指定年月日的文件日期数据 */
#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、将创建日期写入 CH376 内部的缓冲区

Buf[0] = (unsigned char)iCreateDate;

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

发送 向 CH376 内部缓冲区 指定地址写入数据 的命令 CMD20_WR_OFS_DATA(2EH)；

发送偏移地址 16；

写入后续数据长度 2；

写入转换后的日期 Buf[0]；

写入转换后的日期 Buf[1]；

8、将创建时间写入 CH376 内部的缓冲区

Buf[0] = (unsigned char) iCreateTime；

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

发送 向 CH376 内部缓冲区 指定地址写入数据 的命令 CMD20_WR_OFS_DATA(2EH)；

发送偏移地址 14；

写入后续数据长度 2；

写入转换后的时间 Buf[0]；

写入转换后的时间 Buf[1]；

9、发送保存文件目录信息的命令 CMD0H_DIR_INFO_SAVE(38H)；

10、等待 CH376 产生中断

11、发送获取中断状态命令 GET_STATUS(22H)；

12、读取中断状态，14H 表示保存文件目录信息成功.

3.12.2. 修改文件的修改日期和时间

1、先打开要修改的文件，如果文件已经打开，则不需要重新打开

2、读取该文件的目录信息；

发送读取文件目录信息的命令 CMD1H_DIR_INFO_READ(37H)；

发送该文件目录信息的索引号 OFFH 表示当前打开的文件；

3、等待 CH376 产生中断

4、发送获取中断状态命令 GET_STATUS(22H)；

5、读取中断状态，14H 表示读取文件目录信息成功.

6、将时间转换成 16 位的数据

```
/* 文件时间 UINT16 */
/* Time = (Hour<<11) + (Minute<<5) + (Second>>1) */
/* 生成指定时分秒的文件时间数据 */
#define MAKE_FILE_TIME( h, m, s )  ( (h<<11) + (m<<5) + (s>>1) )
/* 文件日期 UINT16 */
/* Date = ((Year-1980)<<9) + (Month<<5) + Day */
/* 生成指定年月日的文件日期数据 */
#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、将修改日期写入 CH376 内部的缓冲区

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

发送 向 CH376 内部缓冲区 指定地址写入数据 的命令 CMD20_WR_OFS_DATA(2EH)；

发送偏移地址 24；

写入后续数据长度 2；

写入转换后的日期 Buf[0]；

写入转换后的日期 Buf[1]；

8、将修改时间写入 CH376 内部的缓冲区

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

发送 向 CH376 内部缓冲区 指定地址写入数据 的命令 CMD20_WR_OFS_DATA(2EH)；

发送偏移地址 22；

写入后续数据长度 2；

写入转换后的时间 Buf[0]；

写入转换后的时间 Buf[1]；

9、发送保存文件目录信息的命令 CMD0H_DIR_INFO_SAVE(38H)；

10、等待 CH376 产生中断

11、发送获取中断状态命令 GET_STATUS(22H)；

12、读取中断状态，14H 表示保存文件目录信息成功.

```
/* FAT 数据区中文件目录信息 */
typedef struct _FAT_DIR_INFO
{
    UINT8   DIR_Name[11]; /* 00H，文件名，共 11 字节，不足处填空格 */
    UINT8   DIR_Attr;   /* 0BH，文件属性，参考后面的说明 */
    UINT8   DIR_NTRes;  /* 0CH */
    UINT8   DIR_CrtTimeTenth; /* 0DH，文件创建的时间，以 0.1 秒单位计数 */
    UINT16  DIR_CrtTime;      /* 0EH，文件创建的时间 */
    UINT16  DIR_CrtDate;      /* 10H，文件创建的日期 */
```

```
    UINT16  DIR_LstAccDate;  /* 12H，最近一次存取操作的日期 */
    UINT16  DIR_FstClusHI;   /* 14H */
    UINT16  DIR_WrtTime;     /* 16H，文件修改时间，参考前面的宏 MAKE_FILE_TIME */
    UINT16  DIR_WrtDate;     /* 18H，文件修改日期，参考前面的宏 MAKE_FILE_DATE */
    UINT16  DIR_FstClusLO;   /* 1AH */
    UINT32  DIR_FileSize;    /* 1CH，文件长度 */
}
FAT_DIR_INFO,  *P_FAT_DIR_INFO;          /* 20H */
```

表 1

3.13. 删除文件

操作步骤：

1、先打开要删除的文件，对于已经打开的文件，则可以直接删除

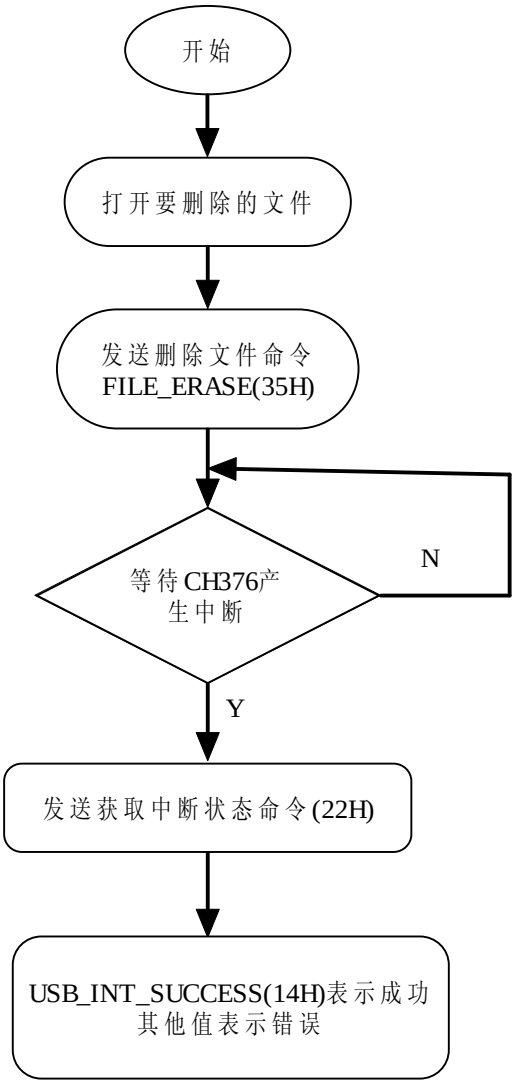
2、发送删除文件命令 FILE_ERASE(35H)；

3、等待 CH376 产生中断

4、发送获取中断状态命令 GET_STATUS(22H)；

5、读取中断状态，14H 表示删除文件成功.

操作流程：



注：删除目录和删除文件的方法一样，但是如果要删除目录，必须先把目录中的所有文件先删除，然后在删除该目录；

3.14. 以字节方式移动文件指针

操作步骤：

1、先打开目标文件，对于已经打开的文件，可以直接进行操作；

2、发送以字节方式移动文件指针的命令 BYTE_LOCATE(39H)；

3、写入 4 个字节的偏移量，低字节在前；

偏移量 0 表示文件开头；

偏移量 FFFFFFFFH 表示文件末尾；

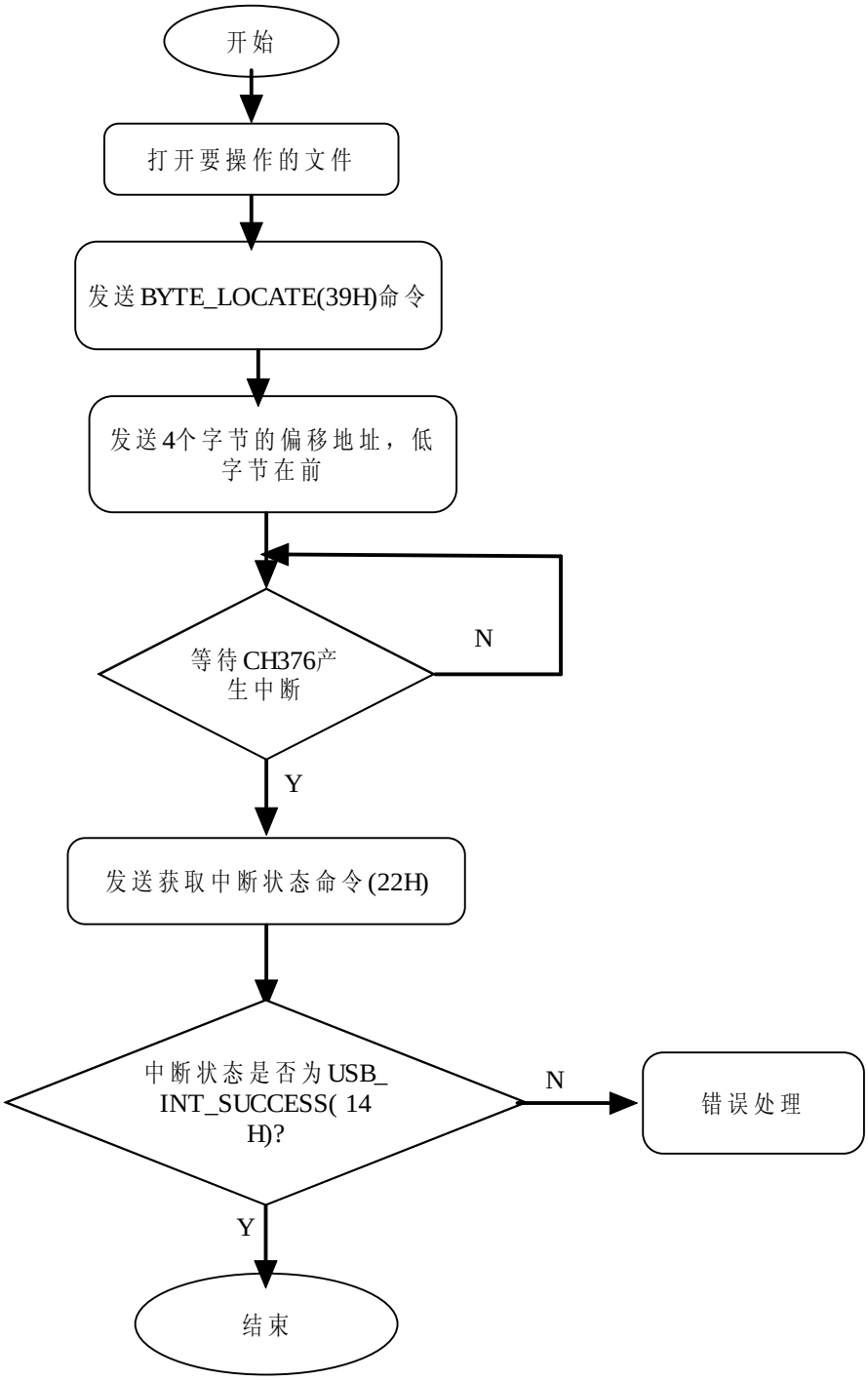
其他值 对应文件的具体位置

4、等待 CH376 产生中断

5、发送获取中断状态命令 GET_STATUS(22H)；

6、读取中断状态，14H 表示操作成功；

操作流程：



注：移动指针的偏移量不得超过文件长度；

以字节方式移动文件指针 不可以与以扇区方式移动文件指针一起使用；

.

3.15. 以扇区方式移动文件指针

操作步骤：

1、先打开目标文件，对于已经打开的文件，可以直接进行操作；

2、发送以扇区方式移动文件指针的命令 SEC_LOCATE（4AH）；

3、发送 4 个字节的扇区偏移地址，低字节在前；

扇区偏移量 0 表示将文件指针移动到文件开头；

扇区偏移量 FFFFFFFFH 表示将文件指针移动到文件末尾；

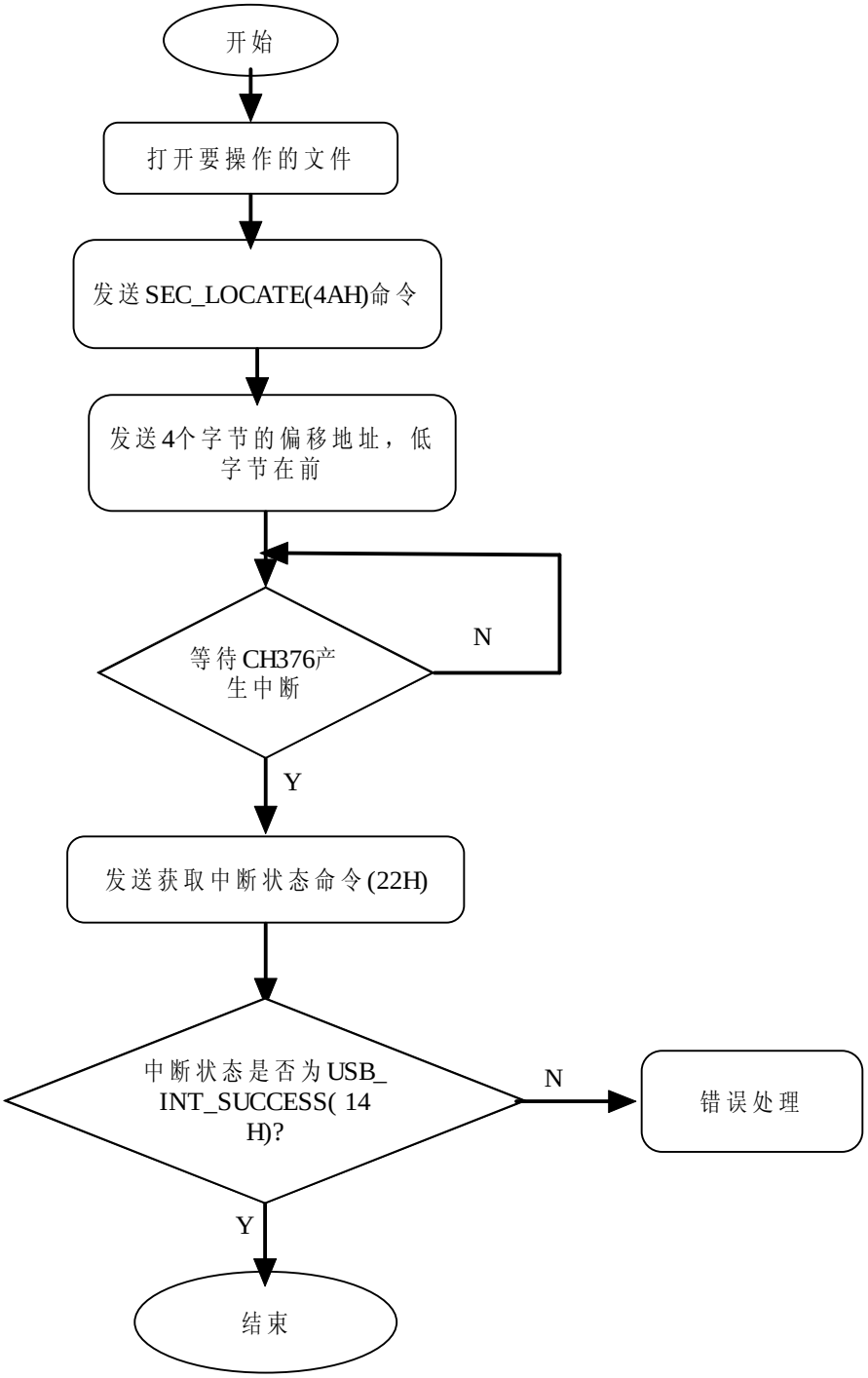
其它值 对应文件的具体位置；

4、等待 CH376 产生中断

5、发送获取中断状态命令 GET_STATUS(22H)；

6、读取中断状态，14H 表示操作成功；

操作流程：



注：以扇区方式移动文件指针，不支持 SD 卡的操作；

文件指针的偏移量不可以超过文件长度；

“以扇区方式移动文件指针”不可以与“以字节方式移动文件指针”一起使用

3.16. 枚举文件

操作步骤：

1、打开需要枚举文件的目录，打开目录的方法和打开文件的方法一样；

“\\” 表示根目录；

“\\ABC” 表示打开 ABC 目录；

2、使用 SET_FILE_NAME(2FH)命令设置 需要枚举和搜索的文件名；

发送 SET_FILE_NAME(2FH)命令设置文件名；

发送文件名，可以使用通配符”*”

“*”表示枚举当前目录下面的所有文件；

“USB*” 表示枚举当前目录下面的所有以 USB 开头的文件名；

通配符”*”后面不可以带有字符；

3、发送 CMD_FILE_OPEN(32H)命令开始枚举文件和目录；

4、CH376 比较每一个文件名，每当找到一个符合要求的文件，将对单片机产出一次中断；

5、等待 CH376 产生中断

6、发送获取中断状态命令 GET_STATUS(22H)；

7、读取中断状态，如果中断状态为 USB_INT_DISK_READ(1DH)表示枚举到匹配文件，并且请求单片机读取文件信息；如果没有枚举到匹配文件，转到步骤 13；

8、单片机发送 RD_USB_DATA0(27H)命令读取文件的目录信息；

发送 RD_USB_DATA0(27H)命令；

读取 CH376 返回后续的数据长度

读取 CH376 返回后续数据；该数据格式符合文件的目录信息结构参考表 1；

9、单片机发送继续枚举文件目录的命令 CMD_FILE_ENUM_GO(33H)；

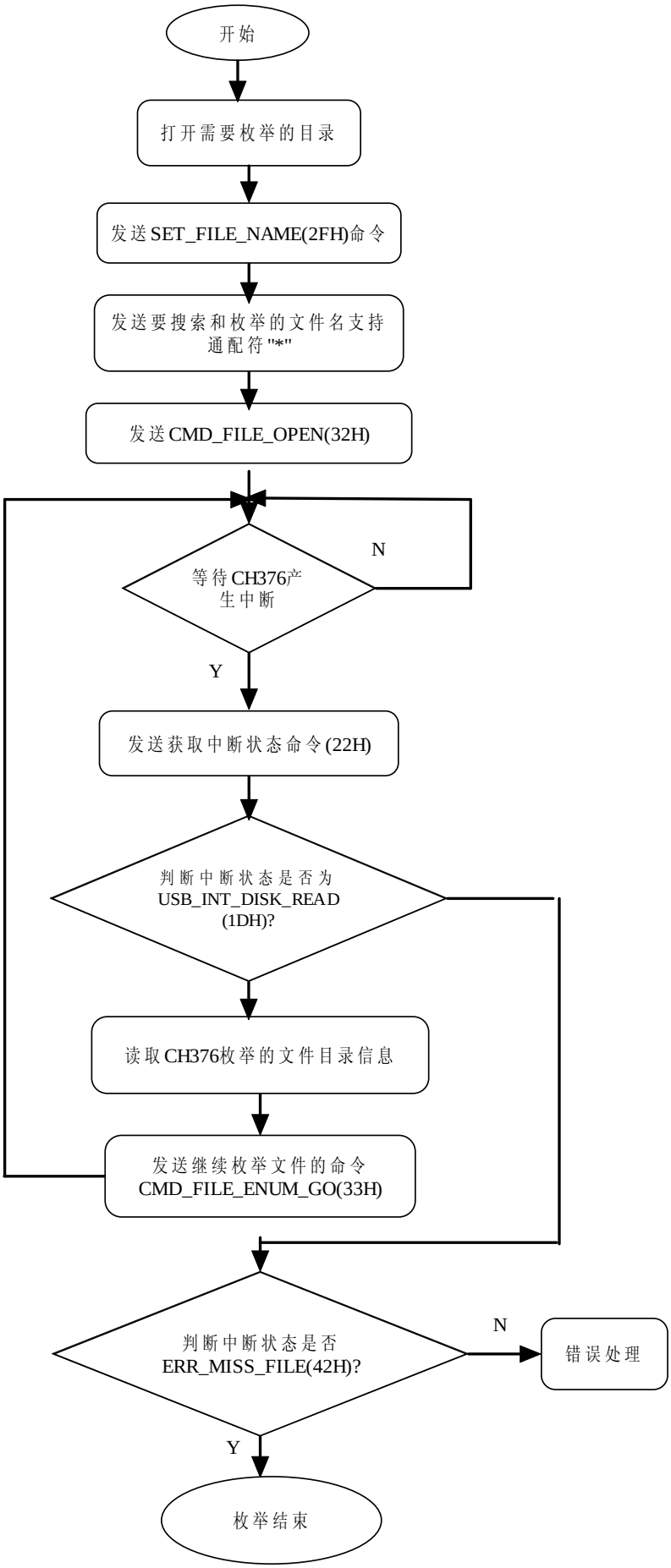
10、等待 CH376 产生中断

11、发送获取中断状态命令 GET_STATUS(22H)；

12、读取中断状态，如果中断状态为 USB_INT_DISK_READ(1DH)表示枚举到匹配文件，并且请求单片机读取文件信息；转到步骤 8；否则往下执行；

13、CH376 对单片机产出一次中断，中断状态为 ERR_MISS_FILE(42H)，说明没有找到更多的符合要求的文件，整个枚举操作结束。

操作流程：


开始
打开需要枚举的目录
发送SET_FILE_NAME(2FH)命令
发送要搜索和枚举的文件名支持通配符"*"
发送CMD_FILE_OPEN(32H)
等待CH376产生中断
N
Y
发送获取中断状态命令(22H)
判断中断状态是否为USB_INT_DISK_READ(1DH)?
读取CH376枚举的文件目录信息
发送继续枚举文件的命令CMD_FILE_ENUM_GO(33H)
判断中断状态是否ERR_MISS_FILE(42H)?
N
错误处理
Y
枚举结束

注：CH376EVT\EXAM\EXAM13 演示如何快速的枚举全盘文件；

3.17. 长文件名操作简述

1、如果要创建长文件名文件，必须先将长文件名转换成 UNICODE 编码(小端数据格式)；

2、为该长文件名分配一个短文件名(符合 8+3 格式的英文大写，数字，中文字符以及一些特殊字符)；

3、为该长文件名分配的短文件名 在同一级目录下必须是唯一对应关系；

4、如果对长文件名的文件进行读写操作，那么设置的文件名应该是该长文件名对应的短文件名，长文件名在文件操作的时候没有任何作用。

5、如果想要打开一个长文件名文件，则必须先要找到与其对应的短文件名，然后通过其短文件名打开该文件。删除长文件名文件同理；

6. 可以通过枚举的方法 找到与长文件名对应的短文件名；

7、可以通过短文件名获取该文件的长文件名；

由于长文件名的操作比较复杂，建议客户参考官方提供的实例程序， CH376EVT\EXAM\EXAM11 该例程演示如何创建一个长文件名文件，以及如何通过其短文件名获取对应的长文件名；