U 盘文件读写模块的使用说明

版本：2D

http://wch.cn

1、概述

很多单片机系统都需要存储器，当前，U 盘（含闪盘、USB 闪存盘、USB 移动硬盘等）已经成为很常用的移动存储设备，U 盘比闪存易于采购和携带，有多种容量可选。所以，单片机系统可以直接采用 U 盘作为大容量的移动存储器，并且方便与使用 WINDOWS 操作系统的计算机交换数据。

单片机可以通过 USB 总线 HOST&DEVICE 接口芯片 CH375 读写 U 盘中的数据，虽然直接调用 CH375 的 U 盘文件级子程序库读写 U 盘文件的效率更高，成本更低，但是该子程序库需要占用单片机系统的资源，大约 5KB 程序空间和 600 字节 RAM 数据存储器，无法应用于某些资源有限的单片机系统。

U 盘文件读写模块用于向嵌入式系统/单片机系统提供读写 U 盘中文件数据的接口，基本不需要占用单片机系统的存储空间，最少只需要几个字节的 RAM 和几百字节的代码。该模块基于 CH375 的 U 盘文件级子程序库设计，外围电路精简，性能价格比很高。

2、功能与特点

用于嵌入式系统/单片机读写 U 盘、闪盘、闪存盘、USB 移动硬盘、USB 读卡器等。
支持符合 USB 相关规范基于 Bulk-Only 传输协议的各种 U 盘/闪存盘/外置硬盘。
支持文件系统 FAT12 和 FAT16 及 FAT32，如果需要支持 FAT32 请看本文后面的说明。
提供工具程序，只要连接计算机 USB 端口，就可以随时升级模块，随时设置模块。
支持小端格式和大端格式的数据字节顺序，适用于绝大多数单片机系统。
文件操作功能：搜索、新建、删除、读写数据，查询和修改信息等。
读写模式：高速的扇区模式、方便的字节模式、简化的数据流模式。
提供 3 种硬件以适应不同的 I/O 接口：标准版、串口版、低电压版。
提供多种软件供随时下载到模块硬件中，通过多种软硬件组合支持各种不同的 I/O 接口。
模块具有简单的自动演示功能，提供串口连接方式下的计算机端的演示工具。

3、外部接口与升级

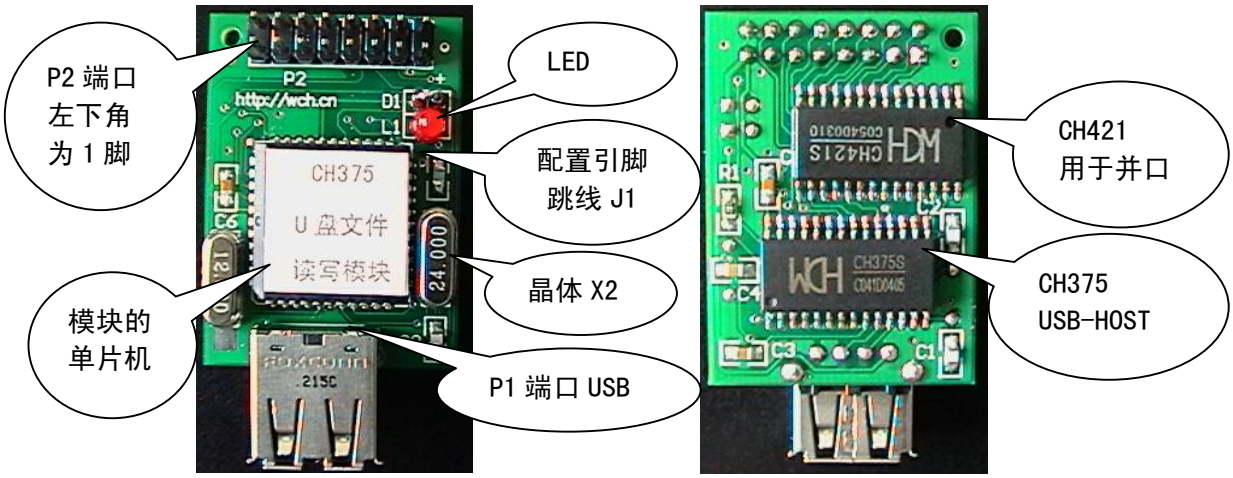
3.1 模块分类

模块的硬件与软件之间相对独立，模块硬件可以在用户端配置功能、下载和升级程序。通过模块软件与硬件之间的多种组合可以支持各种不同的 I/O 接口，具体组合见下表。

模块硬件分类 / I/O 接口 / 模块软件分类	标准版 16 脚双排针 5V-TTL，5V-CMOS	串口版 DB9 针（孔） RS232 电平	低电压版 10 脚双排针 3.3V-CMOS,支持 5V-TTL
标准版 目标程序文件 CH37XDL_	8 位被动并口或 3 线或 4+1 线异步串口	3 线异步串口	3 线异步串口
并口标准版 目标程序文件 PARALLEL	8 位被动并口		
串口标准版 目标程序文件 SERIAL	3 线异步串口或 4+1 线异步串口	3 线异步串口	3 线异步串口
SPI 标准版 目标程序文件 SPI			4 线 SPI 串口 4+1 线 SPI 串口
串口通用版 目标程序文件 SERIAL3	3 线异步串口	3 线异步串口	3 线异步串口

3.2 模块的外观

下面是标准版 U 盘文件读写模块的正反面外观。标准版模块的对外接口为 16 脚的双排针，模块尺寸约长 50mmX 宽 30mmX 高 10mm。



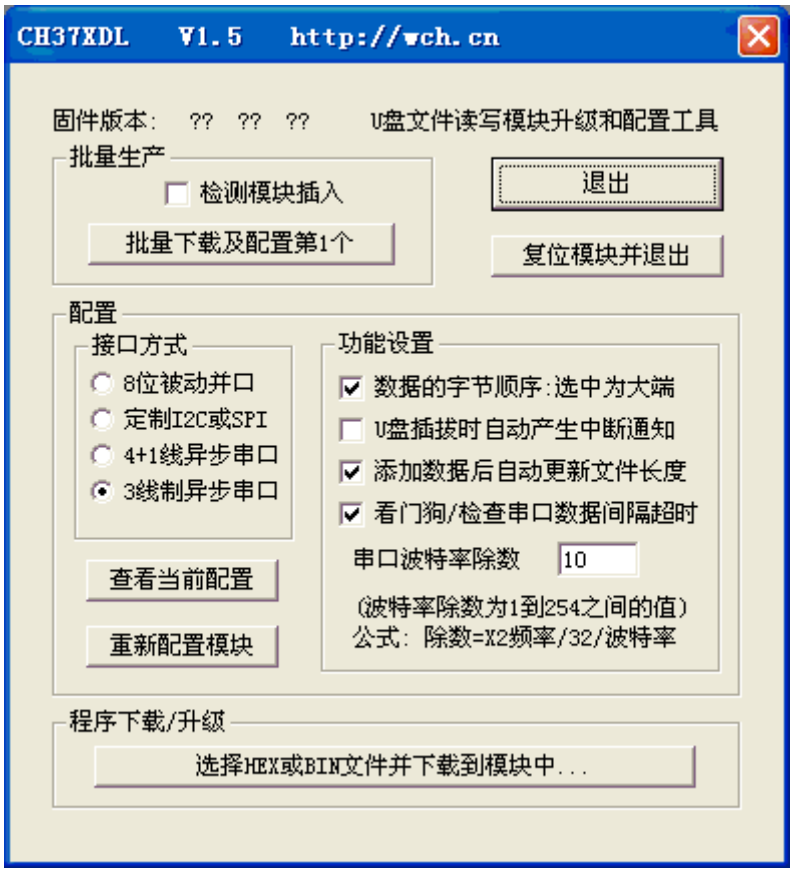
串口版的 U 盘文件读写模块实际上就是标准版的模块工作于三线制异步串口方式，再加上 TTL 电平与 RS232 电平的转换电路。串口版模块的对外接口为 RS232 电平的三线制串口，接口连接器为 DB9 针或孔。关于串口版模块的说明请参考 CH375HMX.PDF 文档。

低电压版的 U 盘文件读写模块实际上就是标准版的模块工作于 3.3V 电压的三线制异步串口方式，以及另外增加了 SPI 接口的 4 线串口方式。低电压版模块的对外接口为 10 脚的双排针。关于低电压版模块的说明请参考 CH375HML.PDF 文档。

3.3 升级和配置模块

在模块上电或者复位时，模块将检查跳线 J1 的状态。当跳线 J1 断开时进入正常工作状态，当跳线 J1 短路时进入功能配置状态。当升级模块程序或者配置完毕后，必须将跳线 J1 断开，重新上电才能进入正常工作状态。

在功能配置状态下，只要用 USB 对连线连接计算机的 USB 端口和模块的 USB 端口，就可以对模块进行在线程序升级和功能配置。首次连接计算机时，WINDOWS 提示找到 USB 新硬件，所以需要执行驱动程序包 CH372DRV.EXE 安装 U 盘模块的驱动程序（就是 CH372 或 CH375 的 WINDOWS 通用驱动程序），然后运行模块工具程序 CH37XDL.EXE，选择目标程序文件进行下载或者重新配置模块的接口。界面通常如右图。



在CH37XDL工具程序中，首先按[刷新]或者[查看当前配置]，显示模块的当前配置，其中固件程序版本是指模块内置的USB下载固件程序的版本，与U盘模块主程序的版本无关。如果需要升级模块的主程序，可以点[选择文件并下载]或者[选择HEX或BIN文件并下载到模块中]按钮，指定厂家提供的新版模块程序BIN文件或者自行定制功能的模块程序HEX文件，将其下载到U盘模块中。如果需要修改模块接口，请在选择好后点[重新配置]或者[重新配置模块]按钮。

在界面中可以选择模块的外部接口为“并口”、“串口”等（参考第3.1节模块分类），并且还可以设置一些附加功能，有些相当于命令CMD_SetupModule的功能。功能设置项有：

- 数据的字节顺序是指模块对外接口数据的格式，选中为大端格式BIG_ENDIAN，未选中为小端格式LITTLE_ENDIAN。通常情况下，如果是使用C语言的MCS51单片机，那么用大端格式；如果是80X86兼容单片机或PC机、ARM、MSP430、AVR单片机等，那么用小端格式。
- U盘插拔后自动产生中断通知是指事件自动通知，在USB主机模式下，是U盘或其它USB设备插拔的事件自动通知；在USB设备模式下，是下传数据成功和上传数据成功的事件自动通知。未选中时禁止该功能，可以用查询状态命令代替。选中后启用该功能，在USB主机模式下，如果模块检测到U盘连接或者断开事件，那么自动通知单片机系统；在USB设备模式下，当模块接收到上位机PC机的下传数据后或者当上位机PC机取走模块的上传数据后，模块自动通知单片机系统。
- 添加数据后自动更新文件长度是指在用CMD_FileWrite或者CMD_ByteWrite命令向文件添加数据后，模块是否自动更新文件长度。如果未选中，则模块不会自动更新文件长度，但是单片机系统可以在需要时用0长度写命令或者其它命令通知模块更新或者修改文件长度。
- 看门狗/检查串口两数据间隔超时是指在单片机系统通过串口向模块输入命令包时，如果连续两个数据字节之间的间隔大于串口输入超时时间，模块是否放弃该命令包。选中后将检查超时，避免处理不完整的命令包，建议选中。如果用PC机串口调试工具测试模块功能，可以临时关闭串口超时检查，以便手工输入命令包。对于各种标准版程序，选中该选项将启用模块内部的看门狗。

如果选择串口，那么还应该指定波特率除数，相当于命令CMD_BaudRate的功能，计算公式是晶体X2的频率除以32再除以波特率。默认情况下，X2的频率为18.432MHz，如果要求模块的串口波特率为115200bps，那么应该指定波特率除数为5（即18432000/32/115200）。

[批量下载及配置]用于批量处理，每点击一次处理一个模块，处理过程中该按钮临时禁止，直到处理完成。每点击一次自动包括两个步骤：先下载程序再配置模块，并计数。该按钮用于为所有模块下载同一程序，并配置为同一接口和同一功能，事先必须在当前界面中设置为所需要的功能设置。如果选中[检测模块插入]，那么每当检测到下一个模块插入后将自动执行[批量下载及配置]，从而可以实现自动批量生产，完全不需要操作键盘和点击鼠标。

3.4 外部接口与连接

标准版模块具有两个外部接口：P1是USB插座，可以直接插入U盘或者通过USB延长线连接U盘，当进行程序升级或者重新配置时应该通过USB对连线连接计算机的USB端口；P2是16脚的双排针或者插座，用于连接单片机系统。

关于并口连接说明请参考CH375HMP.PDF文档。

关于串口连接说明请参考CH375HMS.PDF文档。

模块具有一个空闲状态指示灯LED，在空闲时LED亮，当收到命令正忙时LED灭。如果是在功能配置状态中，那么LED亮说明计算机正在与U盘模块通讯。

4、接口操作

基本操作步骤是，单片机系统将命令码、后续参数长度（因为各命令码所需要的参数不等长）和参数写给模块，并通知其启动操作，模块执行完成后以中断方式通知单片机，并返回操作状态和操作结果。注意，数据流模式的命令执行完成后不返回状态。

因为接口操作看起来比较复杂，所以实际过程可以参考随模块一起提供的几个示例程序，直接用其中的ExecCommand子程序就可以了，不必详细理解接口步骤说明。

关于并口连接说明请参考 CH375HMP.PDF 文档。
关于串口连接说明请参考 CH375HMS.PDF 文档。

5、模块命令

5.1 文件读写模式

模块对 U 盘文件的读写方式分为三种：扇区模式和字节模式，以及数据流模式。

扇区模式下，以扇区（每扇区通常是 512 字节）为基本单位对 U 盘文件进行读写，所以读写速度较快，但是外部单片机系统应该至少具有 512 字节的 RAM 内存，而且 RAM 越大读写效率通常更高，适用于 RAM 多、数据量大、频繁读写数据的系统应用。

字节模式下，以字节为基本单位对 U 盘文件进行读写，只要外部单片机系统的 RAM 不少于几十个字节（最少为十几个字节，RAM 越多效率越高），就可以通过模块读写 U 盘中的文件，读写速度较慢，适用于 RAM 少、数据量小、不经常读写数据的系统应用。

数据流模式是字节模式的高级应用，编程更简单，用于处理时间跨度较大较分散的字符数据流。数据流模式下，以单字节即单个字符为单位对 U 盘文件进行读写，外部单片机系统不需要 RAM，就可以通过模块读写 U 盘中的文件，读写速度最慢，但是编程最简单，适用于没有 RAM、数据量小、纯文本流式数据的系统应用。数据流模式不支持并口，只支持串口，外部单片机系统可以是单片机、PLC 可编程控制器、PC 机等协议较简单的控制设备。

查看磁盘及文件状态 mDiskStatus 可以获取当前的文件模式：为 DISK_OPEN_FILE 则代表扇区模式，为 DISK_OPEN_FILE_B 则代表字节模式或者数据流模式。

每次新建或者打开一个文件后，默认为扇区模式，支持以扇区为单位的文件操作命令 CMD_FileRead 和 CMD_FileWrite 及 CMD_FileLocate。当执行一次以字节为单位的操作命令后将自动进入字节模式（只有关闭文件后再重新打开才能恢复到扇区模式），支持以字节为单位的操作命令 CMD_ByteRead 和 CMD_ByteWrite 以及 CMD_ByteLocate。对于已打开的同一个文件两种模式的操作命令不能混用。数据流模式与字节模式可以混用。

5.2 文件长度

在 FAT 文件系统中，磁盘容量以簇为基本单位进行分配，而簇的大小总是扇区的倍数，所以文件的占用空间总是簇的倍数，也是扇区的倍数。虽然文件占用的空间是扇区的倍数，是 512 的倍数，但是在实际应用中，保存在文件中的有效数据的长度却不一定是 512 的倍数，所以 FAT 文件系统在 FDT 文件目录信息表中专门记录了当前文件中有效数据的长度，即通常所说的文件长度，文件长度总是小于或者等于文件占用的空间。

在对文件写入数据后，如果是覆盖了原数据，那么文件长度可能不发生变化，当超过原文件长度后，变为追加数据，那么文件长度应该发生变化（增大）。如果向文件追加数据后，没有修改 FDT 中的文件长度，那么 FAT 文件系统会认为超过文件长度的数据是无效的，正常情况下，计算机无法读出超过文件长度的数据，虽然数据实际存在。

写数据命令 CMD_FileWrite 和 CMD_ByteWrite 用于将数据写到文件所占用的磁盘空间中，如果是追加数据并且超过原文件长度后，这些命令记录新的文件长度。

如果数据量少或者数据不连续，那么可以在每次追加数据后立即更新 FDT 中的文件长度，但是，如果数据量大并且需要连续写入数据，立即更新 FDT 会降低效率，并且频繁修改 FDT 也会缩短 U 盘中闪存的使用寿命（因为闪存只能进行有限次擦写），所以在这种情况下，应该在连续写入多组数据后再更新一次 FDT 中的文件长度，或者一直等到关闭文件时再更新文件长度。通过 CMD_SetupMdule 命令和 USB 功能配置可以修改模块中的全局变量 CH375LibConfig，从而可以设定在添加数据后是否立即更新 FDT 中的文件长度，默认情况下是不更新文件长度。

在字节模式下，模块能够自动处理任意的文件长度，在 CMD_FileClose 关闭文件时，总是自动更新 FDT 中的文件长度，确保文件长度真实反应文件中有效数据的长度。在字节模式下修改文件长度有 3 种方法：

方法 1：执行 CMD_ByteWrite，写入 0 字节数据，强制更新文件长度

方法 2：执行 CMD_FileClose，关闭文件，总是自动更新文件长度

方法 3：执行 CMD_FileModify，指定修改文件长度，任意的文件长度

在扇区模式下，模块只能自动处理 512 倍数的文件长度，在关闭文件时，可以选择自动更新文件长度或者不更新，如果文件中有效数据的长度是 512 的倍数，那么可以指定自动更新，如果有效数据的长度不是 512 的倍数，那么可以由 CMD_FileModify 修改文件长度。在扇区模式下修改文件长度有 3 种方法：

方法 1：执行 CMD_FileWrite，写入 0 扇区数据，强制更新文件长度，文件长度总是 512 倍数

方法 2：执行 CMD_FileClose，关闭文件，指定自动更新文件长度，文件长度总是 512 倍数

方法 3：执行 CMD_FileModify，指定修改文件长度，任意的文件长度

在数据流模式下，模块能够自动处理任意的文件长度，在退出数据流读写命令时，能够自动更新 FDT 中的文件长度，如果需要，也可以参照字节模式修改文件长度。

5.3 操作命令总表

绝大多数的模块命令与 CH375 的 U 盘文件级子程序库相对应（可以参考 CH375 评估板资料中的 CH375HF.PDF 文档）。如果命令执行失败，那么只返回状态码，不返回任何结果数据。如果命令执行成功，才有可能返回结果数据，而且有些命令总是不返回任何结果数据。命令所需要的输入参数和返回的结果数据都通过同一个联合结构 CMD_PARAM 传递。

命令分类	命令简称	代码	命令用途和概述（详细说明见后）
常用的基本命令	CMD_GetVer	0x0A	获取当前模块的版本号,CH375 子程序库的版本号等
	CMD_QueryStatus	0x60	查询当前模块状态：U 盘是否连接,当前文件长度等
	CMD_DiskReady	0x71	查询 U 盘是否准备就绪,通常在就绪后才能读写
	CMD_FileOpen	0x64	打开指定名称的文件或者目录
	CMD_FileEnumer	0x63	搜索枚举指定目录下的文件,返回文件名
	CMD_FileCreate	0x65	新建文件并打开,如果文件已存在则先删除再新建
	CMD_FileClose	0x67	关闭当前文件
扇区模式读写文件	CMD_FileLocate	0x6A	以扇区为单位移动当前文件指针
	CMD_FileRead	0x6B	以扇区为单位从当前文件读取数据
	CMD_FileWrite	0x6C	以扇区为单位向当前文件写入数据
	CMD_FileReadLast	0x77	从当前文件的尾部读取不足一扇区长度的零碎数据
字节模式读写文件	CMD_ByteLocate	0x7A	以字节为单位移动当前文件指针，进入字节模式
	CMD_ByteRead	0x7B	以字节为单位从当前文件读取数据，进入字节模式
	CMD_ByteWrite	0x7C	以字节为单位向当前文件写入数据，进入字节模式
数据流模式读写文件	CMD_StreamRead	0x7E	数据流模式读文件,只支持串口,只支持文本
	CMD_StreamWrite	0x7F	数据流模式写文件,只支持串口,只支持文本
查询修改文件信息	CMD_FileQuery	0x68	查询当前文件的信息：长度,日期,时间等
	CMD_FileModify	0x69	查询或修改当前文件的信息：长度,日期,时间等
	CMD_FileDirInfo	0x75	查询或修改当前已打开文件的目录信息 FDT
	CMD_SetFileSize	0x78	修改模块系统中子程序库的文件长度变量
偶尔用到	CMD_DiskSize	0x72	查询磁盘总容量：U 盘闪盘移动硬盘的物理容量
	CMD_DiskQuery	0x61	查询磁盘信息：U 盘总容量，U 盘剩余容量等
	CMD_FileErase	0x66	删除文件并关闭
	CMD_DirCreate	0x76	新建目录并打开,如果目录已经存在则直接打开
备用命令	CMD_BulkOnlyCmd	0x70	执行基于 BulkOnly 协议的命令
模块硬件相关命令	CMD_SetupModule	0xA6	临时设置模块配置：事件通知,大小端格式等
	CMD_BaudRate	0xA5	临时设置串口通讯波特率
	CMD_ResetInit	0x0B	复位并重新初始化 CH375 以及模块,命令不返回
	CMD_GetStringSN	0xA0	获取产品序列号字符串,保留命令

连接计算机时 USB 设备模式命令	CMD_SetUsbId	0x12	USB 设备：设置 USB 设备的厂商 VID 和产品 PID
	CMD_SetUsbMode	0x15	设置 USB-HOST 主机或设备模式,只支持串口
	CMD_ReadUsbData	0x28	USB 设备：从模块的数据下传端点读取数据块
	CMD_WriteUsbData	0x2B	USB 设备：向模块的数据上传端点写入数据块
硬件底层备用命令	CMD_DirectWrCmd	0xB9	直接传递给 CH375 芯片,写命令
	CMD_DirectRdDat	0xB5	直接传递给 CH375 芯片,读数据
	CMD_DirectWrDat	0xB6	直接传递给 CH375 芯片,写数据

以下是主要命令的输入参数与返回结果
更详细说明参照 C 语言 CH375HM.H 或 ASM 语言 CH375HM.INC 头文件中 CMD_PARAM 结构的定义

```
struct {
    unsigned char   mFileLibVer;    /* 返回: 子程序库的版本号 */
    unsigned char   mModuleVer;     /* 返回: 模块的版本号 */
    unsigned char   mUsbIcVer;      /* 返回: USB 芯片版本:10H-CH375S,2xH-CH375A */
} GetVer;                           /* CMD_GetVer, 获取当前模块的版本号 */

struct {
    unsigned char   mLastStatus;    /* 返回: 上次的操作状态 */
    unsigned char   mDiskStatus;    /* 返回: 磁盘及文件状态 */
    unsigned char   mIntStatus;     /* 返回: CH375 操作的中断状态 */
    unsigned char   reserved;
    unsigned long   mFileSize;      /* 返回: 当前文件的长度 */
    unsigned long   mCurrentOffset; /* 返回: 当前文件指针,当前读写位置的字节偏移 */
} Status;                           /* CMD_QueryStatus, 查询当前模块的状态 */

struct {
    unsigned char   mPathName[ MAX_PATH_LEN ];  /* 输入参数: 路径/目录名/文件名 */
} Open;                             /* CMD_FileOpen, 打开文件 */

struct {
    unsigned char   mPathName[ MAX_PATH_LEN ];  /* 输入参数: 路径/目录名/文件名 */
} Create;                           /* CMD_FileCreate, 新建文件并打开 */

struct {
    unsigned char   mUpdateLen;     /* 输入参数: 是否允许更新长度: 0 禁止,1 允许 */
} Close;                            /* CMD_FileClose, 关闭当前文件 */

struct {
    unsigned long   mSectorOffset;  /* 输入参数: 扇区偏移,0 则移动到文件头,0FFFFFFFFH 则
移动到文件尾, 返回: 当前文件指针对应的绝对线性扇区号, 0FFFFFFFFH 则已到文件尾 */
} Locate;                           /* CMD_FileLocate, 移动当前文件指针 */

struct {
    unsigned char   mSectorCount;   /* 输入参数: 读取扇区数, 返回: 实际读取扇区数 */
} Read;                             /* CMD_FileRead, 从当前文件读取数据 */

struct {
    unsigned char   mSectorCount;   /* 输入参数: 写入扇区数, 返回: 实际写入扇区数 */
```

```
} Write;                                /* CMD_FileWrite, 向当前文件写入数据 */

struct {
    unsigned char   mSectorCount;    /* 返回: 实际读取扇区数,为 1 则已经读取最后扇区,为 0 则没有零碎数据(文件长度是 512 的倍数) */
} ReadLast;                             /* CMD_FileReadLast, 从当前文件的尾部读

struct {
    unsigned long   mByteOffset;     /* 输入参数: 以字节为单位的偏移量, 以字节为单位的文件指针, 返回: 当前文件指针对应的绝对线性扇区号, OFFFFFFFFH 则已到文件尾 */
} ByteLocate;                           /* CMD_ByteLocate, 以字节为单位移动当前文件指针 */

struct {
    unsigned char   mByteCount;      /* 输入参数: 准备读取的字节数,不得大于 MAX_BYTE_IO, 返回: 实际读出的字节数 */
    unsigned char   mByteBuffer[ MAX_BYTE_IO ]; /* 返回: 读出的数据块 */
} ByteRead;                             /* CMD_ByteRead, 以字节为单位从文件读取数据块 */

struct {
    unsigned char   mByteCount;      /* 输入参数: 准备写入的字节数,不得大于 MAX_BYTE_IO, 如果为 0 则刷新文件长度而不写入,否则写入数据但不刷新文件长度, 返回: 实际写入的字节数 */
    unsigned char   mByteBuffer[ MAX_BYTE_IO ]; /* 输入参数: 准备写入的数据块 */
} ByteWrite;                            /* CMD_ByteWrite, 以字节为单位向文件写入数据块 */

struct {
    unsigned long   mFileSize;  /* 输入:新文件长度,为 OFFFFFFFFH 则不修改,返回:原长度 */
    unsigned short  mFileDate;  /* 输入参数:新文件日期,为 OFFFFH 则不修改,返回:原日期 */
    unsigned short  mFileTime;  /* 输入参数:新文件时间,为 OFFFFH 则不修改,返回:原时间 */
    unsigned char   mFileAttr;  /* 输入参数:新文件属性,为 OFFH 则不修改,返回:原属性 */
} Modify;   /* CMD_FileQuery, 查询文件的信息; CMD_FileModify, 查询或修改文件的信息 */

struct {
    unsigned long   mDiskSizeSec;    /* 返回: 整个物理磁盘的总扇区数 */
    unsigned long   mTotalSector;    /* 返回: 当前逻辑盘的总扇区数 */
    unsigned long   mFreeSector;     /* 返回: 当前逻辑盘的剩余扇区数 */
    unsigned char   mDiskFat;        /* 返回: 当前逻辑盘的 FAT 类型 */
} Query;                                /* CMD_DiskQuery, 查询磁盘信息 */
```

5.4 常用操作步骤

以下是常用步骤简述，可以根据实际情况进行调整

5.4.1 初始化（除数据流模式的命令外，进行任何一种文件操作之前的必要步骤）

⑴ 初始化单片机与模块之间的接口，设置必要的参数等

⑵ 等待 U 盘连接（中断方式依靠 CH375 事件通知，查询方式依靠主动查询）

⑶ CMD_DiskReady，可选步骤，强烈建议执行此步骤

⑷ CMD_DiskSize，可选步骤，通常不需要

5.4.2 顺序读文件

⑴ CMD_FileOpen，打开文件

(2) 多次 CMD_FileRead 或 CMD_ByteRead，读取数据
(3) 如果是在字节模式下请跳过此步骤；如果是在扇区模式下，并且需要读取文件尾部的不足一个扇区的零碎数据，那么通过 CMD_FileReadLast 命令，下面的步骤与此类似
(4) CMD_FileClose，关闭文件

5.4.3 读文件的指定位置
(1) CMD_FileOpen，打开文件
(2) CMD_FileLocate 或 CMD_ByteLocate，移动文件指针到指定位置
(3) 多次 CMD_FileRead 或 CMD_ByteRead，读取数据，期间还可以移动文件指针
(4) CMD_FileClose，关闭文件

5.4.4 顺序改写文件（覆盖原数据，超过原文件长度后转变为追加数据）
(1) CMD_FileOpen，打开文件
(2) 多次 CMD_FileWrite 或 CMD_ByteWrite，写入数据
(3) 如果是追加数据导致文件长度增大，那么需要参考修改文件长度的几种方法
(4) CMD_FileClose，关闭文件，如果是字节模式，将自动更新文件长度

5.4.5 改写文件的指定位置（覆盖原数据，超过原文件长度后转变为追加数据）
(1) CMD_FileOpen，打开文件
(2) CMD_FileLocate 或 CMD_ByteLocate，移动文件指针到指定位置
(3) 多次 CMD_FileWrite 或 CMD_ByteWrite，写入数据，期间还可以移动文件指针
(4) 如果是追加数据导致文件长度增大，那么需要参考修改文件长度的几种方法
(5) CMD_FileClose，关闭文件，如果是字节模式，将自动更新文件长度

5.4.6 向已有文件追加数据
(1) CMD_FileOpen，打开文件
(2) CMD_FileLocate 或 CMD_ByteLocate，移动文件指针到末尾，0xFFFFFFFF
(3) 多次 CMD_FileWrite 或 CMD_ByteWrite，写入数据
(4) 如果是追加数据导致文件长度增大，那么需要参考修改文件长度的几种方法
(5) CMD_FileClose，关闭文件，如果是字节模式，将自动更新文件长度

5.4.7 新建文件
(1) CMD_FileCreate，新建文件，可以用 CMD_FileDirInfo 设置更多附加信息
(2) 多次 CMD_FileWrite 或 CMD_ByteWrite，写入数据
(3) 如果是追加数据导致文件长度增大，那么需要参考修改文件长度的几种方法
(4) CMD_FileClose，关闭文件，如果是字节模式，将自动更新文件长度

5.4.8 先读文件再改写文件
(1) CMD_FileOpen，打开文件
(2) 多次 CMD_FileRead 或 CMD_ByteRead，读取数据
(3) CMD_FileLocate 或 CMD_ByteLocate，移动文件指针到头部，00000000
(4) 多次 CMD_FileWrite 或 CMD_ByteWrite，写入数据，覆盖原数据
(5) 如果是追加数据导致文件长度增大，那么需要参考修改文件长度的几种方法
(6) CMD_FileClose，关闭文件，如果是字节模式，将自动更新文件长度

5.4.9 先写文件再读文件复查
(1) CMD_FileOpen，打开文件
(2) 多次 CMD_FileWrite 或 CMD_ByteWrite，写入数据
(3) CMD_FileLocate 或 CMD_ByteLocate，移动文件指针到头部，00000000
(4) 多次 CMD_FileRead 或 CMD_ByteRead，读取数据，复查

(5) 如果是追加数据导致文件长度增大，那么需要参考修改文件长度的几种方法
(6) CMD_FileClose，关闭文件，如果是字节模式，将自动更新文件长度

5.4.10 如果文件已经存在则追加数据，如果文件不存在则新建文件再写入数据
(1) CMD_FileOpen，打开文件，如果返回 ERR_MISS_FILE 文件不存在，则转到步骤(4)
(2) CMD_FileLocate 或 CMD_ByteLocate，移动文件指针到末尾，0xFFFFFFFF
(3) 转到步骤(5)，准备追加数据
(4) CMD_FileCreate，新建文件，准备写入数据
(5) 多次 CMD_FileWrite 或 CMD_ByteWrite，写入数据
(6) 如果是追加数据导致文件长度增大，那么需要参考修改文件长度的几种方法
(7) CMD_FileClose，关闭文件，如果是字节模式，将自动更新文件长度

5.4.11 定期采集数据（适用于数据量较小的情况）
(1) 采集之前，CMD_FileCreate，新建文件
(2) 采集数据，转换为相应的格式，例如二进制数据、字符串等
(3) CMD_ByteWrite，写入数据，一次写不完，可以分多次写入
(4) 如果要等很长时间才有下一组数据，为了避免在此期间发生断电、U 盘拔出等事件，导致文件长度不正确，可以用 CMD_ByteWrite 写入空数据，强制更新文件长度
(5) 如果整个采集过程结束，或者文件已经太大，那么转到步骤(6)，否则转到步骤(2)
(6) CMD_FileClose，关闭文件，自动更新文件长度
(7) 如果是因为文件已经太大的原因，那么转到步骤(1)，新建另一个文件继续

5.4.12 枚举/搜索文件并读写
用 CMD_FileEnumer 命令，指定通配符和指定序号枚举文件名，如果返回 ERR_MISS_FILE 那么说明再也没有匹配的文件名，完成枚举；如果返回 ERR_SUCCESS 那么说明搜索到指定序号的文件名，直接用 CMD_FileOpen 打开后再读写，完成后关闭。增加序号后可以继续枚举，直到搜索到指定的文件名或者枚举完毕。枚举完成后，可以用 CMD_QueryStatus 命令查询当前文件长度，如果是 0xFFFFFFFF 那么说明枚举到的是子目录（可以考虑进入该子目录搜索），否则说明是普通文件。

5.4.13 以数据流模式从默认文件读取数据
(1) 上电或复位后，初始化单片机与模块之间的串行接口，设置波特率等参数
(2) CMD_StreamRead，准备读取文件。模块会自动等待 U 盘插入，然后自动打开默认文件。例如在三线制串口连接时，该命令及同步码共 4 个字节“57H、ABH、7EH、00H”
(3) 模块从文件中读取一个字符并从串口输出，文件指针自动向后移动一个字节
(4) 单片机系统接收到该字符后，从串口发出应答，或者在需要时发出控制码
(5) 模块收到应答后，转到步骤(3)继续读取下一个字符并输出，直到文件结束

5.4.14 以数据流模式向默认文件写入数据
(1) 上电或复位后，初始化单片机与模块之间的串行接口，设置波特率等参数
(2) CMD_StreamWrite，准备写入文件。模块会自动等待 U 盘插入，然后自动打开默认文件并移动文件指针到文件末尾以便追加数据，如果原文件不存在则自动创建新文件。例如在三线制串口连接时，该命令及同步码共 4 个字节“57H、ABH、7FH、00H”
(3) 单片机系统将准备写入的字符从串口输出，或者在需要时发出控制码
(4) 模块收到该字符后，先添加到缓冲区中，并从串口发出应答。如果缓冲区已满、或者收到刷新控制码、或者串口收发连续空闲 10 秒以上，模块都将缓冲区中的数据写入 U 盘文件中并清空缓冲区和刷新文件长度，文件指针自动向后移动
(5) 单片机系统收到应答后，转到步骤(3)继续准备下一个字符并输出，直到没有数据

5.4.15 工作于 USB 设备模式与上位机 PC 机通讯
(1) 可选操作：CMD_SetupModule，设置模块配置，事件自动通知，可在功能配置时设定

(2) 可选操作：CMD_SetUsbId，设置 USB 设备 ID
(3) CMD_SetUsbMode，设置 USB 工作模式，进入 USB 设备模式
(4) 单片机系统等待模块发出事件状态通知或者定期查询模块的状态，如果收到事件状态码 ERR_USB_DAT_DOWN 则转到步骤(5)，如果收到事件状态码 ERR_USB_DAT_UP 则转到步骤(6)，如果需要退出 USB 设备模式则转到步骤(7)
(5) 发出 CMD_ReadUsbData 读取下传数据，然后转到步骤(4)
(6) 如果需要继续上传则发出 CMD_WriteUsbData 写入下一组数据，然后转到步骤(4)
(7) 如果需要切换到 USB 主机模式则由命令 CMD_SetUsbMode 回到 USB 主机模式

5.5 模块命令说明

以下是模块支持的操作命令的详细说明

5.5.1 常用的基本命令

```
#define CMD_GetVer              0x0A    /* 获取当前模块的版本号 */
```
命令成功后，返回 2 个字节的数据，依次是子程序库的版本号和模块的版本号。
```
    mCmdParam.GetVer.mFileLibVer    /* 模块所用的子程序库的版本号 */
    mCmdParam.GetVer.mModuleVer     /* 模块的版本号 */
    mCmdParam.GetVer.mUsbIcVer      /* USB 芯片版本:10H-CH375S,2xH-CH375A */
```

```
#define CMD_QueryStatus     0x60    /* 查询当前模块的状态 */
```
命令成功后，返回 12 个字节的数据，依次是 mLastStatus、mDiskStatus、mIntStatus、保留的一个字节、mFileSize、mCurrentOffset。
```
    mCmdParam.Status.mLastStatus    /* 上次操作的结果状态 */
```
模块上电复位或者重新复位及初始化后，是初始化的返回状态。正常操作后，是上次操作的结果状态。
```
    mCmdParam.Status.mDiskStatus    /* 磁盘及文件状态 */
```
模块中的全局变量，用于表示当前 CH375 以及磁盘的工作状态。如果操作结果返回错误，单片机系统也可以查询该状态，以便了解当前 CH375 或 U 盘的工作状态。状态定义请参考头文件。
```
    mCmdParam.Status.mIntStatus     /* CH375 操作的中断状态 */
```
当工作于 USB 设备模式时，如果没有设置事件自动通知，那么单片机系统应该定期查询模块的状态，根据 mIntStatus 的状态进行处理。如果 mIntStatus 为 ERR_USB_DAT_DOWN 则说明下传成功，如果为 ERR_USB_DAT_UP 则说明上传成功。
```
    mCmdParam.Status.mFileSize      /* 当前文件的长度，为 0xFFFFFFFF 说明是目录 */
    mCmdParam.Status.mCurrentOffset /* 当前文件指针，当前读写位置的字节偏移 */
```

```
#define CMD_DiskReady       0x71    /* 查询磁盘是否准备好 */
```
检查 U 盘是否准备就绪，大多数 U 盘不需要这一步，但是某些 U 盘必须要执行这一步才能允许数据读写，所以建议在 U 盘连接后先执行该命令，再进行文件读写。示例：
```
    i=ExecCommand( CMD_DiskReady, 0 );  /* 查询 U 盘是否准备好 */
    if ( i!=ERR_SUCCESS )  /* 还未准备好 */  else  /* 准备好了，可以读写数据 */
```

```
#define CMD_FileOpen        0x64    /* 打开文件或者目录 */
```
打开文件或者目录，输入参数在 mCmdParam.Open.mPathName 中提供文件名，包括完整的路径名。打开成功后，可以通过命令 CMD_QueryStatus 或者 CMD_FileQuery 查询当前文件的长度，如果文件长度为 0FFFFFFFFH，那么说明打开的是子目录，否则说明打开的是文件。例如
```
        C:\WINDOWS/SYSTEM/MYLIB.DLL
```

\TODAY1.TXT
\YEAR2004\MONTH05\DATE18.DAT

路径名和文件名的格式与DOS文件名格式相同，但是盘符和冒号可以省略，左斜杠与右斜杠等效，所有字符必须是大写，不能使用通配符，文件名长度不超过 11 个字符，其中主文件名不超过 8 个字符，扩展名不超过3个字符，如果有扩展名，则用小数点与主文件名隔开。由于模块RAM有限，路径名有长度限制，最大长度是MAX_PATH_LEN-1个字符。示例：

```
strcpy( mCmdParam.Open.mPathName, "\\YEAR2004\\CH375HFT.C" );
i=ExecCommand( CMD_FileOpen, MAX_PATH_LEN );  /* 打开文件，参数长度为最大值 */
if ( i!=ERR_SUCCESS )  /* 出错 */  else  /* 成功 */
```

如果路径名太长，那么可以分多次逐级打开，首先打开子目录，直到最后再打开文件，其中，首次打开必须是从根目录开始，所以路径名首字符必须是斜杠，以后接着前级再打开时的首字符必须不是斜杠。示例：打开文件"/YEAR2004/MONTH05/DATE18/HOUR08/ADC.TXT"

```
strcpy( mCmdParam.Open.mPathName, "/YEAR2004/MONTH05/DATE18" );  /* 目录名 */
i=ExecCommand( CMD_FileOpen, sizeof("/YEAR2004/MONTH05/DATE18") );  /* 打开目录 */
if ( i==ERR_SUCCESS ) {  /* 前3级子目录成功打开，下面接着打开下级目录及文件 */
  strcpy( mCmdParam.Open.mPathName, "HOUR08/ADC.TXT" );  /* 首字符不是斜杠 */
  i=ExecCommand( CMD_FileOpen, strlen(mCmdParam.Open.mPathName) );  /* 打开文件 */
}
if ( i!=ERR_SUCCESS )  /* 出错 */  else  /* 成功打开文件 */
```

如果文件名参数为"/"则可以打开根目录，从而便于自行处理长文件名，用完后必须关闭。

#define CMD_FileEnumer 0x63 /* 枚举文件,返回文件名 */

枚举文件（查询文件），输入参数在mCmdParam.Enumer.mPathName中指定带有通配符*的路径名和文件名，格式与命令CMD_FileOpen相同，枚举方法是，以通配符*代替需要查询的文件名中的全部或者部分字符，通配符*后面不能再有字符，而必须跟有枚举序号0到254，例如

C:\WINDOWS\SYSTEM*
  枚举C:\WINDOWS\SYSTEM\目录下的所有文件，例如ABC.EXE或者NEW.TXT等
\TODAY1.*
  枚举根目录下以TODAY1为主文件名的所有文件，例如TODAY1.C或者TODAY1
/YEAR2004/MONTH05/DATE1*
  枚举/YEAR2004/MONTH05/目录下以DATE1开头的所有文件，例如DATE12.TXT或者DATA1

由于符合条件的文件名通常会很多，所以需要指定序号，序号指定在通配符*之后，例如

\TODAY1.*\x0 （这是C语言表达方式，“\x0”代表一字节数据00H）
  枚举根目录下以TODAY1为主文件名的文件，返回搜索到的第1个匹配文件名
\TODAY1.*\x5 （这是C语言表达方式，“\x5”代表一字节数据05H）
  枚举根目录下以TODAY1为主文件名的文件，返回搜索到的第6个匹配文件名
\YEAR2004\DATE1*\x10 （这是C语言表达方式，“\x10”代表一字节数据10H）
  枚举\YEAR2004\目录下以DATE1为开头的文件，返回搜索到的第17个匹配文件名
\YEAR2004\DATE1*\xFF （这是C语言表达方式，“\xFF”代表一字节数据255）
  枚举\YEAR2004\目录下以DATE1为开头的文件，返回搜索到的由CMD_SetFileSize命令指定序号的匹配文件名，即当通配符*后的序号为255时，将由模块内子程序库的文件长度变量指定枚举序号

该命令计数到通配符*后的序号再返回搜索到的匹配文件名。枚举文件并不会打开文件，如果需要对搜索到的文件进行读写，可以再发出CMD_FileOpen命令打开返回的文件名。示例：

```
for ( c=0; c<255; c++ ) {  /* 最多搜索前255个文件 */
  strcpy( mCmdParam.Enumer.mPathName, "\\C51\\CH375*" );
  /* 在C51子目录下搜索以CH375开头的文件名，*为通配符 */
  for ( i=0; mCmdParam.Enumer.mPathName[i]!=0; i++ ); /* 指向文件名的结束符 */
  mCmdParam.Enumer.mPathName[i] = c;  /* 将结束符替换为搜索的序号，从0到255 */
  i=ExecCommand( CMD_FileEnumer, i+1 );  /* 文件名中含有通配符*，搜索文件 */
```

```
    if ( i==ERR_MISS_FILE ) break;  /* 再也搜索不到匹配文件，已经没有匹配的文件名 */
    if ( i!=ERR_SUCCESS && i!=ERR_FOUND_NAME ) break;  /* 出错 */
    printf( "found name %d#: %s\n", (unsigned int)c, mCmdParam.Enumer.mPathName );
    /* 搜索到相匹配的文件名，显示序号和搜索到的匹配文件名或者子目录名 */
  }
```

枚举更多数量的文件的例子：

```
  for ( int count=0; count<20000; count++ ) { /* 最多搜索前 20000 个文件*/
    mCmdParam.SetFileSize.mFileSize = count; /* 指定搜索的序号，几乎没有上限 */
    ExecCommand( CMD_SetFileSize, 4 );  /* 修改模块内子程序库的文件长度变量 */
    strcpy( mCmdParam.Enumer.mPathName, "\\*" ); /* 在根目录下搜索所有文件名*/
    i=strlen( mCmdParam.Enumer.mPathName ); /* 计算文件名长度，指向结束符*/
    mCmdParam.Enumer.mPathName[i] = 0xFF; /* 将结束符替换为 255 说明序号在变量中*/
    i=ExecCommand( CMD_FileEnumer, i+1 );  /* 文件名中含有通配符*，搜索文件 */
    if ( i==ERR_MISS_FILE ) break;  /* 再也搜索不到匹配文件，已经没有匹配的文件名 */
    if ( i!=ERR_SUCCESS && i!=ERR_FOUND_NAME ) break;  /* 出错 */
    printf( "found name %d#: %s\n", count, mCmdParam.Enumer.mPathName );
    /* 搜索到相匹配的文件名，显示序号和搜索到的匹配文件名或者子目录名 */
    ExecCommand( CMD_QueryStatus, 0 );  /* 查询模块状态 */
    if ( mCmdParam.Status.mFileSize!=0xFFFFFFFF ) printf( "file\n" ); /* 枚举到文件 */
    else printf( "this is a Sub-Directory\n" ); /* 枚举到子目录 */
  }
```

```
#define CMD_FileCreate      0x65    /* 新建文件并打开,如果文件已存在则先删除再新建 */
```

新建文件，输入参数在 mCmdParam.Create.mPathName 中指定新文件的路径名和文件名，格式与命令 CMD_FileOpen 相同，不支持通配符。如果存在同名文件，那么该同名文件将首先被删除，然后再新建文件。如果不希望已有文件被删除，那么应该事先发出 CMD_FileOpen 命令确认文件不存在后再新建。新建文件的属性默认为存档 ATTR_ARCHIVE，文件日期和时间默认为 2004 年 1 月 1 日 0 时 0 分 0 秒，文件默认长度为 1，如果需要修改则可以发出 CMD_FileModify 命令。示例：

```
strcpy( mCmdParam.Create.mPathName, "\\C51\\NEWFILE.TXT" );
/* 新文件名，在 C51 子目录下新建文件 NEWFILE.TXT，要求 C51 子目录已经事先存在 */
i=ExecCommand( CMD_FileCreate, sizeof("\\C51\\NEWFILE.TXT") );  /* 新建文件并打开 */
if ( i!=ERR_SUCCESS )  /* 出错 */  else  /* 成功 */
```

```
#define CMD_FileClose       0x67    /* 关闭当前文件 */
```

打开文件使用完毕后，应该关闭文件。对于读操作，关闭文件是可选操作。对于写操作，关闭文件的同时，可以让模块自动更新文件长度。在扇区模式下，自动更新的文件长度是以扇区为单位计算的，所以文件长度通常是 512 的倍数，mCmdParam.Close.mUpdateLen 为 1 时自动更新文件长度（如果已经对该文件执行写操作添加了数据），为 0 时不要自动更新文件长度。在字节模式下，自动更新的文件长度是以字节为单位，所以可以获得适当的长度。在扇区模式下，如果希望文件长度不是 512 的倍数,那么单片机可以在关闭文件前发出 CMD_FileModify 命令修改文件为指定的长度，并且在关闭文件时指定不要自动更新文件长度。对于以字节为单位的文件读写，关闭文件时能够自动更新为适当的文件长度，所以不需要执行 CMD_FileModify 修改文件长度。

5.5.2 扇区模式读写文件

```
#define CMD_FileLocate      0x6A    /* 以扇区为单位移动当前文件指针 */
```

移动当前已打开文件的指针，用于从指定位置读取数据，或者向指定位置写入数据。例如，单片机希望跳过文件的前 1024 字节再读取数据，那么可以在 mCmdParam.Locate.mSectorOffset 参数中输入 2，调用该命令将文件指针移动到 2 个扇区后，也就是从 1024 字节开始。对于写操作，如果单片机打算在原文件的尾部继续添加数据，而不希望影响前面的原有数据，那么可以指定一个

很大的扇区偏移，例如在 mCmdParam.Locate.mSectorOffset 参数中输入 0FFFFFFFFH，将文件指针移动原文件的末尾，以便追加数据。该命令将文件长度取整转换为扇区数后作为最大文件指针，如果文件长度不是 512 的倍数，那么文件尾部不足一个扇区的零碎数据部分将被忽略（可以通过 CMD_FileReadLast 命令读取尾部数据）。该命令返回时，mCmdParam.Locate.mSectorOffset 为当前文件指针所指向的数据在磁盘中的 LBA 扇区号，如果是 0FFFFFFFFH 则说明已到文件尾。

#define CMD_FileRead 0x6B /* 以扇区为单位从当前文件读取数据 */

从当前已打开文件中读取数据，每次读取后自动移动文件指针，第二次执行命令时将从第一次读取数据的后面继续读取数据。输入参数应该在 mCmdParam.Read.mSectorCount 中指定需要读取的扇区数，所以读取数据的长度总是 512 的倍数。该命令会根据文件长度自动检查文件是否结束，如果文件已经结束，那么返回时在 mCmdParam.Read.mSectorCount 中为实际读出的扇区数，所以判断 mCmdParam.Read.mSectorCount 如果变小就说明文件已经结束。如果文件长度不是 512 的倍数，那么文件尾部不足一个扇区的零碎数据部分将被忽略，如果必须读出文件尾部不足一个扇区的零碎数据，那么可以发出 CMD_FileReadLast 命令读取。

#define CMD_FileWrite 0x6C /* 以扇区为单位向当前文件写入数据 */

向当前已打开文件中写入数据，每次写入后自动移动文件指针，第二次执行命令将从第一次写入数据的后面继续写入数据。输入参数应该在 mCmdParam.Write.mSectorCount 中指定需要写入的扇区数，所以写入数据的长度总是 512 的倍数。该命令会检查文件结束簇，并且在需要时会自动分配磁盘空间以便继续写入。默认情况下（由 CMD_SetupModule 命令和 USB 功能配置决定），该命令只负责修改或者追加数据，而不修改文件长度。如果发出 CMD_FileWrite 命令写空数据（指定写入扇区数 mCmdParam.Write.mSectorCount 为 0），那么该命令将更新文件长度。如果文件长度不是 512 的倍数，那么可以在关闭文件前发出 CMD_FileModify 命令指定文件长度。示例：

```
mCmdParam.Locate.mSectorOffset = 0xffffffff;  /* 移动文件指针到文件末尾 */
i=ExecCommand( CMD_FileLocate, 4 );  /* 移动文件指针，以便在原文件的末尾追加数据 */
if ( i!=ERR_SUCCESS ) ?  /* 出错 */
mCmdParam.Write.mSectorCount = 2;  /* 写入 2 个扇区，也就是追加 1024 字节的数据 */
i=ExecCommand( CMD_FileWrite, 1 );  /* 向文件写入数据，实际数据传送另外实现 */
if ( i!=ERR_SUCCESS )  /* 出错 */  else  /* 成功 */
```

为了提高写操作的效率，尽量一次写入较大的数据块，扇区数为 64 或者更多。

#define CMD_FileReadLast 0x77 /* 从当前文件的尾部读取不足一扇区长度的零碎数据 */

在扇区模式下，如果文件长度不是 512 的倍数，那么文件尾部不足一个扇区的零碎数据部分将被忽略，如果必须读出文件尾部不足一个扇区的零碎数据，那么可以发出该命令读取。执行后返回实际读取的扇区数 mCmdParam.ReadLast.mSectorCount，为 1 则说明已经读取文件最后扇区中的零碎数据，为 0 则说明文件尾部没有零碎数据（文件长度是 512 的倍数）。

5.5.3 字节模式读写文件

#define CMD_ByteLocate 0x7A /* 以字节为单位移动当前文件指针，进入字节模式 */

移动当前已打开文件的指针，用于从指定位置读取数据，或者向指定位置写入数据。例如，单片机希望跳过文件的前 18 字节再读取数据，那么可以在 mCmdParam.ByteLocate.mByteOffset 参数中输入 18，调用该命令将文件指针移动到 18 个字节后，也就是紧接在后面的读写操作将从第 18 字节开始。对于写操作，如果单片机打算在原文件的尾部继续添加数据，而不希望影响前面的原有数据，那么可以指定一个很大的字节偏移，例如在 mCmdParam.ByteLocate.mByteOffset 参数中输入 0FFFFFFFFH，将文件指针移动原文件的末尾，以便追加数据。示例：

```
mCmdParam.ByteLocate.mByteOffset = 0xffffffff;  /* 移动文件指针到文件末尾 */
ExecCommand( CMD_ByteLocate, 4 );  /* 移动文件指针，以便在原文件的末尾追加数据 */
```

#define CMD_ByteRead 0x7B /* 以字节为单位从当前文件读取数据，进入字节模式 */

从当前已打开文件中读取数据，每次读取后自动移动文件指针，第二次执行命令时将从第一次读取数据的后面继续读取数据。输入参数应该在 mCmdParam.ByteRead.mByteCount 中指定需要读取的字节数，字节数不能超过 MAX_BYTE_IO 和 sizeof(mCmdParam.ByteRead.mByteBuffer)，命令执行后，读出的返回数据块被存放在 mCmdParam.ByteRead.mByteBuffer 中。该命令会自动检查文件是否结束，如果文件已经结束，那么返回时在 mCmdParam.ByteRead.mByteCount 中为实际读出的字节数，所以判断 mCmdParam.ByteRead.mByteCount 如果变小就说明文件已经结束。示例：

```
mCmdParam.ByteRead.mByteCount = 5;  /* 准备读出 5 个字节，如果返回小于 5 则文件结束 */
i=ExecCommand( CMD_ByteRead, 1 );  /* 以字节为单位从文件读出数据 */
if ( i!=ERR_SUCCESS )  /* 出错 */  else  /* 成功 */
if ( mCmdParam.ByteRead.mByteCount < 5 ) 已经到文件结尾，所以实际读出的长度变小了
/* 在 mCmdParam.ByteRead.mByteBuffer 中为读出的数据块 */
mCmdParam.ByteRead.mByteCount = 48;  /* 准备再读出 48 个字节，接着刚才的位置向后读 */
ExecCommand( CMD_ByteRead, 1 );  /* 以字节为单位从文件读出数据 */
```

#define CMD_ByteWrite 0x7C /* 以字节为单位向当前文件写入数据，进入字节模式 */

向当前已打开文件中写入数据，每次写入后自动移动文件指针，第二次执行命令将从第一次写入数据的后面继续写入数据。输入参数应该在 mCmdParam.ByteWrite.mByteCount 中指定需要写入的字节数，字节数不能超过 MAX_BYTE_IO 和 sizeof(mCmdParam.ByteWrite.mByteBuffer)，准备写入的数据块被存放在 mCmdParam.ByteWrite.mByteBuffer 中。该命令会自动检查文件是否结束，并且在需要时会自动分配磁盘空间以便继续写入。默认情况下（由 CMD_SetupModule 命令和 USB 功能配置决定），该命令只负责修改或者追加数据，而不修改文件长度。如果发出 CH375ByteWrite 命令写空数据（指定写入字节数 mCmdParam.ByteWrite.mByteCount 为 0），那么该命令将更新文件长度。示例：

```
mCmdParam.ByteWrite.mByteCount = 32;  /* 写入 32 个字节的数据 */
/* 将准备写入的数据块复制到 mCmdParam.ByteWrite.mByteBuffer 中 */
i=ExecCommand( CMD_ByteWrite, 1+32 );  /* 以字节为单位向文件写入数据 */
if ( i!=ERR_SUCCESS )  /* 出错 */  else  /* 成功 */
mCmdParam.ByteWrite.mByteCount = 7;  /* 再写入 7 个字节，如果为 0 则更新文件长度 */
/* 将准备写入的数据块复制到 mCmdParam.ByteWrite.mByteBuffer 中 */
ExecCommand( CMD_ByteWrite, 1+mCmdParam.ByteWrite.mByteCount );  /* 以字节为单位写 */
```

为了提高读写操作的效率，尽量一次读写较大的数据块，字节数不超过 MAX_BYTE_IO。

5.5.4 数据流模式读写文件

纯文本文件是最普通也最通用的文件格式，纯文本文件中只包含下述字符：数字 30H-39H、大小写字母 41H-7AH、回车 0DH、换行 0AH、各种汉字 80H-FFH、空格 20H、跳格 09H、各种标点符号等，通常纯文本文件可以用 WINDOWS 下的记事本、写字板、WORD 等各种编辑软件打开。

#define CMD_StreamRead 0x7E /* 数据流模式读文件,只支持串口,只支持文本 */

以数据流模式读取文件，只支持串口连接方式，文件格式应该是纯文本文件。如果执行该命令之前尚未打开文件，那么模块自动等待 U 盘插入，然后自动打开默认文件“/模块数据.TXT“后读取。如果执行该命令之前已经打开某个文件，那么模块将从已经打开的文件中的当前文件指针位置开始接着读取，所以，如果不希望使用默认文件名，那么可以事先打开指定文件。

该命令本身没有状态返回，执行该命令后，模块自动从文件中不断地读取字符并从串口输出，每输出一个字符，单片机系统必须发出一个应答码 SER_STREAM_ACK 后模块才能继续。如果模块在读取文件过程中出现错误，那么模块将发出一个错误码 SER_STREAM_ERROR 通知单片机系统；如果模块检查到文件结束，那么模块将发出一个结束码 SER_STREAM_END 通知单片机系统，然后自动完成该命令。在正常读取过程中，如果单片机系统希望回到文件开头重新读取（文件指针清 0），那么可以发出一个刷新码 SER_STREAM_FLUSH，模块将从文件第一个字符开始读取并输出。如果单片机系统希望提前结束该命令，那么可以发出一个结束码 SER_STREAM_END 通知模块完成该命令。模块完成该命令后并

不关闭文件，并且与其它常规命令不同，该命令完成后不返回任何操作状态。示例：

```
SendByte( SER_SYNC_CODE1 ); SendByte( SER_SYNC_CODE2 );  /* 发送三线制串口同步码 */
SendByte( CMD_StreamRead ); SendByte( 0 );  /* 命令码和参数长度，无参数 */
while ( 1 ) {  /* 读取文件直到错误或者文件结束，可以等待 U 盘插入并自动打开文件 */
    unsigned char c = RecvByte( );  /* 接收文件的下一个字符 */
    if ( c == SER_STREAM_END ) {  /* 文件已经结束 */
        /* SendByte( SER_STREAM_FLUSH ); continue; 可以发出刷新命令重新读取文件 */
        break;  /* 如果无需重新读取文件，那么退出 */
    }
    if ( c == SER_STREAM_ERROR ) { break; }  /* 出现错误，查询状态后处理或退出 */
    处理以数据流模式从文件中读取的字符 c，时间不限
    SendByte( SER_STREAM_ACK );  /* 应答，通知模块发送下一个字符 */
}
```

```
#define CMD_StreamWrite     0x7F    /* 数据流模式写文件,只支持串口,只支持文本 */
```

以数据流模式写入文件，只支持串口连接方式，文件格式应该是纯文本文件。如果执行该命令之前尚未打开文件，那么模块自动等待 U 盘插入，然后自动打开默认文件“/模块数据.TXT“后在其尾部追加数据，如果文件不存在那么自动创建空文件后再追加数据。如果执行该命令之前已经打开某个文件，那么模块将从已经打开的文件中的当前文件指针位置开始接着写入，所以，如果不希望使用默认文件名，那么可以事先打开指定文件，如果希望覆盖原数据，那么可以事先打开文件。

该命令本身没有状态返回，执行该命令后，模块通过串口不断地从单片机系统接收字符并写入缓冲区，每接收一个字符，模块都发出一个应答码 SER_STREAM_ACK，单片机发出字符并收到应答码后才能发出下一个字符。当模块的缓冲区中的字符满 64 字节或者 512 字节后，模块将其真正写入 U 盘并清空缓冲区；如果连续 10 秒（与晶体 X2 的频率有关）没有收到单片机系统发出的下一个字符，那么模块自动将缓冲区中的字符写入 U 盘，防止断电后数据丢失。使用缓冲区而不是直接将单个字符写入 U 盘，是因为 U 盘中的闪存有擦写次数的限制，频繁擦写 U 盘将缩短其使用寿命。如果模块在写入文件过程中出现错误，那么模块将发出一个错误码 SER_STREAM_ERROR 通知单片机系统。在正常写入过程中，如果单片机系统希望立即刷新缓冲区，那么可以发出一个刷新码 SER_STREAM_FLUSH，通知模块立即将缓冲区中的字符写入 U 盘，避免断电后数据丢失。如果单片机系统希望提前结束该命令，那么可以发出一个结束码 SER_STREAM_END 通知模块完成该命令。模块完成该命令后并不关闭文件，并且与其它常规命令不同，该命令完成后不返回任何操作状态。示例：

```
SendByte( SER_SYNC_CODE1 ); SendByte( SER_SYNC_CODE2 );  /* 发送三线制串口同步码 */
SendByte( CMD_StreamWrite ); SendByte( 0 );  /* 命令码和参数长度，无参数 */
while ( 1 ) {  /* 写入文件直到错误或者主动结束，可以等待 U 盘插入并自动追加文件 */
    unsigned char c, d;
    if ( 希望主动结束写文件 ) d = SER_STREAM_END;
    else if ( 即将断电，希望强制刷新模块中的缓冲区 ) d = SER_STREAM_FLUSH;
    else d = 准备写入的下一个字符，以数据流模式写入文件中，时间不限
    SendByte( d );  /* 将一个字符写入文件(先到缓冲区中)，或者向模块输出控制码 */
    c = RecvByte( );  /* 接收来自模块的应答 */
    if ( c == SER_STREAM_ERROR ) { break; }  /* 出现错误，查询状态后处理或退出 */
    if ( d == SER_STREAM_END ) { break; }  /* 单片机系统主动要求结束，那么退出 */
}
```

5.5.5 查询修改文件信息

```
#define CMD_FileQuery       0x68    /* 查询当前文件的信息 */
```

查询当前已打开文件的属性、日期、时间、长度等信息。命令成功返回时

mCmdParam.Modify.mFileSize 中是文件的长度，以字节为单位，长度可以是 0

mCmdParam.Modify.mFileDate 中是文件修改时间，格式参考 CH375HM.H 中的说明

mCmdParam.Modify.mFileTime 中是文件修改时间，格式参考 CH375HM.H 中的说明
mCmdParam.Modify.mFileAttr 中是文件属性，例如数值 01H 说明该文件是只读文件

#define CMD_FileModify 0x69 /* 查询或者修改当前文件的信息 */

与 CMD_FileQuery 对应，该命令用于修改当前已打开文件的属性、日期、时间、长度等。输入参数中，如果参数的新值指定为 0xFFFFFFFF，那么该参数将保持原值，否则将被修改为新值。

mCmdParam.Modify.mFileSize 指定新的文件长度，为 0FFFFFFFFH 则不修改，返回原长度
mCmdParam.Modify.mFileDate 指定新的文件日期，为 0FFFFH 则不修改，返回原日期
mCmdParam.Modify.mFileTime 指定新的文件时间，为 0FFFFH 则不修改，返回原时间
mCmdParam.Modify.mFileAttr 指定新的文件属性，为 0FFH 则不修改，返回原属性

例如在 mCmdParam.Modify.mFileSize 中指定 1105，而在其它单元中指定 0FFFFH，则将已打开文件的长度修改为 1105，但是不修改文件属性、时间和日期。示例：

```
mCmdParam.Modify.mFileAttr = ATTR_READ_ONLY;  /* 指定新的文件属性为只读 */
mCmdParam.Modify.mFileTime = 0xffff;  /* 不修改原文件时间 */
mCmdParam.Modify.mFileDate = MAKE_FILE_DATE( 2006, 3, 28 );
/* 通过宏生成文件日期的数据格式，指定新的文件日期是 2006.03.28 */
mCmdParam.Modify.mFileSize = 1105;  /* 指定新的文件长度是 1105 字节 */
i=ExecCommand( CMD_FileModify, 4+2+2+1 );  /* 修改当前文件的信息：属性/日期/长度 */
if ( i!=ERR_SUCCESS )  /* 出错 */  else  /* 成功 */
```

#define CMD_FileDirInfo 0x75 /* 存取当前已打开文件的目录信息 FDT */

用于存取当前已打开文件的目录信息 FDT，该信息为 32 字节的结构，结构成员数据在输入输出时必须都是小端格式。该命令可以用于修改文件名、设置文件创建日期和时间等。输入参数中，需要指定存取方式 mCmdParam.FileDirInfo.mAccessMode，其值为 0 时是读取文件目录信息，其值为 0F0H 时是写入/更新文件目录信息，替换原文件目录信息，模块不检查信息是否正确。示例：

```
mCmdParam.FileDirInfo.mAccessMode = 0;  /* 读取文件目录信息 */
ExecCommand( CMD_FileDirInfo, 4 );  /* 存取当前已打开文件的目录信息 */
/* 读取的 32 字节的文件目录信息在 mCmdParam.FileDirInfo.mDir 中，如果需要可以修改 */
/* 以下将修改过的文件目录信息写回到 U 盘中 */
mCmdParam.FileDirInfo.mAccessMode = 0xF0;  /* 写入/更新文件目录信息 */
ExecCommand( CMD_FileDirInfo, 4 );  /* 存取当前已打开文件的目录信息 */
```

#define CMD_SetFileSize 0x78 /* 修改模块内子程序库的文件长度变量 */

用于修改模块内子程序库的变量 CH375vFileSize，通常用于指定枚举文件的序号。不建议在打开文件修改或添加数据后直接修改该变量，如果直接修改将会在关闭文件或更新文件长度后生效。

5.5.6 偶尔用到

#define CMD_DiskSize 0x72 /* 查询磁盘总容量 */

查询磁盘物理总容量，执行成功后返回信息

mCmdParam.DiskSize.mDiskSizeSec 为整个物理磁盘的总扇区数，每扇区通常为 512 字节

#define CMD_DiskQuery 0x61 /* 查询磁盘信息 */

查询磁盘信息，该命令在 FAT32 文件系统的磁盘中执行时最快，在 FAT16 文件系统的磁盘中执行时最慢，可能需要几秒甚至更长时间。执行成功后返回信息

mCmdParam.Query.mDiskSizeSec	为整个物理磁盘的总扇区数
mCmdParam.Query.mTotalSector	为当前磁盘的总扇区数
mCmdParam.Query.mFreeSector	为当前磁盘的剩余扇区数，为 0 时说明磁盘已满。
mCmdParam.Query.mDiskFat	为当前磁盘（第一逻辑盘）的文件系统标志：0 对应于未知文件系统，1 对应于 FAT12，2 对应于 FAT16，3 对应于 FAT32

#define CMD_FileErase 0x66 /* 删除文件并关闭 */

删除当前已打开的文件或者指定文件名的文件。如果当前有文件已经打开或者尚未关闭，该命令直接删除该文件并关闭，如果当前没有文件被打开，那么应该在 mCmdParam.Erase.mPathName 中指定被删除文件的路径名和文件名，格式与命令 CMD_FileOpen 相同，不支持通配符。

#define CMD_DirCreate 0x76 /* 新建目录并打开,如果目录已经存在则直接打开 */

新建子目录，输入参数在 mCmdParam.DirCreate.mPathName 中指定新目录的完整路径名，格式与命令 CMD_FileOpen 相同，不支持通配符。如果存在同名子目录，那么将直接打开该子目录，否则将新建子目录，并自动产生自身目录项和上级目录项。新建目录的日期和时间默认为 2004 年 1 月 1 日 0 时 0 分 0 秒，如果需要修改则可以发出 CMD_FileModify 命令。示例：

```
strcpy( mCmdParam.DirCreate.mPathName, "/C51/NEW_DIR" );  /* C51 目录必须事先存在 */
i=ExecCommand( CMD_DirCreate, sizeof("/C51/NEW_DIR") );  /* 新建目录并打开 */
```

5.5.7 备用命令

#define CMD_BulkOnlyCmd 0x70 /* 执行基于 BulkOnly 协议的命令 */

这是底层协议操作命令，用于执行 BulkOnly 协议的磁盘命令，输入参数在 mCmdParam.BOC.mCBW 结构中，执行后返回信息在 mCmdParam.BOC.mCSW 结构中。示例：

```
mCmdParam.BOC.mCBW.mCBW_DataLen = 0;  /* 该命令没有数据传输 */
mCmdParam.BOC.mCBW.mCBW_Flag = 0x00;
mCmdParam.BOC.mCBW.mCBW_CB_Len = 6;  /* 命令块长度 */
mCmdParam.BOC.mCBW.mCBW_CB_Buf[0] = 0x1E;  /* 防止或者允许媒体移除的 SCSI 命令码 */
mCmdParam.BOC.mCBW.mCBW_CB_Buf[4] = 0x01;  /* 防止媒体移除 */
mCmdParam.BOC.mCBW.mCBW_CB_Buf[2] = mCmdParam.BOC.mCBW.mCBW_CB_Buf[1] = 0;
mCmdParam.BOC.mCBW.mCBW_CB_Buf[5] = mCmdParam.BOC.mCBW.mCBW_CB_Buf[3] = 0;
ExecCommand( CH375BulkOnlyCmd, MAX_PATH_LEN );  /* 执行 BulkOnly 协议的命令 */
```

5.5.8 模块硬件相关命令

#define CMD_SetupModule 0xA6 /* 设置模块配置，临时设定*/

输入参数在 mCmdParam.Setup.mSetup 中指定模块配置值，上电和复位后的默认配置值为 0。

模块配置值的位 0 用于设置事件自动通知。在 USB 主机模式下，是 U 盘或其它 USB 设备插拔的事件自动通知；在 USB 设备模式下，是下传数据成功和上传数据成功的事件自动通知。

在 USB 主机模式下，如果模块配置值的位 0 为 1 则空闲时模块自动查询 U 盘连接状态，如果检测到 U 盘连接或者断开事件，那么自动以中断方式通知单片机系统。在并口方式下，收到这种事件中断后，单片机系统可以发出 CMD_QueryStatus 命令或者其它命令使模块撤消中断（使模块的 INT#引脚恢复为高电平），否则模块也会在几十毫秒后自动撤消中断请求。在串口方式下，如果模块检测到 U 盘连接或者断开事件，那么会从串口发出事件通知状态码，单片机系统的串口将收到由模块发出的一个字节的事件状态码：ERR_USB_CONNECT 或者 ERR_DISK_DISCON。

在 USB 设备模式下，如果模块配置值的位 0 为 1，则当模块接收到上位机 PC 机的下传数据后或者当上位机 PC 机取走模块的上传数据后，模块自动以中断方式通知单片机系统，下传成功的中断通知为 ERR_USB_DAT_DOWN，上传成功的中断通知为 ERR_USB_DAT_UP。如果模块配置值的位 0 为 0，那么单片机系统只能定期查询模块状态，才能知道与 PC 机通讯的下传或上传成功。

模块配置值的位 4 对应于 U 盘模块中的全局变量 CH375LibConfig 的位 4，位 4 为 1 时，在用 CMD_FileWrite 或者 CMD_ByteWrite 向文件添加数据后，模块将自动更新文件长度，否则模块不更新文件长度，可以在需要时再通知模块更新或者修改文件长度。

模块配置值的位 7 和位 6 用于指定对外接口数据的字节顺序，位 7 为 0 位 6 为 1 指定小端格式 LITTLE_ENDIAN，位 7 为 1 位 6 为 0 指定大端格式 BIG_ENDIAN，位 7 与位 6 相同时不指定数据字节顺序，保持当前格式。通常情况下，如果是使用 C 语言的 MCS51 单片机，那么应该用大端格

式；如果是 80X86 兼容单片机或 PC 机、ARM 单片机、MSP430 单片机，那么应该用小端格式。
上述配置是变量型的软件设定，每次断电或者复位后都会失效而需要重新设定。新版 V1.4 以上模块可以在功能配置时通过计算机直接指定，是记忆型的硬件设定，不受断电或者复位的影响，从而可以不必在每次启动后由单片机系统执行该命令。

#define CMD_BaudRate 0xA5 /* 设置串口通讯波特率，临时设定 */

输入参数应该在 mCmdParam.BaudRate.mDivisor 中指定波特率除数，计算方法是“波特率除数=晶体 X2 的频率/32/波特率”。模块复位后默认为 120，在晶体 X2 的频率为 18.432MHz 时，波特率为 18432000/32/120=4800bps。如果波特率除数为 1，那么波特率为 18432000/32/1=576000。如果将 X2 的频率换成 12MHz，在波特率除数为 3 时，波特率为 12M/32/1=375000。在 CMD_BaudRate 命令执行完成后，外部单片机系统应该修改自身的波特率，然后等待 2mS 后再进行后续操作。
新版 V1.4 以上模块可以在功能配置时直接指定，不必在每次启动后由单片机执行该命令。

晶体 X2 频率	波特率除数	实际波特率	支持波特率
18.432MHz	120	4800	默认值
18.432MHz	60	9600	9600
18.432MHz	10	57600	57600
18.432MHz	5	115200	115200
18.432MHz	1	576000	576000
11.0592MHz	120	2880	默认值
11.0592MHz	3	115200	115200
22.1184MHz	120	5760	默认值
22.1184MHz	6	115200	115200
22.1184MHz	3	230400	230400
12MHz	120	3125	默认值
12MHz	1	375000	375000
24MHz	120	6250	默认值
24MHz	13	57692	57600
24MHz	1	750000	750000
36.864MHz	120	9600	默认值
36.864MHz	10	115200	115200
36.864MHz	1	1152000	1152000

#define CMD_ResetInit 0x0B /* 复位并重新初始化 CH375 以及模块，命令无返回 */

如果工作环境差易受干扰，或者命令执行后返回 ERR_CH375_ERROR，那么可以复位模块。

#define CMD_GetStringSN 0xA0 /* 获取产品序列号字符串 */

用于获取模块产品的序列号，该命令执行后在 mCmdParam.GetSN.mStringSN 中返回产品的序列号字符串，长度不超过 16 个字符。

5.5.9 连接计算机时 USB 设备模式命令

如果使模块工作于 USB 设备模式，那么单片机系统就可以通过模块与上位机 PC 机通讯。模块提供了两个数据通道，分别用于下传数据（从 PC 到单片机）和上传数据（从单片机到 PC）。由于 USB 是主从方式通讯，所以：下传数据是指 PC 机在需要时，将数据下传到模块的 CH375 芯片中，再由单片机系统从模块读取该下传数据块；上传数据是指，单片机系统将数据写入模块，也就是放于模块的 CH375 芯片中，当 PC 机需要时，它将从模块的 CH375 芯片中取走数据，实现被动意义上的“上传数据“，而不是主动意义上的由单片机系统发起的上传。为了及时了解 PC 机的主动操作，通常应该由 CMD_SetupModule 命令设置模块配置（位 0 为事件自动通知），使模块在收到下传数据和上传数据完成时，自动通知单片机系统，便于单片机系统及时取出下传数据和了解上传是否成功。

#define CMD_SetUsbId 0x12 /* USB设备模式：设置USB设备的厂商VID和产品PID */

该命令用于设置工作于USB设备模式下的模块的USB设备识别码，需要4个字节参数来指定厂商ID和产品ID。对于每一种USB设备，应该有唯一的厂商ID和产品ID，默认情况下，模块使用CH375芯片的默认ID，相应地也就使用CH372或CH375的WINDOWS通用驱动程序。如果需要使用另一种驱动程序，或者需要区别于已有的CH372或CH375芯片，那么可以设置新的ID代替默认ID。

#define CMD_SetUsbMode 0x15 /* USB设备模式：设置USB主机/设备模式,只支持串口 */

设置模块的USB工作模式，需要一个参数指定新工作模式：0为空闲模式或未启用的USB设备模式，2为USB设备模式，6为USB主机模式。模块上电或者复位后的默认模式为USB主机模式，可以用于读写U盘或控制其它USB设备，如果需要连接PC机，那么可以使模块工作于USB设备模式。

#define CMD_ReadUsbData 0x28 /* USB设备模式：从模块的数据下传端点读取数据块 */

从模块的CH375芯片中读取数据块，数据块通常是上位机PC机的下传数据。在USB设备模式下，当收到ERR_USB_DAT_DOWN事件通知时或者查询状态mIntStatus为ERR_USB_DAT_DOWN时，说明上位机已经下传数据，单片机系统应该及时从模块中读取下传数据块。

#define CMD_WriteUsbData 0x2B /* USB设备模式：向模块的数据上传端点写入数据块 */

向模块的CH375芯片中写入数据块，数据块通常是用于向上位机PC机上传。在USB设备模式下，当收到ERR_USB_DAT_UP事件通知时或者查询状态mIntStatus为ERR_USB_DAT_UP时，说明上位机已经取走上传数据，单片机系统可以根据需要接着上传下一组数据或者结束上传。

5.5.10 硬件底层备用命令

#define CMD_DirectWrCmd 0xB9 /* 直接传递给CH375,写命令 */

输入参数中，mCmdParam.Direct.mData为命令

#define CMD_DirectRdDat 0xB5 /* 直接传递给CH375,读数据 */

返回结果中，mCmdParam.Direct.mData为数据

#define CMD_DirectWrDat 0xB6 /* 直接传递给CH375,写数据 */

输入参数中，mCmdParam.Direct.mData为数据

6、参数

6.1. 绝对最大值（临界或者超过绝对最大值将可能导致模块工作不正常甚至损坏）

名称	参数说明	最小值	最大值	单位
TA	工作时的环境温度	-20	70	℃
TS	储存时的环境温度	-55	125	℃
VCC	电源电压（VCC接电源，GND接地）	-0.5	6.5	V
VIO	输入或者输出引脚上的电压 （低电压版模块请参考CH375HML.PDF）	-0.5	VCC+0.5	V

6.2. 电气参数（测试条件：TA=25℃，VCC=5V）

名称	参数说明	最小值	典型值	最大值	单位
VCC	电源电压	4.7	5	5.5	V
ICC	电源电流（不含U盘）		40	80	mA

VIL	低电平输入电压	-0.5		0.7	V
VIH	高电平输入电压 （低电压版模块请参考 CH375HML.PDF）	2.0		5.5	V
VOL	低电平输出电压（2mA 吸入电流）			0.5	V
VOH	高电平输出电压（200uA 输出电流）	2.8			V
IUP	内置上拉电阻的输入端的输入电流	30	100	200	uA
IDN	内置下拉电阻的输入端的输入电流	-30	-50	-100	uA

6.3. 时序参数（测试条件：TA=25℃，VCC=5V）

名称	参数说明	最小值	典型值	最大值	单位
TPR	电源上电的复位时间	20	100	200	mS
TPR1	由 uP 监控 MAX810 提供复位的模块 电源上电的复位时间	200	300	500	mS

7、其它说明

⑴ 为了提高处理效率和速度，建议对 USB 闪存盘使用 FAT12 或者 FAT16 文件系统。对于容量较大的 U 盘，为了节约 U 盘空间，减少浪费，可以使用 FAT32 文件系统。

⑵ 在 WINDOWS 2000 或者 XP 下的控制面板/管理工具/计算机管理中有磁盘管理工具，可以将 U 盘格式化成指定的 FAT12、FAT16 或者 FAT32 文件系统，当总容量除以分配单元大小后的结果小于 4085 时是 FAT12，大于 65525 时是 FAT32，否则是 FAT16。分配单元较大时，通常读写效率稍高，分配单元较小时，通常会节约磁盘容量。

⑶ 如果操作 USB 外置硬盘或者耗电较大的 USB 闪存盘，需要考虑其电源供应，确保提供足够的工作电流，否则在其插入过程以及读写过程中会导致电源电压波动，甚至导致 CH375 模块以及单片机复位，建议在电源与地之间并联较大的电解电容，或者为 U 盘或者 USB 外置硬盘单独提供足够的电源。

⑷ 单片机的 C 语言效率比汇编语言低，以 MCS51 单片机为例，纯 C 语言数据读写的速度可能只有汇编语言速度的一半，所以，根据需要部分子程序可以嵌入汇编。

⑸ 串口方式下可以由功能配置设定通讯波特率或者由 CMD_BaudRate 命令临时修改通讯波特率，如果必要还可以自行调换晶体 X2，其频率范围为 5MHz 到 36MHz，建议优先选用与外部单片机的时钟频率相同或者成倍数关系的晶体。而在并口方式下 X2 的频率范围只能为 10MHz 到 20MHz。在满足传输速度的前提下，优先选用较低频率的晶体可以降低单片机以及整个模块的功耗。

⑹ 通常情况下，单片机系统的 RAM 内存越大读写效率越高，如果单片机系统的空闲 RAM 多于 1K 字节，那么建议采用标准的以扇区为单位的文件读写命令，读写速度较快。如果单片机系统的空闲 RAM 少于 512 字节，那么可以采用以字节为单位的文件读写命令，读写速度较慢，但是使用方便，无论通过并口或者串口连接，只要外部单片机系统的 RAM 不少于十几个字节，就可以通过模块读写 U 盘中的文件。

⑺ 以字节为单位的文件读写命令，优点是对单片机的 RAM 没有要求，最少十几个字节即可，缺点是速度比以扇区为单位的文件读写慢，并且频繁地向 U 盘中的文件写数据，会缩短 U 盘中闪存的使用寿命（因为闪存只能进行有限次擦写）。

⑻ 如果用串口监控调试工具通过计算机的串口试用模块，那么可以使用三线制串口连接方式，并且禁止“检查串口超时”，这样，就可以在计算机上手工输入命令码和参数，注意，在每个命令码之前，两个串口同步码字节 57H 和 ABH 必须紧接在一起从串口首先发出，然后才是命令码等。例如 57H、ABH、60H、00H 是查询状态命令 QueryStatus。

⑼ 数据流模式进一步简化了模块的外部接口协议，可以作为设备级接口，连接单片机仪器、PLC 设备、PC 机/工控机等。数据流读文件可以用于流式控制设备，数据流写文件可以用于记录设备状态信息。除此之外，特定应用还可以在此基础上定制功能。

⑽ 虽然 U 盘模块最大支持 1GB 的文件，但是为了提高效率，建议单个文件的长度不要超过 100MB，

通常在几 KB 到几 MB 范围是比较正常的。如果文件较多，建议使用多级目录结构分类管理，创建多个子目录，将具有部分相似性质的多个文件放在同一个子目录下，采用目录分类管理可以缩短打开、枚举搜索或创建文件的时间。

8、示例程序

单片机系统与 U 盘文件读写模块之间的电路连接方式，可以有 8 种：

⑴ 并口连接，INT#中断通知，提供 MCS51 单片机的示例程序；
⑵ 并口连接，INT#查询状态，提供 MCS51 单片机的示例程序；
⑶ 串口连接，INT#中断通知，暂无示例程序；
⑷ 串口连接，INT#查询状态，暂无示例程序；
⑸ 串口连接，串口中断通知，暂无示例程序；
⑹ 串口连接，串口等待状态，提供 MCS51 单片机的示例程序；
⑺ 三线制串口连接，串口中断状态，暂无示例程序；
⑻ 三线制串口连接，串口等待状态，提供 MCS51 单片机的示例程序；
⑼ 4+1 线 SPI 连接，INT#查询状态，提供 MCS51 单片机的示例程序；
⑽ 4 线 SPI 连接，SPI 操作查询状态，提供 MCS51 单片机的示例程序。

其中，方式 7 和 8 的电路连接方式最简单，只需要连接两个信号线：SIN 和 SOUT，并且 INT#接低电平；方式 5 和 6 的电路连接方式也比较简单，需要三个信号线：SIN，SOUT 和 STA#；这 4 种串口连接方式都不需要连接中断信号线 INT#。

示例程序提供 C 或者 ASM 源程序和 HEX 目标程序，适用于 89C5X 等类似单片机。

在 PARA_INT 子目录中的 CH375HMP.C 是方式 1 的 C 示例程序，演示以扇区为单位对 U 盘文件进行读写，读写速度较快，需要至少 512 字节的外部 RAM。

在 PARALLEL 子目录中的 CH375HMP.C 是方式 2 的 C 示例程序，演示以扇区为单位对 U 盘文件进行读写，读写速度较快，需要至少 512 字节的外部 RAM。

在 PARA_ASM 子目录中的 CH375HMP.ASM 是方式 2 的汇编示例程序，演示以扇区为单位对 U 盘文件进行读写，读写速度较快，需要至少 512 字节的外部 RAM。

在 SERIAL 子目录中的 CH375HMS.C 是方式 6 的 C 示例程序，演示以扇区为单位对 U 盘文件进行读写，读写速度一般，需要至少 512 字节的外部 RAM。

在 SER_BYTE 子目录中的 CH375HMS.C 是方式 6 的 C 示例程序，演示以字节为单位对 U 盘文件进行读写，读写速度较慢，单片机只需要几十个字节的内部 RAM 就可以了。

在 SER_SYNC 子目录中的 CH375HMS.C 是方式 8 的 C 示例程序，演示以字节为单位对 U 盘文件进行读写，读写速度较慢，单片机只需要几十个字节的内部 RAM 就可以了。

在 SER_ASM 子目录中的 CH375HMS.ASM 是方式 8 的汇编示例程序，演示以字节为单位对 U 盘文件进行读写，读写速度较慢，单片机只需要几十个字节的内部 RAM 就可以了。

在 SER_ADC 子目录中的 CH375HMS.C 是方式 8 的 C 示例程序，演示以字节为单位对 U 盘文件进行读写，读写速度较慢，单片机只需要几十个字节的内部 RAM 就可以了。该程序用于演示单片机将 ADC 模数采集的数据以字符串文本形式保存到 U 盘文件中。

在 SER_PC 子目录中的 CH375HMX.C 是工作于 USB 设备模式的示例程序，演示 USB 主机和设备模式的切换，与 PC 机的通讯等。

在 SER_INFO 子目录中的 CH375HMS.C 是方式 8 的 C 示例程序，演示处理文件目录项，修改文件名，设置文件的创建日期和时间等。

在 SPI5 子目录中的 CH375HMI.C 是方式 9 的 C 示例程序，演示以扇区为单位对 U 盘文件进行读写，读写速度一般（此程序为软件模拟 SPI，换成硬件 SPI 则较快），需要至少 512 字节的外部 RAM。

在 SPI4 子目录中的 CH375HMI.C 是方式 10 的 C 示例程序，演示以字节为单位对 U 盘文件进行读写，读写速度较慢，单片机只需要几十个字节的内部 RAM 就可以了。

除此之外，还可以参考 CH375 评估板资料中子程序库的示例程序。

其它示例程序可以参照修改，相对而言，串口连接方式的程序容易理解和编写。

在文件数据读写速度方面，如果单片机系统本身没有限制，可以参考以下数据。以扇区为单位进

行文件读写（写比读慢），并口连接方式的最高速度可达 100K 到 200K 字节；而串口连接方式的速度通常为串口波特率的 70%到 95%，在 700Kbps 波特率下的速度可达 50K 字节左右，在 115200bps 波特率下的速度约为每秒 10K 字节；SPI 连接方式的速度通常为串口时钟 SCK 速率的 80%到 95%，在 SCK 频率为 1MHz 时速度可达每秒 70K 到 95K 字节（如果模块单片机晶体频率为 36.864MHz 则速度加倍）；在 SCK 频率为 100KHz 时速度约为每秒 10K 字节。如果以字节为单位进行文件读写，那么占用 RAM 资源较少，但是读速度可能下降为原来的 30%到 80%，写速度可能下降为原来的 10%到 40%。

如果将 CH375 芯片换成 CH374 芯片，并且提高单片机的时钟频率，那么速度还可以提高 1 倍。

9、模块功能演示

9.1. 脱机自行演示

模块内置了简单的自动演示功能，方法是：在模块处于正常工作状态时（此时跳线 J1 是断开的，模块的 LED 亮起说明其空闲），将跳线 J1 插上，模块开始自动演示。

开始演示后，模块的 LED 灭掉，等待 U 盘插入，当 U 盘插入后，模块打开 U 盘中根目录下的“模块演示.TXT”文件，如果文件不存在则新建文件，如果文件存在则移动文件指针到末尾，添加两行字符串，其中包含当前状态信息，然后关闭文件。上述演示结束后，模块的 LED 重新亮起。如果演示过程中出现错误，那么模块的 LED 会闪烁。

演示完毕后，只有断开电源或者复位，模块才能恢复正常工作状态。

模块的演示结果中包括模块程序版本号和当前功能配置值以及串口通讯波特率除数，所以也可以用于检查模块当前接口以及功能配置是否设置正确。

9.2. 串口联机演示

配套的“模块调试”工具可以用于在计算机端通过串口控制和操作模块以及演示模块功能。比较方便的是使用串口版模块进行功能演示，只需要用 3 线串口交叉线连接模块的串口与计算机串口并提供电源给模块即可，如果是标准版模块，那么还应该额外使用 TTL 电平与 RS232 电平的转换电路。

操作步骤是：执行该工具程序，选择串口并打开，直接点击所需演示的按钮，或者选择手工输入命令码进行操作，建议直接使用已经预置好的常用命令按钮进行演示。