USB 接口芯片 CH372（或 CH375 设备方式）
数码管驱动及键盘控制芯片 CH452（或 CH451）
演示板说明及应用参考

版本：2D

http://wch.cn

1、概述

演示板采用 USB 总线接口芯片 CH372（或 CH375）、数码管驱动和键盘控制芯片 CH452（或 CH451）、通用的 MCS51 系列单片机构成，用于演示 CH372 的 USB 数据通讯功能、CH452 的数码管驱动/键盘扫描功能、以及 CH372 与单片机的连接、CH452 与单片机的连接等。

CH372 是 CH371 的升级产品，数据缓冲区更大，传输速度更快，除此之外，CH372 还支持 USB 产品制造商自定义厂商 ID 和设备 ID。CH375 包含 CH372 的全部功能，所以本说明也适用 CH375。

CH372 或者 CH375 必须在单片机或者 DSP 的控制下才能进行 USB 通讯，如果仅仅是实现一个简单的、低速的 AD 模数采集、I/O 控制等，那么可以用 CH341 芯片，CH341 可以不需要单片机和 DSP 就能独立实现 USB 通讯，CH341 通过异步串口、并口、兼容 I2C 两线同步串口等连接外部逻辑电路。

CH451 与 CH452 引脚兼容，时序基本兼容，所以本说明也基本适用于 CH451 芯片。除了 4 线接口，CH452 芯片还支持兼容 I^2C 的两线串口。相比之下，CH452 附加功能更多、价格更低，CH451 接口速度更快、驱动能力更强。类似的芯片还有 CH450（仅 6 个数码管+键盘扫描）、CH422（I/O 扩展或者仅 4 个数码管）、CH423（I/O 扩展可支持 16 个数码管）等，性能比较可以参考 CH451 目录下 ADVERT 目录中的 CH451PLN.PDF 等相关文档。

2、演示板的原理图

原理图和 PCB 请分别参考 CH372SCH.PDF 文档，内有当前版本和以前版本的原理图。

图中，元器件说明如下：

在本演示板中，USB 芯片的位置为 DIP28 插座，可以插入 CH372 转换板或者 CH375 转换板，互为代替。默认情况下只提供 CH372 的 SSOP20 转 DIP28 转换板，如果需要可以另外提供 CH375 的 SOP28 转 DIP28 转换板。在 CH372SCH.PDF 文档中有该转换板的原理图。

在本演示板中，CH452 芯片的位置可以使用 DIP24S 封装的 CH452L 或者 CH451L（以前版本的演示板还提供 DIP28 插座），互为代替。默认情况下，随板只提供 CH452L 或者 CH451L 中的一种，如果需要可以另外一种或者提供 SOP28 贴片封装的 CH452S 或者 CH451S。

晶体 X1 为标准的 12MHz，误差要求小于 1‰，普通的 12MHz 晶体都可以满足要求；振荡电容 C1、C2 的容量大小会少量影响振荡频率。如果使用 CH372 转换板或者 CH375 转换板，转换板中通常会自带晶体和电容。强烈建议应用电路中尽可能缩短时钟引线，减少干扰。

电容 C12 的容量可以是 1000pF～0.1uF，用于 CH372 或 CH375 内部电源节点退耦，可以降低 USB 传输过程中的 EMI，如果要求不高也可以省去 C12。

如果调试过程中使用单片机仿真器，或者演示板由 P2 口或者 P4 口从外部供电，那么电阻 R1 可以防止在接入 USB 电源时，两组电源电压不同而产生的较大的短路电流。

U4（单片机 89C51 等）用于测试 CH372 的 USB 数据通讯功能，本例中 CH372 的地址 A0 由 U4 的 P20 驱动，CH372 的片选线 CS#由 U4 的 P21 驱动，所以 CH372 的命令口的地址可以是 FDXXH（例子程序中使用地址 BDF1H），数据口的地址可以是 FCXXH（例子程序中使用地址 BCF0H）。CH372 的 INT#与单片机 U4 的 INT0 相连接，U4 通过 CH372 的被动并行接口与 CH372 进行数据交换。

如果是 CH375，那么 L1（发光二极管 LED）用于指示 CH375 的配置状态，当连接 USB 总线并且配置成功，CH375 的 ACT#引脚将输出低电平，点亮 LED。CH372 没有 ACT#引脚。

跳线 J2 用于选择单片机 U4 的上电复位信号的来源，CH375 和 CH451 都具有上电复位的功能。如果需要引出 28 脚封装的 CH451 的 RSTI（第 28 脚），建议在其与 VCC 或者 GND 之间跨接一个几百 pF

的电容，以防止高频干扰。

J5 预留给支持 CH375/CH372/CH374 的 MCS51 单片机通过 USB 进行程序下载，J5 默认是断开的，先短接 J5 再通过 P1 端口的 USB 连接计算机，3 秒内再断开 J5，即可支持 USB 下载。

电阻 R5 为数码管的限流电阻，阻值可以在 60Ω至 1KΩ之间选择，阻值为 200Ω时限定段电流为 13mA 左右，相应的字电流峰值为 104mA；电阻 R6 为按键扫描的限流电阻，阻值可以在 1KΩ至 10KΩ之间选择，本例中为 2KΩ，如果不使用 CH452 的按键扫描功能，则电阻 R6 可以省去。

N1 至 N8 为 8 个共阴数码管，K0 至 K63 为轻触按钮（部分行已被略去），各代表一个按键。

整个演示板通过普通的 USB 对连线与 PC 机进行通讯，并且可以由 PC 机 DEMO 应用程序控制其进行功能演示。该 USB 对连通讯线的两个端头是一样的，一头插在演示板的 P1 端口中，另一头插在 PC 机的任意一个 USB 端口中。

3、演示板的使用说明

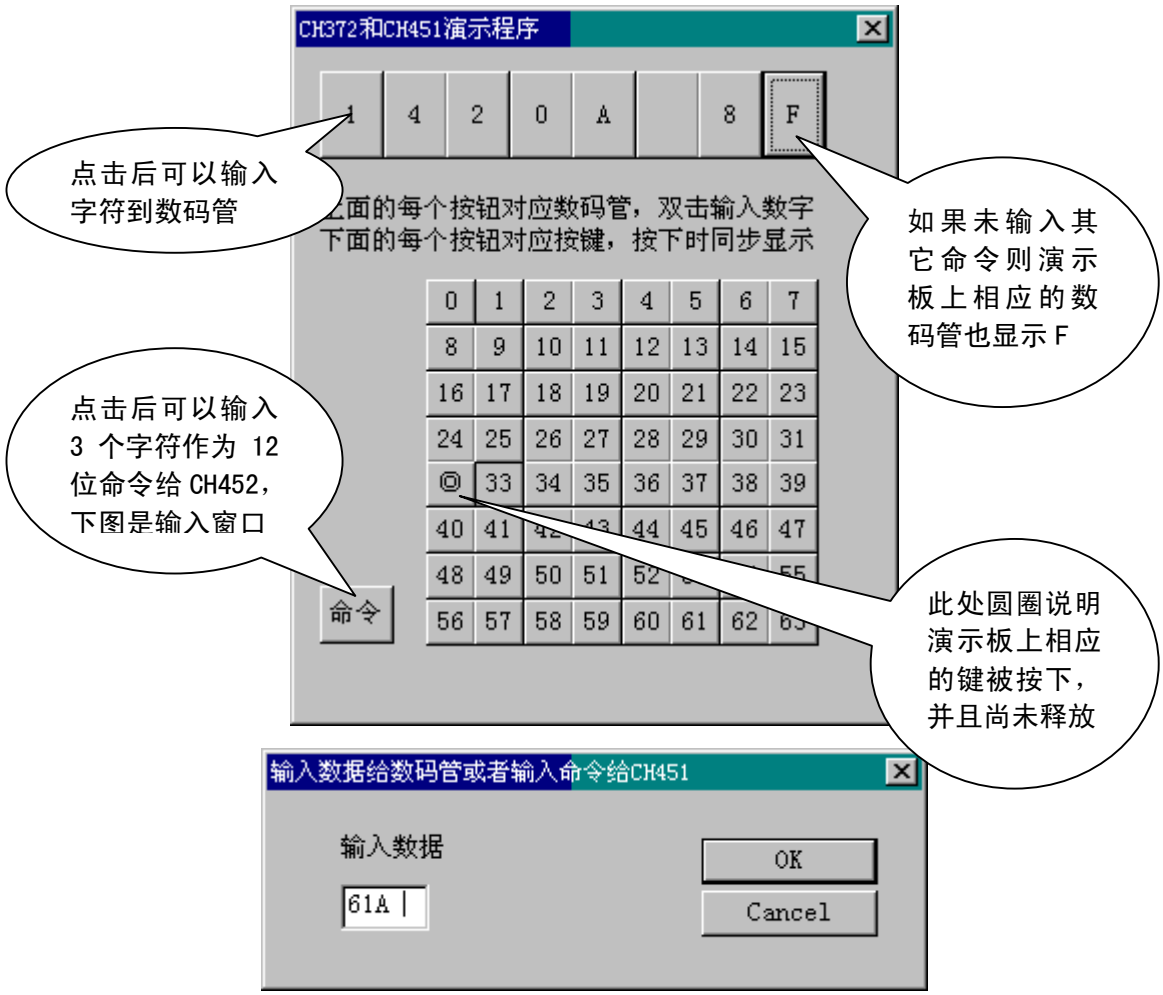
如果演示板上没有 89C51 芯片，请用编程器将 HEX 目标程序代码 CH375451.HEX 写入单片机，然后将其插到演示板上。单片机的程序在 DEMO\MCU 子目录下，相应的汇编源程序是 CH375451.ASM，C 语言源程序是 CH375451.C。

用 USB 通讯线连接演示板和 PC 机，WINDOWS 将提示找到新硬件，只要指定 CH372 的驱动程序文件所在的目录 CH372/DRIVER/DRIVER，WINDOWS 就会安装驱动程序，或执行驱动安装包 CH372DRV.EXE 进行安装。演示板使用了 CH375 的通用驱动程序和动态链接库（可以支持 WINDOWS 98/ME/2000/XP），包括三个主要文件：安装信息 CH375WDM.INF、驱动程序 CH375WDM.SYS、动态链接库 CH375DLL.DLL，其中的动态链接库不是必要的，如果应用程序中使用了该库才需要安装。

安装好 CH375 的驱动程序后，查看 WINDOWS 系统的设备硬件列表，将多显示“外部接口”项，其下标有“USB 设备 CH372/CH375”的设备就是演示板或者其它 CH372 设备。

确定有 CH372 设备后，就可以执行功能演示程序 DEMO.EXE，该应用层程序仅仅是一个简单的功能演示程序，相应的 C 语言源程序是 DEMO.C 以及 DEMO.H。DEMO 程序首先打开 CH372 设备，测试与单片机的数据通讯是否正常，最后安装中断服务程序并进入事件驱动的消息循环。

新版的 DEMO 演示程序支持设备插拔事件通知，可以自动检测该演示板的插入与拔出。新版演示工具还提供 CH452 和 CH451 芯片的常用命令按钮，可以直接点击查看其效果。



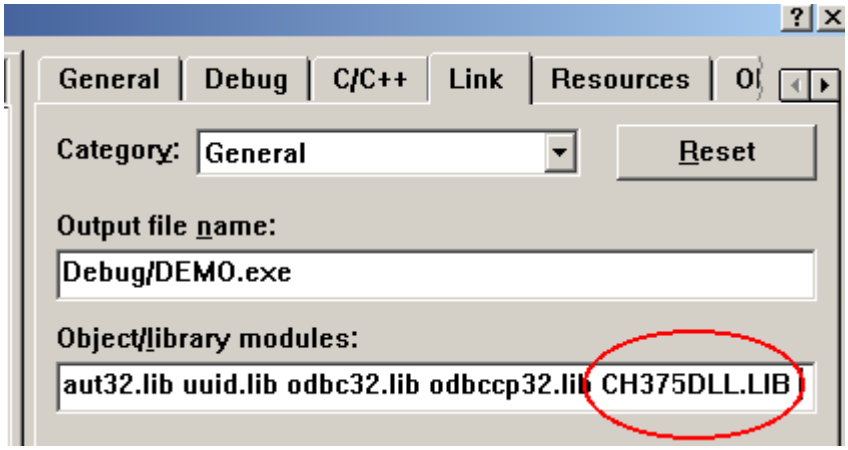
点击数码管相应的按钮，输入一个字符（后面可以加一个小数点字符），该字符将会被输出到演示板上的相应位置的数码管。演示板上的单片机 U4 在启动时将 CH452 设置为 BCD 译码方式。如果输入字符 E 则数码管显示 E；如果输入字符 4.(字符 4 后有小数点)则数码管显示 4.。

点击命令按钮时，需要输入 3 个字符，字符为十六进制字符 0 至 9 或者 A 至 F，将被解释为 12 位的命令然后传送给 CH452。如果输入 3 个字符 61A，则 CH452 收到的 12 位命令是 011000011010，结果是 CH452 控制数码管 N4、N5、N7 闪烁；如果输入字符 581，则 CH452 收到的 12 位命令是 010110000001，结果是 CH452 设定显示占空比为 1/16，也就是显示亮度调整为最低档。

当按下演示板上的按键时，DEMO 程序将在对应的键位显示“◎”；当释放该按键或者按下其它键后时，对应的键位才恢复显示为键号。这个演示是由 CH372 驱动程序提供的中断服务实现的。

建议在退出应用程序后，再拔插演示板的 USB 通讯线（关闭演示板）。

该应用层程序的 C 语言源程序是 DEMO.C 以及 DEMO.H，设计 WINDOWS 应用程序时可以作为参考。如果调用 CH375 提供的动态链接库，则要用到 CH375DLL.H 和 CH375DLL.LIB 两个文件。对于一般的 VC 编译环境，将上述两个文件放置在应用程序的同一目录下，然后参考下图将 CH375DLL.LIB 添加到应用程序的链接库中，当应用程序被编译时就会自动链接。



如果不使用工程文件，则可以参考 DEMO 的 MAKEFILE 文件。

如果需要，可以提供 VB、BC、DELPHI 或者其它语言的应用程序示例。

4、没有 PC 机情况下的演示

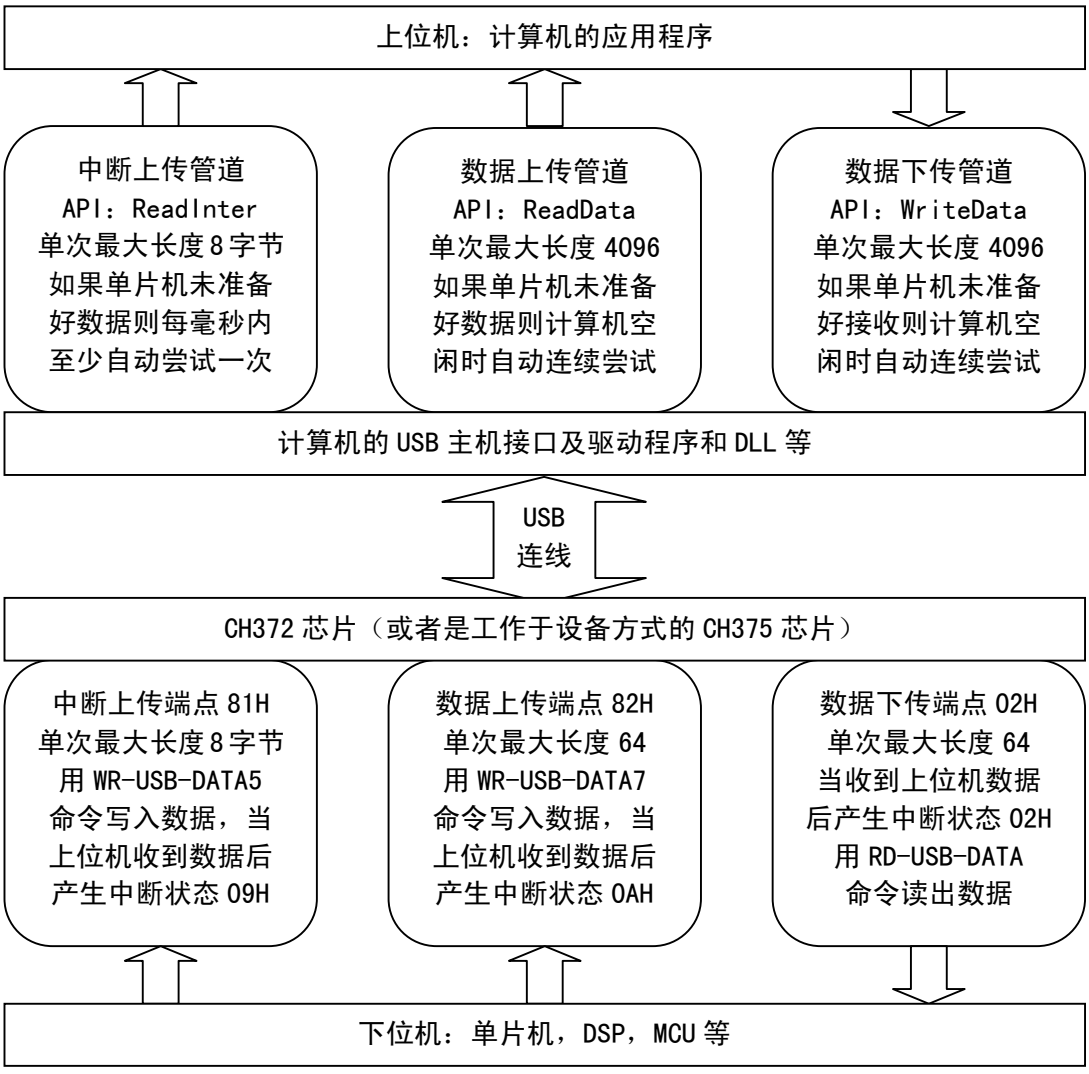
如果不连接 PC 机，可以由单片机控制 CH452 进行简单的功能测试。这种方式下，P2 的第 5 脚和第 6 脚应该相连接（插上短路子），也就是将单片机 U4 的 T1 引脚接地，单片机工作于自行测试方式。5V 工作电源可以通过 P2 的第 1 脚和第 2 脚输入。

自行测试方式下，U4 在检测到按键时，将原显示内容左移两位，空出的两位显示新收到的按键值。这种方式下 U4 不处理来自 CH372 的数据请求。

如果使用其它单片机，通过 P4 口连接电源和数据线可以测试 CH452 的功能。P5 口将 CH452 的段驱动和字驱动引出，可以用于反相后驱动共阳数码管和扩展电流。

5、有关 USB 应用程序的设计参考

CH372 提供了 3 个相互独立的端对端的逻辑传输通道，USB 产品的设计人员可以根据需要自行定义各个通道的用途，并在上位机与下位机之间约定各个通道中的数据格式。对于 CH372A 和 CH372B 芯片，还有一个额外的逻辑传输通道“辅助数据下传管道”，类似于数据下传管道，只是下位机的单次最大长度为 8 字节，端点地址和中断状态为 01H。



(1)、动态库 DLL 提供的主要 API

```
HANDLE  WINAPI  CH375OpenDevice(  // 打开 CH375 设备,返回句柄,出错则无效
    ULONG    iIndex );  // 指定 CH375 设备序号,0 对应第一个设备
VOID    WINAPI  CH375CloseDevice(  // 关闭 CH375 设备
    ULONG    iIndex );  // 指定 CH375 设备序号
```

在使用 CH372/CH375 设备之前，必须先打开，用完后再关闭。如果有多个 CH372/CH375 设备，那么可以在 iIndex 指定序号进行区分，先连接的是 0#，接着是 1#，依次排序。为了在已排序的多个 USB 设备中找到特定功能的 USB 设备，应用程序可以按设备序号依次查询，直到某 USB 设备对自定义的识别命令产生特定的响应，就可以认为是找到。对于多线程应用程序，可以先获取 CH375 的设备名称后，再通过 WIN32-API 的 CreateFile 以异步方式打开 CH375 设备。

```
PVOID   WINAPI  CH375GetDeviceName(  // 返回指向 CH375 设备名称的缓冲区,出错则返回 NULL
    ULONG    iIndex );  // 指定 CH375 设备序号,0 对应第一个设备
```

打开 CH375 设备后，应用程序通过该 API 可以获得其名称，是一个由系统自动生成的较长字符串的名称，有了设备名称后，可以用于多线程应用程序以 CreateFile 自行打开 CH375 设备并通讯。

```
BOOL    WINAPI  CH375ResetDevice(  // 复位 USB 设备
    ULONG    iIndex );  // 指定 CH375 设备序号
```

出现严重的 USB 通讯错误时才需要调用，一般不需要用到。

```
BOOL    WINAPI  CH375ReadData(  // 读取数据块
    ULONG    iIndex,  // 指定 CH375 设备序号或者打开设备的句柄
    PVOID    oBuffer,  // 指向一个足够大的缓冲区,用于保存读取的数据
    PULONG   ioLength );  // 指向长度单元,输入时为准备读取的长度,返回后为实际读取的长度
```

从 CH375 端点 2（批量上传管道）的上传缓冲区中读出数据，调用前应该确定单片机已经或者即将用 WR-USB-DATA7 命令写入上传数据。如果单片机还没有写入上传数据则一直等待，如果单片机进入中断服务程序中没有释放缓冲区也会一直等待。一次读取的最大长度是 4096 字节，实际上被分解为不超过 64 字节的小数据包进行多次传输，如果单片机写入的某个数据包不足 64 字节，那么将提前结束读操作。为了避免长时间等待，可以设置 USB 读数据超时，使得在超时指定时间仍未读完数据时提前终止。这是“立即上传”模式，也就是调用 CH375ReadData 将立即尝试执行一个 USB 上传。
如果已经通过 CH375SetBufUpload 设置“缓冲上传”模式，那么调用 CH375ReadData 将只是到内存缓冲区中查看是否已经收到数据（在“缓冲上传”模式下有一个系统线程专门负责无间隔 USB 上传并暂存到该内存缓冲区中），有数据则取走，没有数据也不会等待。注意不要连续频繁查询。

```
BOOL    WINAPI  CH375WriteData(  // 写出数据块
    ULONG    iIndex,  // 指定 CH375 设备序号或者打开设备的句柄
    PVOID    iBuffer,  // 指向一个缓冲区,放置准备写出的数据
    PULONG   ioLength );  // 指向长度单元,输入时为准备写出的长度,返回后为实际写出的长度
```

向 CH375 端点 2（批量下传管道）的下传缓冲区中写入数据，调用前应该确定单片机已经或者即将用 RD-USB-DATA 命令将前次的下传数据取走。如果单片机还没有取走前次的下传数据则一直等待。一次写入的最大长度是 4096 字节，实际上被分解为不超过 64 字节的小数据包进行多次传输。为了避免长时间等待，可以设置 USB 写数据超时，使得在超时指定时间仍未写完数据时提前终止。这是“立即下传”模式，也就是调用 CH375WriteData 将立即尝试执行一个 USB 下传。
如果已经通过 CH375SetBufDownload 设置“缓冲下传”模式，那么调用 CH375WriteData 将只是将待发送数据送到内存缓冲区中（在“缓冲下传”模式下有一个系统线程专门负责将该内存缓冲区中的数据进行无间隔 USB 下传），而不管该数据是否已经下传完成。注意应该及时查询防止缓冲区溢出。
调用一次 CH375ReadData 或 CH375WriteData 可以读写 0 字节到 4096 个字节，大于 4096 字节可以分

多个操作实现。分多次每次读写 64 个字节与一次读写 4096 字节的区别在于后者效率高，速度快。

```
BOOL    WINAPI  CH375ReadInter(  // 读取中断数据
    ULONG   iIndex,  // 指定 CH375 设备序号
    PVOID   oBuffer,  // 指向一个足够大的缓冲区,用于保存读取的中断数据
    PULONG  ioLength );  // 指向长度单元,输入时为准备读取的长度,返回后为实际读取的长度
```

从 CH375 端点 1（中断上传管道）的上传缓冲区中读出数据，调用前应该确定单片机已经或者即将用 WR-USB-DATA5 命令写入中断上传数据。如果单片机还没有写入中断上传数据则一直等待，如果单片机进入中断服务程序中没有释放缓冲区也会一直等待。一次读取的最大长度是 8 字节。

中断数据与批量数据的主要区别在于：中断数据实时性好（理论响应时间是不超过 1 毫秒）但是数据量小（8KB/S），而批量数据实时性差（计算机空闲时不超过 1 毫秒，计算机忙时可达几百毫秒）但是数据量大（1MB/S，理论值，实际速度要略小一些）。由于 Windows 是多任务操作系统，虽然中断数据对于 WDM 驱动程序可以获得 1 毫秒的响应时间，但是对于应用程序却达不到，因为 Windows 要进行任务调度和环境切换。

```
BOOL    WINAPI  CH375SetIntRoutine(  // 设定中断服务程序
    ULONG   iIndex,  // 指定 CH375 设备序号
    mPCH375_INT_ROUTINE iIntRoutine );  // 指定中断程序,为 NULL 则取消,否则在中断时调用
```

由于 USB 总线是主从结构，计算机应用程序很容易通知单片机，但是单片机却不容易通知计算机应用程序。除了查询方式之外，CH375 的 DLL 动态链接库还提供了中断方式，用于单片机在需要时及时通知计算机应用程序。方法是

应用程序事先用 CH375SetIntRoutine 指定一个中断服务程序 iIntRoutine（实际是一个子程序或者函数或者过程），当单片机用 WR-USB-DATA5 命令向 CH375 写入中断数据后，计算机端的 WDM 驱动程序可以在 1 毫秒之内收到该中断传输的数据，随后 WDM 通知 DLL，由 DLL 调用 iIntRoutine 中断服务程序，该程序可以分析中断传输的数据，并通知主程序进行相应的处理。

简单地说，单片机在需要时，可以用 WR-USB-DATA5 命令通知计算机应用程序指定的中断服务程序，并同时传递不超过 8 个字节的中断特征数据。

该中断服务实际上是由 DLL 另外创建一个线程，调用 CH375ReadInter 尝试从 CH375 的端点 1（中断上传管道）的上传缓冲区中读出中断特征数据，当读出数据后，再调用中断服务程序 iIntRoutine。

```
BOOL    WINAPI  CH375Abort????  // 取消未完成操作
```

如果应用程序已经打算退出，但是某个 USB 数据读写操作尚未完成（API 未返回），那么可以由该 API 取消对应的操作。共有 3 个 API，分别对应 3 个管道的取消操作。通常在另外一个线程中，或者在当前线程因为 USB 通讯超时返回时，才有机会取消未完成的操作，另外，如果 USB 通讯超时，并且不打算继续等待原操作完成，那么应该取消原操作。实际上，在用 CH375CloseDevice 关闭设备时，驱动程序通常会自动取消未完成的 USB 操作。

```
BOOL    WINAPI  CH375SetTimeout(  // 设置 USB 数据读写的超时
    ULONG   iIndex,  // 指定 CH375 设备序号
    ULONG   iWriteTimeout,  // 指定 USB 写出数据块的超时时间,以毫秒 mS 为单位
    ULONG   iReadTimeout );  // 指定 USB 读取数据块的超时时间,以毫秒 mS 为单位
```

为了避免在单片机未准备好的情况下读写 USB 数据时长时间等待，可以设置 USB 通讯超时，超时值以毫秒为单位，如果设置为 0xFFFFFFFF 则说明没有超时，也就是一直等待，默认值就是 0xFFFFFFFF，所以将一直等待到读写操作完成。

例如，CH375SetTimeout(0, 5000, 1000)设置写超时为 5 秒，设置读超时为 1 秒，当调用 CH375WriteData 下传数据时，如果单片机忙或者拒绝接收数据，那么该子程序将在 5 秒后超时而返回，当调用 CH375ReadData 上传数据时，如果单片机忙或者未准备好上传数据，那么该子程序将在 1 秒后超时而返回，从而避免长时间等待导致应用程序无法响应其它事件。注意，如果读写操作因为超时而返回，那么返回时的“实际读写长度”参数可能小于期望值，并且可能为 0。考虑到 WINDOWS 应用程序自身的响应时间较长，并且实际传输 4K 字节的数据至少需要 10 毫秒，所以超时时间通常不宜

小于 50 毫秒。

```
BOOL    WINAPI  CH375SetExclusive(  // 设置独占使用当前 CH375 设备
    ULONG   iIndex,  // 指定 CH375 设备序号
    ULONG   iExclusive );  // 为 0 则设备可以共享使用,非 0 则独占使用
```

如果不希望同一个 CH375 设备被多个应用程序打开并且使用，那么可以在打开 CH375 设备后，调用该 API 设置独占使用。如果需要设置中断服务程序，那么应该在该 API 之前执行，因为中断服务程序也需要再次打开 CH375 设备。

```
BOOL    WINAPI  CH375SetBufUpload(  // 设定内部缓冲上传模式
    ULONG   iIndex,  // 指定 CH375 设备序号,0 对应第一个设备
    ULONG   iEnableOrClear );  // 为 0 则“直接上传”,非 0 则“缓冲上传”并清除缓冲区
```

如果启用内部缓冲上传模式，那么 CH375 驱动程序创建线程自动接收 USB 上传数据到内部缓冲区，同时清除缓冲区中的已有数据，当应用程序调用 CH375ReadData 后将立即返回缓冲区中的已有数据。

```
LONG    WINAPI  CH375QueryBufUpload(  // 查询内部上传缓冲区中的已有数据包个数
    ULONG   iIndex );  // 指定 CH375 设备序号,0 对应第一个设备
```

该 API 仅用于“缓冲上传”模式，如果成功返回数据包个数，出错返回-1，不宜连续频繁查询。启用内部缓冲上传后，CH375 驱动程序自动接收 USB 上传数据到内部缓冲区，同时记录这个数据包的长度。内部缓冲区中可以保存多个数据包，每个数据包的最大长度是 64 字节，而包的实际长度是实际这个包的字节数。当应用程序调用 CH375ReadData 要求读数据块时，驱动程序总是按时间顺序返回最早接收到的多个数据包，直到满足应用程序所要求的字节数或者遇到一个长度不足 64 字节的数据包。

```
BOOL    WINAPI  CH375SetBufDownload(  // 设定内部缓冲下传模式
    ULONG   iIndex,  // 指定 CH375 设备序号,0 对应第一个设备
    ULONG   iEnableOrClear );  // 为 0 则“直接下传”,非 0 则“缓冲下传”并清除缓冲区
```

如果启用内部缓冲下传模式，那么 CH375 驱动程序创建线程自动发送内部缓冲区中的 USB 下传数据，同时清除缓冲区中的数据，当应用程序调用 CH375WriteData 时将直接放入缓冲区并立即返回。

```
LONG    WINAPI  CH375QueryBufDownload(  // 查询内部下传缓冲区中的剩余数据包个数
    ULONG   iIndex );  // 指定 CH375 设备序号,0 对应第一个设备
```

该 API 仅用于“缓冲下传”模式，如果成功返回数据包个数，出错返回-1，不宜连续频繁查询。启用内部缓冲下传后，当应用程序调用 CH375WriteData 要求写数据块时，驱动程序总是将大数据块分割为多个不超过 64 字节的基本 USB 包，记录每个数据包的长度，并按时间顺序放置到内部下传缓冲区中，然后由 CH375 驱动程序逐个自动发送内部缓冲区中的待下传 USB 数据包。内部缓冲区中可以暂存多个数据包，每个数据包的最大长度是 64 字节，而包的实际长度是实际这个包的字节数。

```
BOOL    WINAPI  CH375SetDeviceNotify(  // 设定设备事件通知程序
    ULONG   iIndex,  // 指定 CH375 设备序号,0 对应第一个设备
    PCHAR   iDeviceID,  // 可选参数,指向字符串,指定被监控的设备的 ID,字符串以\0 终止
    mPCH375_NOTIFY_ROUTINE  iNotifyRoutine );  // 指定设备事件回调程序,为 NULL 则取消事件
                                               通知,否则在检测到事件时调用该程序
```

USB 设备都支持动态插拔，即插即用，作为 USB 产品的应用程序有必要了解 USB 设备何时插入（连接）以及何时拔出（断开），从而避免对已断开的 USB 设备进行无效操作。CH375 的 DLL 动态链接库提供了一种方法，即设备插拔事件通知，能够让应用程序及时了解 USB 设备的插拔事件。

应用程序事先用 CH375SetDeviceNotify 指定一个事件回调程序 iNotifyRoutine（实际是一个子程序或者函数或者过程），当指定的 USB 设备插入或者拔出后，计算机端的 DLL 动态库可以在几十毫秒之内收到该事件通知，随后由 DLL 调用 iNotifyRoutine 事件回调程序，该程序可以分析设备事件和当前状态，并通知主程序进行相应的处理。如果是 USB 设备插入事件，那么可以通知主程序延后数百毫秒再打开该 USB 设备进行操作，如果是 USB 设备拔出事件，那么应该通知主程序及时停止 USB 传输并

关闭 USB 设备。具体实例可以参考 DEMO 演示程序。
当应用程序的主程序启动后，可以首先设置设备事件通知程序，然后使用循环计数逐个尝试打开 USB 设备，有则开始 USB 传输，无则等待 USB 设备插入事件。当收到插入事件后，也应该使用循环计数逐个尝试打开 USB 设备，无则继续等待。

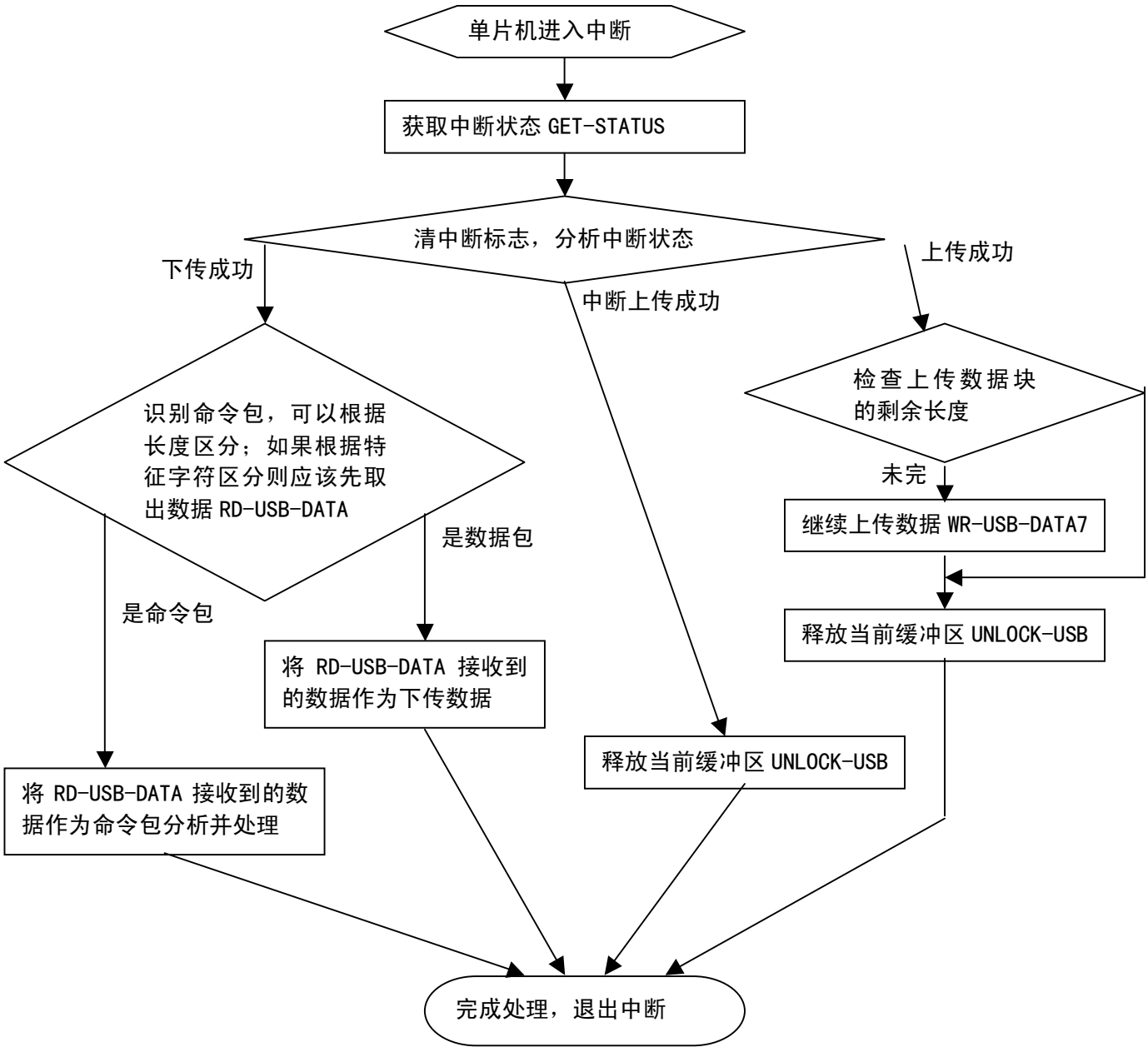
(2)、计算机作为传输节奏控制方

在计算机的应用程序中，总是先发送命令包，然后可选地进行数据传输。

如果是以下传长度区分数据包与命令包，那么应该检查数据总长度除以 64（单个数据包的最大长度）后的余，如果与命令包的长度相同，则应该调整长度，防止误认。

对于 CH375B 或者 CH372B 芯片，还可以通过辅助数据下传管道传送命令包，通过数据上传和下传管道传送数据，从而能够直接识别出命令包。

单片机在中断服务程序中处理 USB 传输，流程如下。



(3)、单片机作为传输节奏控制方

计算机应用程序在打开 CH375 设备后，调用 CH375SetIntRoutine 并指定一个中断数据接收子程序（或者 C 函数）IntRoutine，然后主程序被动地等待消息通知。

当单片机需要上传数据给计算机时，首先用 WR-USB-DATA7 将上传数据的前 64 字节写入 CH372，然后用 WR-USB-DATA5 写入不超过 8 个字节的中断特征数据。不超过 1 毫秒（理论值），计算机 CH375 的 WDM 驱动程序将收到这些中断特征数据，接着通知 DLL 动态链接库的线程，该线程直接调用应用程序指定的 IntRoutine 子程序并将中断特征数据作为其参数。该子程序分析中断特征数据，然后发出消息通知应用程序的主程序用 CH375ReadData 去读取上传数据。单片机可以在中断特征数据中注明本次上传数据的长度，例如是 4096 个字节，那么主程序在调用 CH375ReadData 时应该指明是要 4096 个字节，而单片机在收到上传成功中断后，应该继续用 WR-USB-DATA7 写入上传数据的第二组 64 字节，直到数据块结束。

当单片机需要计算机下传数据给单片机时，可以用 WR-USB-DATA5 写入不超过 8 个字节的中断特征数据。当应用程序的 IntRoutine 子程序分析中断特征数据时，可以区分出是要下传数据，所以发出消息通知应用程序的主程序用 CH375WriteData 下传数据。

如果不使用中断传输向计算机提供中断特征数据，还可以在设置较长的读超时后，直接尝试从单片机读取具有特殊长度的中断特征数据块，然后分析处理。

⑷、双向数据流

计算机通过 CH375WriteData 下传数据给单片机，通过 CH375ReadData 从单片机读取数据，或者由单片机提供中断后再读取。建议设置读写超时，便于取消操作和退出。

上传与下传分别属于物理上隔离的两个通道，两者之间不同步。

对于上传数据，建议通过 CH375SetBufUpload 选择内部“缓冲上传”模式，一方面提高上传速度，另一方面，由驱动程序提供线程处理上传数据比应用程序的线程效率高。如果数据量较小，那么缓冲上传会出现长期空载的情况而降低 USB 总线的使用效率。对于下传数据，也可以使用“缓冲下传”模式，缓冲下传不影响 USB 总线的使用效率。

⑸、参考资料及例子

①、 在 CHECK/DEMO 目录下是一个计算机作为传输节奏控制方的应用实例，命令包与数据包之间通过长度区分，用于演示 CH372 的数据传输和 CH452 显示及键盘功能。计算机应用程序在初始化后调用 CH375SetIntRoutine 指定了中断服务程序，以便在单片机检测到 CH452 的按键后及时通知计算机。计算机应用程序也可以在用户的操作下将 CH452 的命令下传给单片机。当单片机检测到 CH452 按键按下或者释放时都会以中断方式通知计算机应用程序。该程序同时演示 USB 设备插拔事件通知的用法，以及演示 CH452 和 CH451 的常用命令等。

②、 在 CHECK/BULK 目录下是另一个计算机作为传输节奏控制方的应用实例，所有下传数据包的首字节是命令码，没有专门的命令包，用于测试 CH375 或者 CH372 的数据传输速度。对于 24MHz 的标准 MCS-51 单片机，如果只传输而不处理数据，那么汇编程序下速度可达 260KB/S（C 语言 160KB/S），如果是在 WDM 驱动程序中则还要快些，如果单片机的时钟改为 12MHz 则速度降到 150KB/S。为了演示该测试程序，需要将目标程序 BULK\MCU\CH375BLK.HEX 通过编程器写入单片机，然后该测试程序就可以在本演示板上运行。在 WINDOWS 下，如果 CH375B 或者 CH372B 芯片以 0.6uS 为周期与单片机交换数据，实测速度最高约为 600KB/S。

③、 在 CHECK/BULK/WIN 目录中的 SPEED372.C 是“缓冲上传”模式的例子，用于测试 USB 传输速度，正常情况下，上传速度比“立即上传”模式提高 10%到 20%。“缓冲上传”模式通常用于双向数据流结构的应用程序。

④、 在 CHECK/TEST 目录中是一个很简单的测试程序，包括上位机程序和 MCS51 单片机程序，PC 机应用程序只要不按空格键就可以长时间测试，当使用 CH372 或者 CH375 设计 USB 产品时，可以用于测试 USB 数据传输的正确性、USB 硬件的长期稳定性。

⑤、 在 CHECK/VB 目录中为对应的 VB 语言的应用程序示例，在 VC 目录中为对应的 VC 语言的应用程序示例，在 DELPHI 目录中为对应的 DELPHI 语言的应用程序示例。

⑥、 在 MCU_IF 子目录中为常用的 C 和 ASM 常量定义，其中 C 子目录中 CH372FRM.C 为单片机使

用 CH372 或者 CH375 进行 USB 通讯的框架 C 源程序，该程序可以支持 USB 总线挂起状态的低功耗模式。在 CH452 或 CH451 的 MCU_IF 目录中有 CH452 或 CH451 芯片的接口子程序。

⑦、 在 CHECK/XFIRM 子目录中为 CH372/CH375 外置固件模式的 C 和 ASM 示例程序，可供参考设计 HID 类的 USB 设备以及其它标准 USB 类设备。

⑧、 在编写单片机程序的过程中，可以使用 WINDOWS 端的 DEBUG372 简单调试工具程序，支持 CH372 或 CH375 芯片和各种单片机，可以通过 USB 收发数据。注意，工具 DEBUG372 是多线程应用程序，该源程序不推荐参考，工具程序目录为 DEBUG/DEBUG372。

⑨、 对于 MCS51 单片机，可以直接内置 CH372 的 USB 调试固件程序，从而能够在 PC 机端用对应的工具程序简单控制和检查单片机的 SFR 和 RAM 等。进一步的应用是，在单片机与计算机程序之间约定将单片机中某块内存作为数据交换区，由于计算机端的程序可以任意读写 MCS51 单片机的内存，自然也就实现了单片机与计算机端两者之间的数据交换。网上提供相关资料供参考，文件目录为 CH372/DEBUG/MCS51。

6、其它注意事项

(1) 如果用频率计测量 CH372 的振荡频率要考虑探头电容对频率的影响，普通晶体都完全可以满足晶体 X1 的 1‰精度要求，所以不必测量频率。当 CH372 或者 CH375 正常工作时，XI 和 XO 引脚用数字万用表测量，其电压应该在 VCC/2±1V 左右，注意测量过程本身可能会导致停振，并且测量完毕必须关闭电源再重开。如果电源电压为 3.3V，建议将 XI 引脚的电容 C1 容量选用小些（例如 15pF）。如果 XO 端电压接近 VCC 而 XI 端电压低于 VCC/2，那么通常是晶体或电容对地漏电。

(2) 为了提高可靠性和稳定性，强烈建议参考 USB 电路设计及 PCB 设计注意事项 README.PDF 文档。为了降低电磁辐射，并减少干扰，晶体 X1 的金属外壳应该接地，晶体 X1 以及电容 C1、C2 应该尽量靠近 CH372（或者 CH375），相关的 PCB 走线应该尽量短，并且可以在周边环绕接地线或者敷铜。USB 数据线 D+和 D-应该平行布线，长度保持差不多，两侧可以环绕接地线或者敷铜。

(3) 由于电源 VCC 对地接了较大的电解电容 C5，所以当电源断开后，VCC 将在几十秒之内保持 0.6V～1.5V 的电压，当再次接通 5V 电源后，由于 VCC 不是从 0V 上升到 5V，所以有时 CH372 得不到可靠的上电复位而不能正常工作，尤其是间隔几秒连续插拔 USB 连接线时比较明显。解决方法有三种：

① 在 VCC 与 GND 之间接一个放电电阻，例如，CH372 演示板就在电容 C9 的位置加了一个阻值为 1KΩ的电阻，使电容 C5 能够及时的放电到 0V。

② 将 VCC 与 GND 之间的电解电容取小些，例如 10uF 以下，但是如果板上有 CH451 驱动数码管或者 LED 显示，那么电解电容也不能小于 100uF。

③ 使用 CH375 代替 CH372，由于 CH375 具有外部复位输入引脚 RSTI，如果在 RSTI 与 VCC 之间接电容 C14（容量为 0.01uF～1uF，图中为 0.47uF），可以使 CH375 可靠上电复位，或者由额外的μP 监控 IC 提供复位，可以与单片机使用同一个电源监控电路。演示板中的按键 K64 用于对 CH375 进行手工复位，与 CH372 无关。

(4) 如果单片机系统已经有电源供电（不依赖于计算机的 USB 供电），那么可以去掉电阻 R7，完全不使用 USB 电源。如果打算同时使用 USB 电源，那么 R7 可以减小两路电源供电时直接并联引起的短路电流，其阻值可以根据实际情况调整为 0Ω到 5Ω。特别要注意，CH372 必须与单片机使用同一组电源，绝对不应该分开供电，否则会导致 CH372 或者单片机损坏。

(5) 如果 CH451 的 RSTI 引脚连接到仪器的面板作为手工复位输入，那么必须在 RSTI 与 VCC 之间或者与 GND 之间接电容 C13（容量为 100pF～0.1uF），可以减少外部信号干扰引起的不正常复位。

(6) 计算机端的应用程序应该在确定单片机已经或者即将准备好上传数据（或中断数据）时，才调用 CH375ReadData（或 CH375ReadInter）读取。否则，如果单片机一直没有准备好数据时，应用程序的调用线程将一直挂起，API 调用将一直等待收到数据或者出现错误时才会返回，当然这个问题还可以通过设置 USB 通讯超时解决。另外，如果计算机应用程序打算读取 1024 个字节，而单片机没有那么多的数据，那么单片机可以将最后一个数据的长度设为 0 到 63 字节，而不是 64 字节，那么读操作将提前结束。例如，只上传 511 个字节或 513 个字节，而如果只打算上传 512 个字节（64 的倍数），那么应该在最后再上传 0 个字节。

(7) 如果调用读写 API 返回错误，那么有几种可能：

① 该 USB 设备可能尚未打开，或者已经关闭。
② 下位机的单片机程序发生错误，程序跑飞了或者程序设计不佳。
③ 该 USB 设备已经断开，或者计算机特别忙导致操作超时。
④ 可能是 USB 传输受到干扰，导致多次重试数据仍然有误。
⑤ CH372 或者 CH375 芯片受到干扰，尤其是要防止 XI 引脚的时钟受到干扰。
⑥ 对于读操作，可能是 CH375 上传的长度超过应用程序准备读取的长度，例如，调用 CH375ReadData 指定 Length 的值为 15（只打算读 15 个字节），而单片机却向 CH375 写入了 28 个字节的数据，那么读操作将失败。这种情况发生在单片机与计算机的命令失去同步，或者未能正确理解命令时。
⑦ 发生错误后，建议步骤：先关闭设备，然后重新打开。如果能够打开，说明设备仍然存在，可以继续读写，否则说明设备已经断开。

(8) 在单片机程序设计中，向 CH372 发出命令后应该至少延时 1.5 微秒再读写数据，连续读写数据之间至少要有 0.6 微秒以上的间隔时间。有些命令需要几微秒的执行时间，具体参数可以参考 CH372 芯片手册。在更换为高速单片机或者提高时钟速度时，要考虑该时序。

(9) CH372 可以自行定义 USB 的厂商 ID 和设备 ID，用于与其它 USB 设备区分开，如果重新设计驱动程序，则强烈建议另行定义 ID，避免与 CH372 的默认驱动程序发生冲突。注意，设置 USB 产品 ID 的 SET-USB-ID 命令必须在 SET-USB-MODE 命令之前执行。如果在修改上 USB 的 ID 后，仍然希望使用 CH372 的通用驱动程序，那么可以修改驱动程序安装信息文件 CH375WDM.INF，找到其中类似 USB\VID_????&PID_????的内容，另外增加一组类似内容，并将问号位置的 ID 替换为自行定义的 VID 和 PID。在 USB 产品设计初期，建议先用默认的 ID。