mciSendCommand

[求助编辑](http://baike.baidu.com/view/4189925.htm)**百科名片**

MCI(Media Control Interface，媒体控制接口)向Windows程序提供了在高层次上控制媒体设备接口的能力。程序不必关心具体设备，就可以对激光唱机（CD）、视盘机、波形音频设备、视频播放设备和MIDI设备等媒体设备进行控制。

目录

[mciSendCommand](http://baike.baidu.com/view/4189925.htm" \l "1)

[MCI简介](http://baike.baidu.com/view/4189925.htm#2)

[mciSendCommand详细](http://baike.baidu.com/view/4189925.htm#3)

[编辑本段](http://baike.baidu.com/view/4189925.htm)**mciSendCommand**

　　MCI的命令消息接口，所有MCI命令消息都是通过mciSendCommand函数发送的，函数声明为：

　　MCIERROR mciSendCommand(

　　MCIDEVICEID wIDDevice, //设备的ID，在打开设备时不用该参数

　　UINT　uMsg, //命令消息

　　DWORD fdwCommand, //命令消息的标志

　　DWORD dwParam //指向包含命令消息参数的结构

　　); //若成功则返回0，否则返回错误码

[编辑本段](http://baike.baidu.com/view/4189925.htm)**MCI简介**

　　对于程序员来说，可以把MCI理解为设备面板上的一排按键，通过选择不同的按键（发送不同的MCI命令）可以让设备完成各种功能，而不必关心设备内部实现。

　　比如，对于play，视盘机和CD机有不同的反应（一个是播放视频，一个播放音频），而对用户来说却只需要按同一按钮。

　　应用程序通过向MCI发送命令来控制媒体设备。MCI命令接口分命令字符串和命令消息两种，两者具有相同的功能。命令字符串具有使用简单的特点，但是它的执行效率不如命令消息。

[编辑本段](http://baike.baidu.com/view/4189925.htm)**mciSendCommand详细**

　　MCI的命令消息接口提供了C语言接口，它速度更快，并且更能符合C/C++程序员的需要。

　　所有MCI命令消息都是通过mciSendCommand函数发送的，函数声明为：

　　MCIERROR mciSendCommand(

　　MCIDEVICEID wIDDevice, //设备的ID，在打开设备时不用该参数

　　UINT　uMsg, //命令消息

　　DWORD fdwCommand, //命令消息的标志

　　DWORD dwParam //指向包含命令消息参数的结构

　　); //若成功则返回0，否则返回错误码

　　变量wDeviceID用来保存设备的ID，系统用ID来标识不同的设备，以保证命令发给正确的对象。

　　void CTttView::OnMyMenu()

　　{

　　MCI\_OPEN\_PARMS mciOpen;

　　UINT wDeviceID;

　　mciOpen.lpstrDeviceType = "avivideo";

　　mciOpen.lpstrElementName = "myfolder\\clock.avi";

　　mciSendCommand(0, MCI\_OPEN, MCI\_OPEN\_ELEMENT, (DWORD)&mciOpen);

　　wDeviceID=mciOpen.wDeviceID;

　　MCI\_PLAY\_PARMS mciPlay;

　　mciSendCommand(wDeviceID, MCI\_PLAY, MCI\_WAIT, (DWORD)&mciPlay);

　　);

　　---- MCI可以通过mciSendCommand（）和mciSendString（）来完成，本文仅使用mciSendCommand（）函数。

　　---- 原型：DWORD mciSendCommand（UINT wDeviceID，UINT wMessage，DWORD dwParam1，DWORD dwParam2）；

　　参数： wDeviceID：接受消息的设备ID

　　wMessage：MCI命令消息

　　dwParam1：命令的标志位

　　dwParam2：所使用参数块的指针

　　---- 返值：调用成功，返回零；否则，返回双字中的低字存放有错误信息。

　　二MIDI的播放控制

　　---- 1． 打开设备

　　MCI\_OPEN\_PARMS OpenParms;

　　OpenParms.lpstrDeviceType =

　　(LPCSTR) MCI\_DEVTYPE\_SEQUENCER;//MIDI类型

　　OpenParms.lpstrElementName = (LPCSTR) Filename;

　　OpenParms.wDeviceID = 0;

　　mciSendCommand (NULL, MCI\_OPEN,

　　MCI\_WAIT | MCI\_OPEN\_TYPE |

　　MCI\_OPEN\_TYPE\_ID | MCI\_OPEN\_ELEMENT,

　　(DWORD)(LPVOID) &OpenParms)

　　---- MCI设备ID指明打开了哪个设备，当发送了MCI\_OPEN命令时，这个值在参数块中返回——应被保存备用。

　　---- 2． 关闭设备

　　mciSendCommand (m\_wDeviceID, MCI\_CLOSE, NULL, NULL);

　　---- 3． 播放

　　MCI\_PLAY\_PARMS PlayParms;

　　PlayParms.dwFrom = 0;

　　// 指定从什么地方（时间）播放

　　mciSendCommand (m\_wDeviceID, MCI\_PLAY,

　　MCI\_FROM, (DWORD)(LPVOID)

　　&PlayParms))；

　　---- 4． 暂停

　　MCI\_PLAY\_PARMS PlayParms;

　　mciSendCommand (m\_wDeviceID, MCI\_PAUSE, 0,

　　(DWORD)(LPVOID) &PlayParms)；

　　---- 5． 停止

　　mciSendCommand (m\_wDeviceID, MCI\_STOP, NULL, NULL);

　　---- 6． 跳跃

　　\* 跳转到任意地方

　　MCI\_SEEK\_PARMS SeekParms;

　　SeekParms.dwTo = (nMinute \* 60 + nSecond) \* 1000;

　　//跳转的目标时间，时间单位为毫秒

　　mciSendCommand (m\_wDeviceID, MCI\_SEEK, MCI\_TO

　　| MCI\_WAIT,(DWORD)(LPVOID)

　　&SeekParms)；

　　\* 跳到文件头

　　mciSendCommand (m\_wDeviceID, MCI\_SEEK,

　　MCI\_SEEK\_TO\_START, NULL);

　　\* 跳到文件尾

　　mciSendCommand (m\_wDeviceID, MCI\_SEEK,

　　MCI\_SEEK\_TO\_END, NULL);

　　---- 7． 查询当前信息

　　MCI\_STATUS\_PARMS StatusParms;

　　StatusParms.dwItem = MCI\_SEQ\_STATUS\_DIVTYPE;

　　mciSendCommand (m\_wDeviceID, MCI\_STATUS,

　　MCI\_WAIT | MCI\_STATUS\_ITEM,

　　(DWORD)(LPVOID) &StatusParms)；

　　返回信息存放于StatusParms.dwReturn中。

　　MCI\_STATUS标志

　　MCI\_STATUS\_LENGTH 获得文件长度

　　MCI\_STATUS\_MODE 获得文件播放的当前状态

　　MCI\_STATUS\_POSITION 获得文件播放的当前位置

　　MCI\_STATUS\_TIME\_FORMAT 获得当前的时间格式

　　MCI\_SEQ\_STATUS\_DIVTYPE 判断文件是PPQN类型还是SMPTE类型

　　MCI\_SEQ\_STATUS\_TEMPO 获得当前播放速度，PQRN类型，

　　此值为节拍/分，SMPTE类型，此值为祯/秒

　　---- 8． 设置时间格式及播放速度

　　MCI\_SET\_PARMS SetParms;

　　SetParms.dwTimeFormat = MCI\_FORMAT\_MILLISECONDS;

　　//设置时间单位为毫秒

　　mciSendCommand (m\_wDeviceID,

　　MCI\_SET, MCI\_SET\_TIME\_FORMAT,

　　(DWORD)(LPVOID) &SetParms)；

　　MCI\_SEQ\_SET\_TEMPO 设置播放速度，

　　PQRN类型，此值为节拍/分，

　　SMPTE类型，此值为祯/秒

　　第二部分 WAV文件的播放

　　---- 一、原理

　　---- MicroSoft API提供了三种不同的方法来实现WAV的播放：

　　PlaySound（）函数。它可以通过单行编码来播放Wave格式的声音。此函数有两个限制：必须将声音数据完整地载入物理内存；数据格式必须被所配置的某一音频驱动器支 持。根据经验，PlaySound（）适用于100K以下的文件。

　　MCI（The Media Control Interface），与上一章播放MIDI文件相似，可以播放100K 以上的文件。

　　低级Wave音频设备。用这些设备可以运行完全控制Wave数据的应用文件。

　　---- 二、 WAV文件播放控制

　　---- 因为同样使用MCI，与上一章相同，只列出不同的部分。

　　---- 1． 打开设备

　　---- 将MIDI的MCI\_DEVTYPE\_SEQUENCER 改为"waveaudio"

　　---- 2． 录音

　　MCI\_RECORD\_PARMS RecordParms;

　　mciSendCommand (m\_wDeviceID, MCI\_RECORD,

　　NULL, (DWORD)(LPVOID)

　　&RecordParms)；

　　---- 3． 保存录音

　　MCI\_SAVE\_PARMS SaveParms;

　　SaveParms.lpfilename = (LPCSTR) Filename;

　　mciSendCommand (m\_wDeviceID, MCI\_SAVE,

　　MCI\_SAVE\_FILE | MCI\_WAIT,

　　(DWORD)(LPVOID) &SaveParms)；

　　第三部分 CD的播放

　　---- CD的独特优势在于，它由作曲家设计，并由音乐厂家生产。不同的计算机播放MIDI 文件时，声音效果也不一样，但是CD的声音效果总是相同的。高品质的音频对计算机用 户产生的效果会使你感到吃惊。 我们依然采用MCI播放CD，大部分的播放控制与前两部分相同，只列出不同的部分

　　---- 1． 开光驱门

　　mciSendCommand (m\_wDeviceID, MCI\_SET,

　　MCI\_SET\_DOOR\_OPEN, NULL);

　　---- 2． 关光驱门

　　mciSendCommand (m\_wDeviceID, MCI\_SET,

　　MCI\_SET\_DOOR\_CLOSED, NULL);

　　---- 3． 打开设备

　　将MIDI的MCI\_DEVTYPE\_SEQUENCER 改为MCI\_DEVTYPE\_CD\_AUDIO

　　---- 4． 播放

　　---- 指定播放起点必须经过MCI\_MAKE\_TMSF(Track，Minute，Second，Frame)转化

　　---- 5． 查询当前信息

　　MCI\_STATUS\_CURRENT\_TRACK 得到当前曲目

　　MCI\_STATUS\_LENGTH 得到CD或指定曲目长度

　　MCI\_STATUS\_MODE 得到驱动器的当前状态

　　MCI\_STATUS\_NUMBER\_OF\_TRACKS 得到CD曲目的数目

　　MCI\_STATUS\_POSITION 得到当前格式下的位置

　　MCI\_STATUS\_READY 检查设备是否就绪

　　MCI\_STATUS\_TIME\_FORMAT 得到当前时间格式

　　MCI\_STATUS\_MEDIA\_divSENT 检查以确认CD是否在驱动器内

　　MCI\_CDA\_STATUS\_TYPE\_TRACK 检查已确认某曲目是否为音频曲目

　　---- 注意：

　　使用MCI\_STATUS\_LENGTH参数查询CD 及曲目长度，返回值通过调用MCI\_MSF\_MINUTE（），MCI\_MSF\_SECOND（）转换为分、秒。

　　MCI\_STATUS\_POSITION参数返回值调用MCI\_TMSF\_TRACK（）, MCI\_TMSF\_MINUTE（）， MCI\_TMSF\_SECOND（），MCI\_TMSF\_FRAME才能得到当前的位置的道、分、秒、帧。

　　---- 6． 跳跃

　　---- 跳转的目标必须经过MCI\_MAKE\_TMSF(Track，Minute，Second，Frame)转化最好将上述三种格式分开建类，或做成动态连接库。在 Project-- >Setting-- >Link-- >Object/library modules中加入winmm.lib，源程序中包含。

　　---- MCI调用简单，功能强大，可以满足日常多媒体编程的基本需要。但是，MCI一次只能播放一个文件，使用DirectSound技术可以实现八个以上WAV文件的同时播放。

使用MCI API，源文件中需要包含头文件 Mmsystem.h，在Project->Settings->Link->Object/libray module中加入库 Winmm.lib。  
  
１、MCI简介  
  
　　MCI(Media Control Interface，媒体控制接口)向Windows程序提供了在高层次上控制媒体设备接口的能力。程序不必关心具体设备，就可以对激光唱机（CD）、视盘机、波形音频设备、视频播放设备和MIDI设备等媒体设备进行控制。  
　　对于程序员来说，可以把MCI理解为设备面板上的一排按键，通过选择不同的按键（发送不同的MCI命令）可以让设备完成各种功能，而不必关心设备内部实现。  
　　比如，对于play，视盘机和CD机有不同的反应（一个是播放视频，一个播放音频），而对用户来说却只需要按同一按钮。  
  
　　应用程序通过向MCI发送命令来控制媒体设备。MCI命令接口分命令字符串和命令消息两种，两者具有相同的功能。命令字符串具有使用简单的特点，但是它的执行效率不如命令消息。  
  
　　所有的MCI命令字符串都是通过多媒体API函数mciSendString传递给MCI的，该函数的声明为：  
　　　　MCIERROR mciSendString(  
　　　　　　LPCTSTR lpszCommand, 　　　//MCI命令字符串  
　　　　　　LPTSTR　lpszReturnString,　//存放反馈信息的缓冲区  
　　　　　　UINT　　cchReturn, 　　　　//缓冲区的长度  
　　　　　　HANDLE　hwndCallback 　　　//回调窗口的句柄，一般为NULL  
　　　　); //若成功则返回0，否则返回错误码。  
　　该函数返回的错误码可以用mciGetErrorString函数进行分析，该函数的声明为：  
　　　　BOOL mciGetErrorString(  
　　　　　　DWORD　fdwError,　　　//函数mciSendString返回的错误码  
　　　　　　LPTSTR lpszErrorText, //接收描述错误的字符串的缓冲区  
　　　　　　UINT　 cchErrorText　 //缓冲区的长度  
　　　　);  
  
　　下面是使用mciSendString函数的一个简单例子：  
　　　　char buf[50];  
　　　　MCIERROR mciError;  
　　　　mciError=mciSendString("open cdaudio",buf,strlen(buf),NULL);  
　　　　if(mciError)  
　　　　{  
　　　　　　mciGetErrorString(mciError,buf,strlen(buf));  
　　　　　　AfxMessageBox(buf);  
　　　　　　return;  
　　　　}  
　　open cdaudio命令打开CD播放器，如果出错（如驱动器内没有CD）则返回错误码，此时可以用mciGetErrorString函数取得错误信息字符串。  
  
２、MCI设备  
  
　　open是MCI打开设备的命令，cdaudio是MCI设备名。MCI的设备类型如下：  
　　　　animation　　动画设备  
　　　　cdaudio　　　CD播放器  
　　　　dat　　　　　数字音频磁带机  
　　　　digitalvideo 某一窗口中的数字视频（不基于GDI）  
　　　　other　　　　未定义的MCI设备  
　　　　overlay　　　重叠设备（窗口中的模拟视频）  
　　　　scanner　　　图象扫描仪  
　　　　sequencer　　MIDI序列器  
　　　　videodisc　　视盘机  
　　　　waveaudio　　播放数字波形文件的音频设备  
  
　　设备名是在注册表或SYSTEM.INI的[mci]部分定义的，典型的[mci]段如下：  
　　　　[mci]  
　　　　cdaudio=mcicda.drv  
　　　　sequencer=mciseq.drv  
　　　　waveaudio=mciwave.drv  
　　　　avivideo=mciavi.drv  
　　　　videodisc=mcipionr.drv  
　　等号的左边是设备名，右边是对应的MCI驱动程序。当安装了新的MCI驱动程序时，系统要用不同的设备名来区分。  
  
３、MCI命令  
  
　　使用MCI设备一般包括打开、使用和关闭三个过程，常用的MCI命令有：  
　　　　open　　　　打开设备  
　　　　close　　　 关闭设备  
　　　　play　　　　开始设备播放  
　　　　stop　　　　停止设备的播放或记录  
　　　　record　　　开始记录  
　　　　save　　　　保存设备内容  
　　　　pause　　　 暂停设备的播放或记录  
　　　　resume　　　恢复暂停播放或记录的设备  
　　　　seek　　　　改变媒体的当前位置  
　　　　capacility　查询设备能力  
　　　　info　　　　查询设备的信息  
　　　　status　　　查询设备状态信息  
　　MCI的大部分命令可以控制不同的媒体设备，但其中record和save命令并不是所有MCI设备都可以使用。  
　　MCI命令的使用是很随意的，只要先打开，最后关闭，中间可以随意调用各种命令。  
  
(1) open　打开设备  
  
　　MCI设备使用前必须先打开，当然，使用后也必须要关闭，以免影响他人的使用。  
  
　　open device\_name type device\_type alias device\_alias    
　　　　device\_name　 　　　要使用的设备名，通常是文件名。  
　　　　type device\_type　　设备类型，例如waveaudio或sequencer，可省略。  
　　　　alias device\_alias　设备别名，指定后可在其他命令中代替设备名。  
  
(2) play　开始设备播放  
  
　　MCI设备打开后即可以播放，可使用设备名或别名。  
  
　　play device\_alias from pos1 to pos2 wait repeat  
　　　　若省略from则从当前磁道开始播放，若省略to则播放到结束。  
　　　　若指明wait则等到播放完毕命令才返回。  
　　　　若指明repeat则会不停的重复播放。  
　　　　若同时指明wait和repeat则命令不会返回，本线程产生堵塞，通常会引起程序失去响应。  
  
(3) 播放CD  
  
　　void CTttView::OnOpenCD()  
　　{  
　　　　mciSendString("open cdaudio",NULL,0,NULL);  
　　　　mciSendString("play cdaudio",NULL,0,NULL);  
　　);  
  
　　void CTttView::OnStopCD()  
　　{  
　　　　mciSendString("stop cdaudio",NULL,0,NULL);  
　　　　mciSendString("close cdaudio",NULL,0,NULL);  
　　);  
  
　　还可以：  
　　　　pause cdaudio 　暂停播放。  
　　　　resume cdaudio　继续被暂停的播放。  
　　　　seek cdaudio to <位置>　移动到指定磁道。  
　　　　set cdaudio door open/closed　弹出或缩进CD盘。  
  
(4) 播放多媒体文件  
  
　　void CTttView::OnMyMenu()  
　　{  
　　　　mciSendString("open myfolder\\tada.wav alias aa",NULL,0,NULL);  
　　或　mciSendString("open myfolder\\flourish.mid alias aa",NULL,0,NULL);  
　　或　mciSendString("open myfolder\\clock.avi alias aa",NULL,0,NULL);  
　　　　mciSendString("play aa wait",NULL,0,NULL);  
　　　　mciSendString("close aa",NULL,0,m\_hWnd);  
　　);  
  
(5) 录制声音  
  
　　void CTttView::OnStartRecord()  
　　{  
　　　　mciSendString("open new type waveaudio alias aa",NULL,0,NULL);  
　　　　mciSendString("record aa",NULL,0,NULL);  
　　);  
  
　　void CTttView::OnStopRecord()  
　　{  
　　　　mciSendString("save aa c:\\aaa.wav wait",NULL,0,NULL);  
　　　　mciSendString("close aa",NULL,0,NULL);  
　　);  
  
４、MCI命令消息  
  
　　到目前为止，我们使用的都是MCI命令字符串。可以发现，命令字符串具有简单易学的优点，但这种接口与C/C++的风格相去甚远，如果程序要查询和设置大量数据，那么用字符串的形式将很不方便。  
　　MCI的命令消息接口提供了C语言接口，它速度更快，并且更能符合C/C++程序员的需要。  
  
　　所有MCI命令消息都是通过mciSendCommand函数发送的，函数声明为：  
　　　　MCIERROR mciSendCommand(  
　　　　　　MCIDEVICEID wIDDevice, 　//设备的ID，在打开设备时不用该参数  
　　　　　　UINT　　　　uMsg, 　　　//命令消息  
　　　　　　DWORD 　　　fdwCommand, //命令消息的标志  
　　　　　　DWORD 　　　dwParam 　　//指向包含命令消息参数的结构  
　　　　); //若成功则返回0，否则返回错误码  
  
　　命令消息uMsg与命令字符串是对应的，例如，open与MCI\_OPEN完成的是一样的功能。  
　　变量wDeviceID用来保存设备的ID，系统用ID来标识不同的设备，以保证命令发给正确的对象。  
  
　　void CTttView::OnMyMenu()  
　　{  
　　　　MCI\_OPEN\_PARMS mciOpen;  
　　　　UINT wDeviceID;  
　　　　mciOpen.lpstrDeviceType = "avivideo";  
　　　　mciOpen.lpstrElementName = "myfolder\\clock.avi";  
　　　　mciSendCommand(0, MCI\_OPEN, MCI\_OPEN\_ELEMENT, (DWORD)&mciOpen);  
　　　　wDeviceID=mciOpen.wDeviceID;  
　　　　MCI\_PLAY\_PARMS mciPlay;  
　　　　mciSendCommand(wDeviceID, MCI\_PLAY, MCI\_WAIT, (DWORD)&mciPlay);  
　　);  
  
　　可以看出，用命令消息比用命令字符串要复杂的多，但它的执行效率高。

重要的函数：  
==================

MCIERROR mciSendCommand(MCIDEVICEID wDeviceID, UINT uMsg, DWord dwFlags, DWORD dwParam);

第一个参数指定了设备标识，这个标识会在[程序员](http://wenwen.soso.com/z/Search.e?sp=S%E7%A8%8B%E5%BA%8F%E5%91%98&ch=w.search.yjjlink&cid=w.search.yjjlink)打开MCI设备时由系统提供。第二个参数指定将如何控制设备，见后面MCI指令清单，其中有有关录音的指令。第三个参数为访问标识，第四个参数一般是一个[数据结构](http://wenwen.soso.com/z/Search.e?sp=S%E6%95%B0%E6%8D%AE%E7%BB%93%E6%9E%84&ch=w.search.yjjlink&cid=w.search.yjjlink)，标识程序在访问MCI时要的一些信息。

MCI的设备类型：  
设备描述 描述字符串 说明  
MCI\_ALL\_DEVICE\_ID 所有设备  
MCI\_DEVTYPE\_ANIMATION Animation 动画设备  
MCI\_DEVTYPE\_CD\_AUDIO Cdaudio CD[音频](http://wenwen.soso.com/z/Search.e?sp=S%E9%9F%B3%E9%A2%91&ch=w.search.yjjlink&cid=w.search.yjjlink)  
MCI\_DEVTYPE\_DAT Dat [数字音频](http://wenwen.soso.com/z/Search.e?sp=S%E6%95%B0%E5%AD%97%E9%9F%B3%E9%A2%91&ch=w.search.yjjlink&cid=w.search.yjjlink)  
MCI\_DEVTYPE\_DIGITAL\_VIDEO Digitalvideo [数字视频](http://wenwen.soso.com/z/Search.e?sp=S%E6%95%B0%E5%AD%97%E8%A7%86%E9%A2%91&ch=w.search.yjjlink&cid=w.search.yjjlink)  
MCI\_DEVTYPE\_OTHER Other 未定义设备  
MCI\_DEVTYPE\_OVERLAY Overlay 重叠视频  
MCI\_DEVTYPE\_SCANNER Scanner [扫描仪](http://wenwen.soso.com/z/Search.e?sp=S%E6%89%AB%E6%8F%8F%E4%BB%AA&ch=w.search.yjjlink&cid=w.search.yjjlink)  
MCI\_DEVTYPE\_SEQUENCER Sequencer MIDI 序列器  
MCI\_DEVTYPE\_VCR Vcr 合式[录像机](http://wenwen.soso.com/z/Search.e?sp=S%E5%BD%95%E5%83%8F%E6%9C%BA&ch=w.search.yjjlink&cid=w.search.yjjlink)  
MCI\_DEVTYPE\_VIDEODISC Videodisc [激光](http://wenwen.soso.com/z/Search.e?sp=S%E6%BF%80%E5%85%89&ch=w.search.yjjlink&cid=w.search.yjjlink)[视盘](http://wenwen.soso.com/z/Search.e?sp=S%E8%A7%86%E7%9B%98&ch=w.search.yjjlink&cid=w.search.yjjlink)  
MCI\_DEVTYPE\_WAVEFORM\_AUDIO waveaudio Wave 音频

MCI指令   
MCI\_BREAK设置中断键，缺省是”CTRL+BREAK"  
MCI\_CAPTURE抓取当前帧并存入指定文件，仅用于数字视频  
MCI\_CLOSE关闭设备  
MCI\_CONFIGURE弹出配置对话框，仅用于数字视频  
MCI\_COPY拷贝数据至[剪贴板](http://wenwen.soso.com/z/Search.e?sp=S%E5%89%AA%E8%B4%B4%E6%9D%BF&ch=w.search.yjjlink&cid=w.search.yjjlink)  
MCI\_CUE延时播放或录音  
MCI\_CUT删除数据  
MCI\_DELETE删除数据  
MCI\_ESCAPE仅用于激光视频  
MCI\_FREEZE将显示定格  
MCI\_GETDEVCAPS获取设备信息  
MCI\_INDEX当前屏幕显示与否，仅用于VCR设备  
MCI\_INFO获取字符串信息  
MCI\_LIST获取[输入设备](http://wenwen.soso.com/z/Search.e?sp=S%E8%BE%93%E5%85%A5%E8%AE%BE%E5%A4%87&ch=w.search.yjjlink&cid=w.search.yjjlink)数量，支持数字视频和VCR设备  
MCI\_LOAD装入一个文件  
MCI\_MARK取消或做一个记号，与MCI\_SEEK配套  
MCI\_MARK取消或做一个记号，与MCI\_SEEK配套  
MCI\_MONITOR为数字视频指定报告设备  
MCI\_OPEN打开设备  
MCI\_PASTE粘帖数据  
MCI\_PAUSE暂停当前动作  
MCI\_PLAY播放  
MCI\_PUT设置源、目的和边框矩形  
MCI\_QUALITY定义设备缺省质量  
MCI\_RECORD开始录制  
MCI\_RESERVE分配硬盘空间  
MCI\_RESTORE拷贝一个bmp文件至帧缓冲  
MCI\_RESUME使一个暂停设备重新启动  
MCI\_SAVE保存数据  
MCI\_SEEK更改媒体位置  
MCI\_SET设置设备信息  
MCI\_SETAUDIO设置音量  
MCI\_SETTIMECODE启用或取消VCR设备的时间码  
MCI\_SETTUNER设置VCR设备频道  
MCI\_SETVIDEO设置video参数  
MCI\_SIGNAL在工作区上设置指定空间  
MCI\_STATUS获取设备信息  
MCI\_STEP使播放设备跳帧  
MCI\_STOP停止播放  
MCI\_SYSINFO返回MCI设备信息  
MCI\_UNDO取消操作  
MCI\_UNFREEZE使使用MCI\_UNFREEZE的视频[缓冲区](http://wenwen.soso.com/z/Search.e?sp=S%E7%BC%93%E5%86%B2%E5%8C%BA&ch=w.search.yjjlink&cid=w.search.yjjlink)[恢复运动](http://wenwen.soso.com/z/Search.e?sp=S%E6%81%A2%E5%A4%8D%E8%BF%90%E5%8A%A8&ch=w.search.yjjlink&cid=w.search.yjjlink)  
MCI\_UPDATE更新显示区域  
MCI\_WHERE获取设备裁减矩形  
MCI\_WINDOW指定图形设备窗口和窗口特性

其中比较常用的指令有MCI\_OPEN、MCI\_RECORD、MCI\_CLOSE、MCI\_PLAY、MCI\_STOP、MCI\_PAUSE、MCI\_STATUS等等。

举例：  
//关闭一个MCI设备  
mciSendCommand(DeviceID, MCI\_CLOSE, NULL, NULL);

//打开CD[播放器](http://wenwen.soso.com/z/Search.e?sp=S%E6%92%AD%E6%94%BE%E5%99%A8&ch=w.search.yjjlink&cid=w.search.yjjlink)  
MCI\_OPEN\_PARMS mciOpen;  
mciOpen.lpstrDeviceType = (LPSTR)MCI\_DEVTYPE\_CD\_AUDIO ;  
mciSendCommand(NULL, MCI\_OPEN, MCI\_OPEN\_TYPE | MCI\_OPEN\_TYPE\_ID, (DWORD)&mciOpen);

其他：

=======

高层音频接口:播放声音资源/文件

调用MCI::Media Control Interface 和Api::MessageBeep,sndPlaySound,PlaySound

底层音频接口:[数据采集](http://wenwen.soso.com/z/Search.e?sp=S%E6%95%B0%E6%8D%AE%E9%87%87%E9%9B%86&ch=w.search.yjjlink&cid=w.search.yjjlink)

开发一般步骤:

检测[音频设备](http://wenwen.soso.com/z/Search.e?sp=S%E9%9F%B3%E9%A2%91%E8%AE%BE%E5%A4%87&ch=w.search.yjjlink&cid=w.search.yjjlink)->打开[驱动程序](http://wenwen.soso.com/z/Search.e?sp=S%E9%A9%B1%E5%8A%A8%E7%A8%8B%E5%BA%8F&ch=w.search.yjjlink&cid=w.search.yjjlink)->分配缓冲区->管理缓冲区->操作音频数据->释放缓冲区->关闭驱动程序.

SDK参考::ms-help://MS.VSCC.v80/MS.MSDN.v80/MS.WIN32COM.v10.en/multimed/htm/\_win32\_multimedia\_functions.htm

多媒体控制两种方式可选:调API,[执行命令](http://wenwen.soso.com/z/Search.e?sp=S%E6%89%A7%E8%A1%8C%E5%91%BD%E4%BB%A4&ch=w.search.yjjlink&cid=w.search.yjjlink).

MCI命令结构:

[系统命令](http://wenwen.soso.com/z/Search.e?sp=S%E7%B3%BB%E7%BB%9F%E5%91%BD%E4%BB%A4&ch=w.search.yjjlink&cid=w.search.yjjlink):直接由MCI解析,并由系统处理,与设备无关

通用命令:由设备驱动处理

可选命令:

扩展命令:

常用MCI命令:

MCI\_OPEN,MCI\_PLAY,MCI\_RECORD,MCI\_STATUS,MCI\_SET,时间操作宏命令.

MIDI播放：  
  
首先在工程头文件中加入：   
#include "mmsystem.h"  
#pragma comment (lib,"WINMM.LIB")   
  
然后在程序初始化部分中加入：   
UINT wMIDIDeviceID;  
MCI\_OPEN\_PARMS mciOpenParams;  
MCI\_PLAY\_PARMS mciPlayParams;  
DWORD dwError;  
  
mciOpenParams.lpstrDeviceType = "sequencer";  
mciOpenParams.lpstrElementName = "main.mid"; //要播放的MIDI  
if(dwError=mciSendCommand(0,MCI\_OPEN,MCI\_OPEN\_ELEMENT|MCI\_OPEN\_TYPE,(DWORD)(LPVOID)&mciOpenParams))  
    ; // 播放  
else  
    wMIDIDeviceID=mciOpenParams.wDeviceID;  
  
if(dwError=mciSendCommand(wMIDIDeviceID,MCI\_PLAY,MCI\_NOTIFY,(DWORD)(LPVOID)&mciPlayParams))  
{  
    mciSendCommand(wMIDIDeviceID,MCI\_CLOSE,0,NULL); //关闭  
}  
  
WAVE播放：  
  
首先在工程头文件中加入：   
#include "mmsystem.h"  
#pragma comment (lib,"WINMM.LIB")   
  
然后在需要播放的地方加入：   
sndPlaySound("sound\_file.name",SND\_ASYNC|SND\_LOOP);  
  
SND\_ASYNC: 在背景播放  
  
SND\_LOOP: 连续播放