OS/2 Warp 多媒体子系统编程指南

[美] IBM 公司 著

刘晓玲 黄 俊 陈 雄
黄永刚 向红军 译

彭 丰 审校

清华大学出版社

目　录

第1章 多媒体子系统概论

OS/2 多媒体(指早先发行的多媒体显示管理器/2 或 MMPM/2)拥有可扩展结构,以便随着多媒体技术的发展,加入新的功能、设备及多媒体数据格式等。本章综述 OS/2 多媒体系统中的各个系统部件。随后的各章节将详细指导读者通过图 1-1 所示的样例程序安装和开发个人的 OS/2 多媒体子系统。根据你对多媒体的需求,可以方便地对样例程序模板进行修改,也可以用来为 OS/2 环境安装和开发多媒体控制驱动程序、流处理器及 I/O 过程。

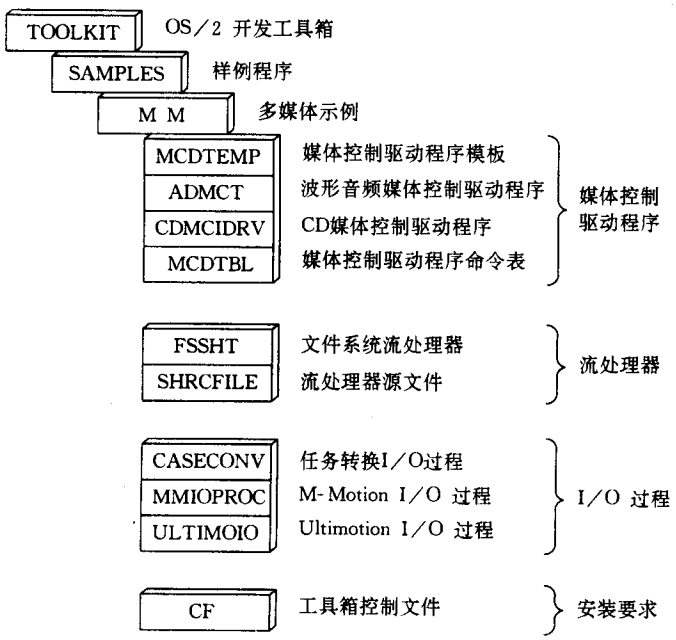

图 1-1 样例程序子目录结构

1.1 OS/2 多媒体系统结构

图 1-2 展示了 OS/2 多媒体系统各子系统部件。在 OS/2 环境下,这些子系统包括多媒体控制驱动程序、流处理器及 I/O 过程,并且均由程序管理器控制、监视其一系列工作。

在第三层环上,OS/2 多媒体使用了多媒体设备管理器 MDM,用于管理逻辑多媒体设备,例如声卡,CD-ROM 驱动器等其它硬件设备。在 MDM 的所有工作中,有一项就是:当有两个或两个以上应用程序申请控制同一台多媒体设备时,由它决定选择哪个应用

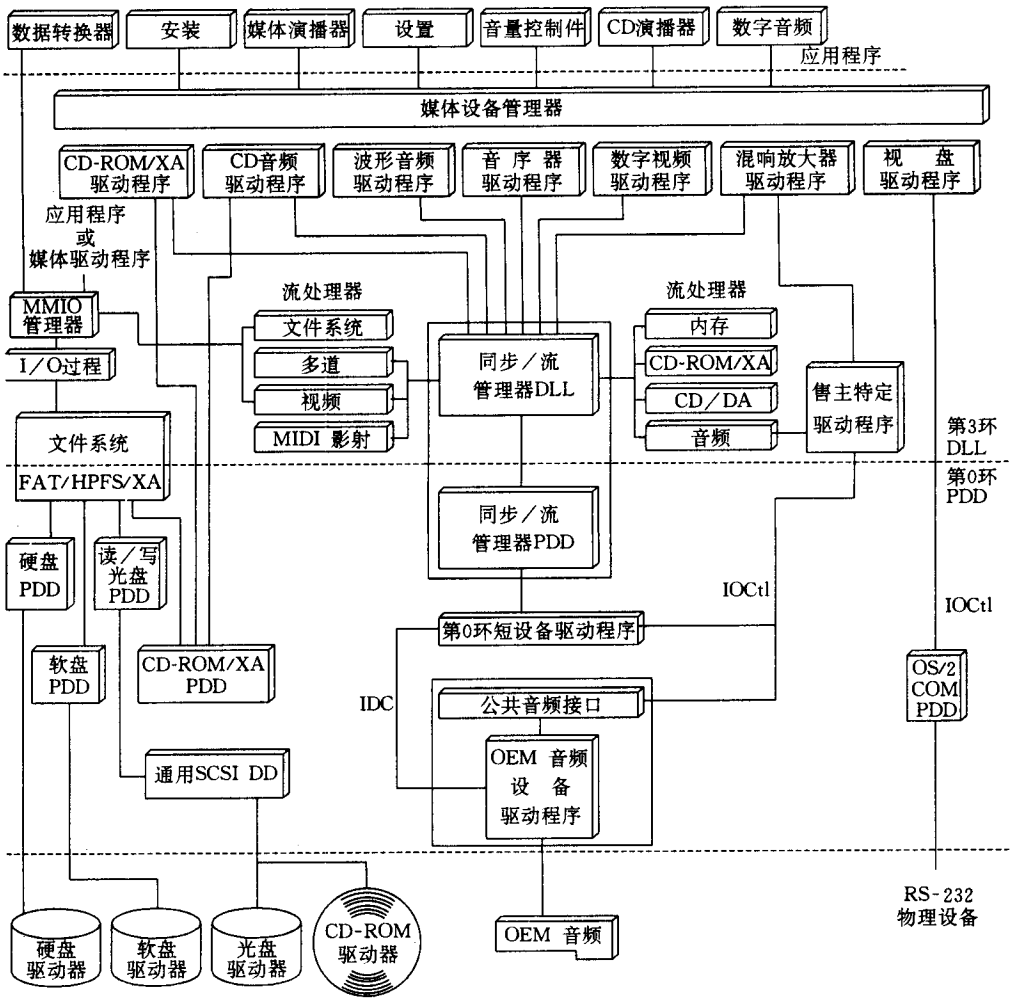

图 1-2　多媒体显示管理器/2 系统结构

程序取得控制权。

同步/流管理器(SSM),同样可以在多媒体控制驱动程序初始化后,用来管理流和同步调用。这样,就可以取消每个多媒体驱动程序对公用的多媒体设备的需要,而由本身找到解决办法。多组流处理器完成从源设备到目的设备的数据传输,而由 SSM 统一协调,并对数据缓冲区和同步数据进行集中管理。

最终由 MMIO 管理器完成诸如多媒体控制驱动程序、应用程序等子系统部件对各种数据的接收和操作,这些数据有图象、图表、数字音频信号、数字视频信号等等。通过不同的存储系统,不同数据以不同的文件格式存储,由 MMIO 管理器通过可安装的 I/O 过程来完成与读写操作有关的、对不同存储类型和文件表格进行的输入输出操作。

1.2 可扩展设备支持

OS/2 多媒体系统结构可以扩展。模块化结构设计允许对新发展的硬盘设备、逻辑设备、文件格式等的支持。

多媒体控制接口设备的示例参见表 1-1。此表提供的是系统能支持的、并且已被多媒体控制接口定义过的逻辑设备类型。打勾表示的是 OS/2 直接支持的设备。

表 1-1 多媒体控制界面逻辑设备类型

多媒体设备类型	OS/2 多媒体	字符串	通用文件名
混响放大器	√	ampmix	MCI_DEVTYPE_AUDIO_AMPMIX
磁带唱机		audiotape	MCI_DEVTYPE_AUDIO_TAPE
CD 音频演播器	√	cdaudio	MCI_DEVTYPE_CD_AUDIO
CD-XA 演播器	√	cdxa	MCI_DEVTYPE_CDXA
数字式录音机		dat	MCI_DEVTYPE_DAT
数字式录象机	√	digitalvideo	MCI_DEVTYPE_DIGITAL_VIDEO
耳机		headphone	MCI_DEVTYPE_HEADPHONE
麦克风		microphone	MCI_DEVTYPE_MICROPHONE
监视器		monitor	MCI_DEVTYPE_MONITOR
其它		other	MCI_DEVTYPE_OTHER
视频覆盖		videooverlay	MCI_DEVTYPE_OVERLAY
音序器管理器	√	sequencer	MCI_DEVTYPE_SEQUENCER
扬声器		speaker	MCI_DEVTYPE_SPEAKER
视盘演播器	√	videodisc	MCI_DEVTYPE_VIDEODISC
录象机		videotape	MCI_DEVTYPE_VIDEOTAPE
波形音频演播器	√	waveaudio	MCI_DEVTYPE_WAVEFORM_AUDIO

注意:M-Control Program Version 2.01 支持 M-Motion Video Adapter/A,并提供OS/2 多媒体覆盖技术。

1.3 多媒体控制驱动程序

多媒体控制接口提供应用程序控制多媒体设备的主要结构。最顶层包含 MDM,由它对多媒体设备进行最初管理;最底层包含各种多媒体驱动程序(MCD)——动态链接库实现对设备的控制。如图 1-3 所示。

应用程序与多媒体控制接口,继之与多媒体设备之间的相互作用有两种方式:控制程序接口(mciSendCommand)和字符串接口(mciSendString)。但是,在 MCD 解释字符命令之前,必须通过查询命令表来完成从字符命令到相应控制程序命令之间的转换。

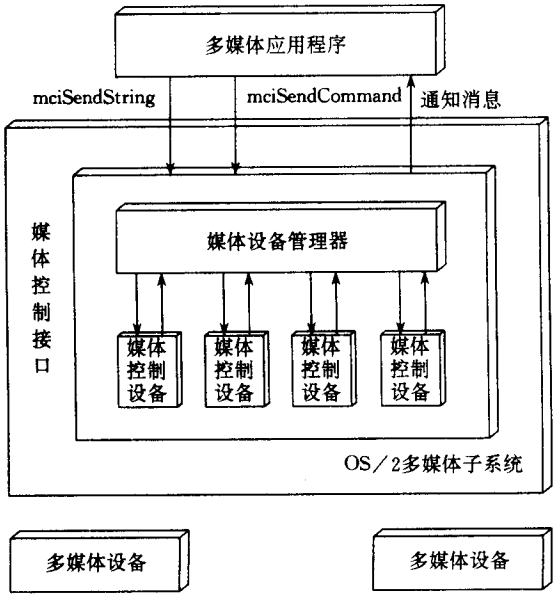

图 1-3　多媒体控制接口结构

MCD 并不直接控制硬件设备，而是通过命令在子系统或物理设备驱动程序接口的传输来实现。这种设计将 MCD 从必须确认硬盘以完成其功能中解脱出来。不过，MCD 不得不熟悉完成那种功能的方法。举例说，一台 CD-Audio 播放器通过使用 CD-ROM 驱动器及其内部的数-模转换器(DAC)，靠简单地分配设备 IOCtl 指令到驱动设备程序中，就能完成。只不过与前面不同的是，一台 CD-Audio 播放器采用独立的数字信号处理器(DSP)记录数字音频数据，采用功能调用实现对 DSP 协处理器的管理及数据在驱动器和协处理器的传输管理。

MCD 也可以使用其它多媒体子系统提供的服务，例如流编程接口(SPI)等。这种系统提供数据流服务，即就是允许流处理器控制设备之间的实时数据流，维持物理设备之间的数据连续流动。

1.4　I/O 控制程序过程

多媒体输入输出(MMIO)服务具有 I/O 和 CODEC 两种控制程序。I/O 控制程序是基于信号的处理程序，直接控制在不同的存储系统和文件表之间与读写操作有关的输入输出操作。通过 MMIO 信号将应用程序和 MMIO 子系统联系起来。当 MMIO 接收到来自应用程序的功能调用指令时，MMIO 管理器就送出与指令相适应的预先定义好的信息给 I/O 过程，负责对特定的文件表或存储系统进行操作。同样，I/O 过程按照 MMIO 管理器或应用程序发出的指令进行操作。

上述信息的设计实现所有 I/O 过程之间的相互有效联接。当然，I/O 过程也必须能处理信息或能传送信息给其子程序。例如，I/O 过程接收到要求数据压缩的信息，则必须

能够处理这条信息或将其传到 CODEC(编码/译码)程序。图 1-4 所示的就是 MMIO 子系统中 I/O 和 CODEC 程序之间的相互关系。

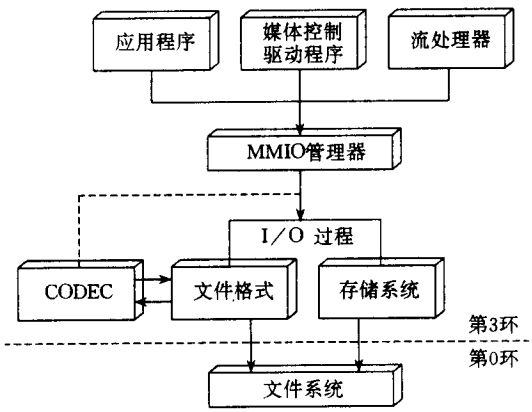

图 1-4　多媒体 I/O(MMIO)子系统结构

MMIO 管理器调用的子程序有：

1. 文件表

文件表子程序是用来对数据进行操作的 I/O 程序，其每一子程序处理不同类型元素，如音频、图象、MIDI 等，并且能够单独处理数据而不依赖其它子程序。但是，文件表子程序可能需要调用存储系统 I/O 过程，以便从含多个元素的文件中取得数据。

2. 存储系统

存储系统子程序是一种相当于为文件表程序解开数据并取出来的 I/O 过程。在上述过程中，存储系统将忽略数据表的格式。

3. 编码/译码子程序(CODEC)

CODEC 子程序对文件或缓冲区数据进行操作。根据数据的内容，I/O 过程可以装入 CODEC 子程序用于压缩或解压缩数据。

1.5　流 处 理 器

多媒体系统可以在系统内核级(第 0 环)或应用程序级(第 3 环)提供流处理器。流处理器安排在这两级是因为许多流理论上在与设备的物理驱动程序(PDD)直接联系时，要受到控制；其它的流则不与源数据或目标数据联系，而物理上直接与特定设备联系。例如，文件系统流处理器是 DLL，由于所有文件系统的 I/O 功能可在第 3 环 OS/2 功能中得到，并为所有文件系统设备服务，这就不用为每一个文件系统进入而建立特定的流处理设备驱动程序。

流处理器负责对应用数据的连续、实时方式的流动实现控制。每一处理程序可以建立多个数据流层，其中每个流包含特定类型的数据，如 MIDI(音响设备数字接口)或 ADPCM(自适应增量脉冲编码调制)等。应用程序通过使用媒体控制驱动程序，调用 SPI 功

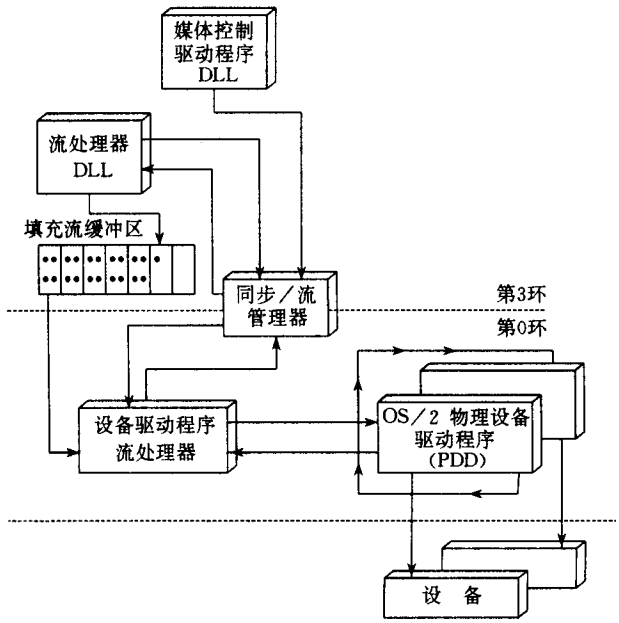

图 1-5 同步/流子系统结构

能来产生流或调用其它 SPI 功能来激活数据流，而不必一直调用 SPI 功能来维持数据流。相反，流处理器能保持 I/O 连续性，并简化应用程序的操作。

第2章 媒体控制驱动程序

媒体控制驱动程序(Media Control Driver 缩写为 MCD)。本章描述了各种流式和非流式的 MCD 的编程接口,并借此来告诉你如何编写 MCD。下面将把讨论的重点放在样例程序二重唱演奏器(Duet Player)中所用的 MCD(详见《OS/2 多媒体应用程序编程指南》一书)。波形音频 MCD 通过 SPI 数据流方式来创建和管理源数据流处理器和目标数据流处理器。而 CD 音频 MCD 则不然,它们无需相关的数据流,因为大多数的 CD 音频设备的数据处理是内部完成的(即相对于本设备是内部的,无需求助于主 CPU)。

在 \ TOOLKIT \ SAMPLES \ MM 子目录中,可以找到下列媒体控制驱动程序样例的源代码。

1. 媒体控制驱动程序模板(MCDTEMP)

提供了一个编写 MCD 的基本模板。而对于具体的流式或 MMIO 的样例,参阅子目录 ADMCT 和 CDMCIDRV。

2. 波形音频媒体控制驱动程序(ADMCT)

提供了一个关于怎样控制流式设备的例子。流式设备使用 OS/2 多媒体的同步/流管理器(Sync/Stream Manager 缩写为 SSM)来控制数据流从源位置流向目标位置。例如,二重唱演奏器 I 中所用的声卡就是一种流式设备,它用来播放存在用户的硬盘上的波形音频文件。PM 应用程序向媒体设备管理器(Media Device Manager,简写为 MDM)发出播放这些文件的请求。然后,MDM 调用波形音频 MCD,而后者使用流处理器将盘上的波形文件(源)中的数据放入声卡(目标)上的缓冲区。

3. CD 媒体控制驱动程序(CDMCIDRV)

提供了一个关于怎样控制非流式设备的例子。非流式设备在设备内部传递数据流。因为数据的源和目标都在设备内部,所以此类设备可以在内部传递数据流,而无需使用具有缓冲区的 I/O。因此,一个非流式设备也就不要求使用 OS/2 多媒体的 SSM 子系统。例如,二重唱演奏器 II 中所用的 CD-ROM 驱动器就是一个非流式设备,当 CD 音频 MCD 从程序中收到 PLAY, PAUSE, 或 STOP 命令后,它向 CD-ROM 驱动器发出相应的 IOCtls 来实现这些动作。在这里硬件包揽了所有的工作:从光盘读取数字化的信息,把它翻译成音频信号并送往某个输出端口,比如耳机插座等。

2.1 媒体控制驱动程序的体系结构

多媒体应用程序使用 mciSendCommand 或者 mciSendString 来发出设备控制命令,而媒体控制驱动程序就是这些命令的接收者。媒体控制驱动程序是一些 OS/2 的动态链接库(DDL),它可在一定程度上使应用程序具有设备无关性;而媒体设备管理器(MDM)则提供了通向 MCD 的接口、另外,MDM 还具有一定程度的局部管理的功能,这使一个应用

程序可以同时使用不同的 MCD 以及与其它应用程序共享这些 MCD。

图 2-1 描述了由 OS/2 多媒体系统提供的 MCD。

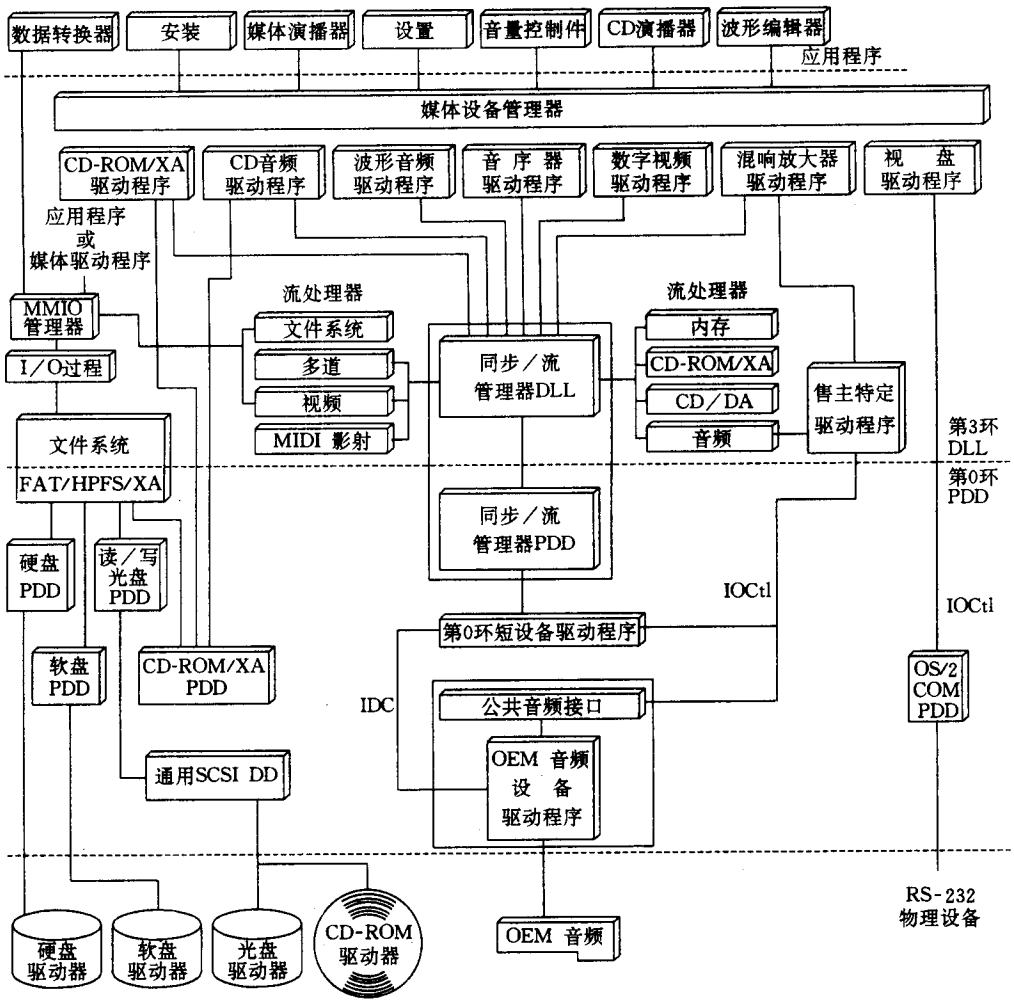

图 2-1 媒体控制驱动程序(MCD)的体系结构

2.2 媒体控制驱动程序的入口点

所有的 MCD 都接收以相同的形式传来的命令，而不论它所支持的是流式还是非流式设备。应用程序总是调用 mciSendCommand 或 mciSendString 函数来把命令发向 MCD 的入口点——mciDriverEntry。附带说明一下，有些命令是由 MDM 自己产生的。它们是一些有关保存和恢复一个实例的命令。

表 2-1 给出与 mciDriverEntry 有关的参数。

表 2-1 媒体控制驱动程序的入口点

参 数	说 明
PVOID pInstance	指向驱动程序的实例数据结构的指针。
USHORT usMessage	请求执行的动作。
ULONG ulParam1	消息的标志。本标志的定义因消息而异。
PVOID pParam2	第二个数据参数,它的意义取决于具体的消息。
USHORT usUserParm	随通知消息返回的用户参数。

mciDriverEntry 的功能是根据不同的消息来切换和执行相应的任务,比如像 MCI_OPEN 这样的消息。

你的驱动程序必须能够按照以下的形式来处理消息。

1. 你的驱动程序必须处理所有传来的消息。

2. 你的驱动程序必须处理它所支持的设备的设备类型消息。例如,如果你在编写一个视盘机的 MCD,你必须分析视盘设备特有消息的特定语法结构。

3. 你的驱动程序必须处理它所支持的设备所特有的消息。假设你的驱动程序控制以下设备类型中的一种:

(1) CD-ROM/XA 设备

(2) CD 音频设备

(3) 波形音频设备

(4) MIDI 音序器

(5) 数字视频设备

(6) 混响放大设备

(7) 视盘设备

MDM 已经定义好了这些设备的设备类型消息,也就是说,针对以上每一种设备类型列出了一张命令表。如果你想要支持某些设备特有消息,必须创建一张设备特有的命令表。

如果你的驱动程序所要支持的设备类型未在以上列出,你必须创建一张命令表,其中同时包括设备类型消息和设备特有消息。

2.3 命令消息的类型

媒体设备接口是一套已定义好的并且可扩展的媒体控制命令。MCD 怎样与相应的硬件设备驱动程序通信以执行命令消息,这完全由 MCD 决定。MCD 所使用的设备命令可分为必需命令消息、基本命令消息和系统命令消息。

注意:关于句法和相关参数,参阅《OS/2 多媒体编程参考》。

2.3.1 必需命令消息

必需命令消息可被所有设备识别,并对所有媒体设备产生相同的动作。表 2-2 列出

了你的 MCD 必须支持的必需命令消息。

表 2-2 必需命令消息

消　息	说　明
MCI_CLOSE	关闭设备。
MCI_GETDEVCAPS	获取设备的能力。
MCI_INFO	从设备获取文本信息。
MCI_OPEN	初始化一个设备的实例。
MCI_STATUS	从设备获取状态信息。
MCIDRV_SAVE	MDM 向 MCD 发此消息以保存上下文。
MCIDRV_RESTORE	MDM 向 MCD 发此消息，以恢复非活动设备的上下文的状态。

必需命令消息使用一个 ULONG 作为 ulParam1 参数，用以存放本命令消息的标志，并使用 pParam2 参数来指向一个本消息特定的数据结构。你的 MCD 如果想要扩展控制命令，则需要在已有基础上增加新的标志和新的数据结构的字段。当你扩展一个命令消息时，你的 MCD 必须保持对所有必要的标志及数据结构字段的支持。

表 2-3 列出了一些必需命令消息的标志和数据结构。如欲知媒体控制接口命令的详细资料，可参阅《OS/2 多媒体编程参考》。

表 2-3 必需命令消息、参数和数据结构

消　息	参数(ulParam1)	数据结构(pParam2)
MCI_CLOSE	MCI_NOTIFY	MCI_GENERIC_PARMS
	MCI_WAIT	
MCI_GETDEVCAPS	MCI_NOTIFY	MCI_GETDEVCAPS_PARMS
	MCI_WAIT	
	MCI_STATUS	
	MCI_GETDEVCAPS_EXTENDED	
	MCI_GETDEVCAPS_MESSAGE	
	MCI_GETDEVCAPS_ITEM	
MCI_INFO	MCI_NOTIFY	MCI_INFO_PARMS
	MCI_WAIT	
	MCI_INFO_PRODUCT	
MCI_OPEN	MCI_WAIT	MMDRV_OPEN_PARMS
	MCI_OPEN_SHARABLE	
	MCI_OPEN_ELEMENT	
	MCI_OPEN_MMIO	
MCI_STATUS	MCI_NOTIFY	MCI_STATUS_PARMS