第一章 CT Access 概述

1.1 CT Access 环境

这一部分提供了 CT Access 的环境信息，概要介绍了 CT Access 环境的组成元件。CT Access （计算机电话访问）作为一个开发环境，为独立于硬件设备的电话应用程序提供标准的编程接口。你必须在系统中安装 CT Access，并使用语音消息服务（Voice Message Service）来构建你的应用程序。

要了解有关 CT Access 的详细信息，请参阅《CT Access Developer's Reference Manual》。

1.1.1 编程模型

为了利用并发处理的优点，CT Access 使用了异步编程模型。调用时，大多数函数能迅速返回结果指示函数已经启动。当 CT Access 处理命令时，应用程序可以同时调用其他的函数。

在 CT Access 中有两种类型的函数：同步和异步。

同步函数在接收到返回值时运行结束。返回值可能是一个表示成功（SUCCESS）的值或者是错误（ERROR）码。

对于异步函数，如果返回值是 SUCCESS，则函数被成功初始化，处理的结果将通过某个事件异步地触发。如果返回值是不成功的，则函数初始化失败；因此，也不会产生与该函数关联的后继事件。

在函数执行过程中，某些特定条件的出现或某个状态的改变，将触发相应的后继事件。如果一个异步函数在初始化后又失败了，那么 CT Access 将给应用程序发送一个 DONE 事件，事件的值域中包含了一个错误码。

下面的表格总结了异步函数和同步函数的不同之处。在第七章中列出了所有语音消息服务的函数，并指明了它们是同步还是异步的。

特征	异步	同步
函数返回时运行结束	否	是
函数结束时返回一个 DONE 事件	是	否
函数返回后仍然可能失败	是	否

对于异步函数，CT Access 给服务发送一个命令，服务给电话语音卡发送一个命令。语音卡执行被请求的函数，给服务发送事件以指明它的状态（举例，函数开始、函数结束等等）。服务把事件发送给 CT Access，从而在应用程序中使用。

参看附录 B 中有关语音消息服务的错误码和事件的完整列表。

1.2 CT Access的组件

一个 CT Access 服务，由一组相关的电话应用函数组成。语音消息服务就是一个 CT Access 服务。

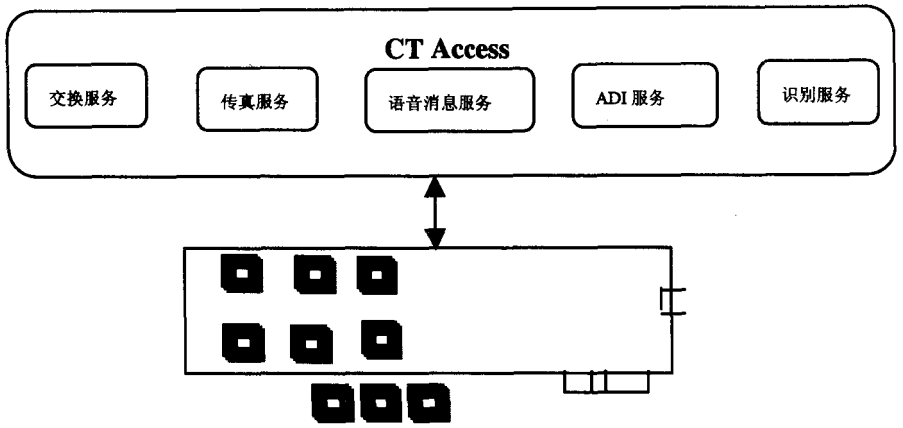

图 1-1 CT Access和语音消息服务

CTA 环境在一个单独的处理环境周边组织服务和伴随资源。一个 CTA 环境经常代表了一个控制单路电话呼叫的应用实例。

事件队列是 CT Access 服务与应用程序之间的通信通道。CT Access 服务通过产生事件来指明某些条件或状态的改变。而应用程序从事件队列中检索到那些事件。

CT Access 服务器是为应用程序提供服务的执行进程，并提供多进程的编程模式。有两种模式的运行方式：库模式和服务器式：

模式	描述
库模式	每一个应用程序分别和它自己的 CT Access 共享库连接起来。应用程序独立地创建和管理自己的运行环境。应用程序间不允许资源的共享
服务器模式	CT Access 服务器（ctdaemon）根据客户端应用程序的要求创建和管理运行环境。多个应用程序可以以共享和互斥的方式使用资源

要进一步了解有关以库模式和服务器模式运行 CT Access 的细节，请参阅《CT Access Developer's Reference Manual》。

1.1.3 CT Access 中的管理参数

通过修改相关参数来改变 CT Access 服务的特征。每一个语音消息服务的参数结构都有其默认值，这些默认参数对大多数配置已经足够。

CT Access 在 CTA 环境基础上对参数进行管理。CTA 环境将为在 CTA 环境中已经启动的服务保留一个参数副本。

下面列出了 CT Access 中的一些函数，通过它们可以获得或改变参数信息：

函数	描述
ctaGetParmByName	根据给出的参数名，检索到它的单个域值
ctaSetParmByName	根据给出的参数名，修改它的单个域值
ctaGetParmID	根据给出参数的描述字名字，检索到参数的 ID 号

（续表）

函数	描述
ctaGetParmInfo	检索参数域的定义
ctaGetParms	根据给出参数的结构，返回参数值
ctaRefreshParms	根据 CTA 环境中的默认值，重新设置环境中的参数值

要了解有关操作参数函数的信息，请参阅《CT Access Developer's Reference Manual》。该手册的附录 C 给出了语音消息服务参数的详细描述。

1.2 构建 CT Access 环境

在可以调用语音消息库的函数之前，应用程序必须先对 CT Access 进行初始化，并打开语音消息服务。建立 CT Access 应用程序要经过以下几个步骤：

1. 初始化 CT Access。
2. 创建事件队列。
3. 创建 CTA 环境，并使它与事件队列关联。
4. 启动每个 CTA 环境的服务。

在可以以服务器模式工作前，请确定 *cta.cfg* 文件中已包含了下面所列的：

- VCE 已在服务列表中指定。
- StartCtaServer 被置为 1（默认值）。

然后按照《CT Access Developer's Reference Manual》中所说的，启动 ctdaemon。

1.2.1 CT Access 的初始化

指定需要的服务和服务管理器的名字，通过调用 ctaInitialize 函数，注册你的服务。只有经过上述初始化，服务才能被应用程序启动。服务管理器在 Windows NT 中是动态链接库（DLL），在 UNIX 中是共享库，它们都可以与应用程序连接。

1.2.2 创建事件队列和 CTA 环境

在 CT Access 初始化后，就要创建事件队列和 CTA 环境。通过调用函数 ctaCreateQueue，创建一个或多个事件队列。指定与每个队列关联的服务管理器。语音消息服务的服务管理器是 VCEMGR。与服务管理器关联后，事件队列就可以使用服务管理器了。

通过调用函数 ctaCreateContext，获得从 ctaCreateQueue 函数返回的队列句柄。CTA 环境中的所有服务都是通过指定的事件队列得到的。

函数 ctaCreateContext 返回一个 CTA 环境的句柄（*ctahd*）。当应用程序调用语音消息服务的函数时，会提供 CTA 环境的句柄。事件和应用程序再次通信时也要与 CTA 环境联系。

参阅《CT Access Developer's Reference Manual》，了解有关使用 CTA 环境和事件队列来创建程序模型的细节。

1.2.3 启动服务

调用 ctaOpenServices 函数，在 CTA 环境中启动服务，并传递一个 CTA 环境句柄以及服务描述字列表。服务描述字指定了服务的名字、服务管理器以及特殊服务的参数（比如，

MVIP地址)。

1.2.4 与语音消息服务的链接

CT Access 的语音消息服务包含两个组件，语音消息服务接口（*vceapi*）和语音消息服务工具(*vcemgr*)。当建立一个新 CT Access 应用，使用语音消息服务时，需要与 *vceapi.lib*（在 UNIX 中，是 *libvceapi.so*）连接。在运行时动态载入 *Vcemgr.lib*(在 UNIX 中，是 *libvcemgr.so*)。

在 CT Access 的早期版本中，上述的语音消息服务接口和语音消息服务工具装载在同一个库中，*vcemgr* 库。所以提醒你必须修正原来的应用程序，使之能与 *vceapi.lib*(*libvceapi.so*) 连接。

参阅《CT Access Developer's Reference Manual》，可以了解有关服务实现的更多细节。

第二章　语音消息服务概述

2.1　CT Access 语音消息服务

CT Access 语音消息服务包括：

- 提供了一组函数，可以实现语音的播放、录制以及在文件或内存中对语音消息进行编辑。
- 支持 NMS 的 VOX 文件、WAVE 文件、平面（没有格式化）文件，而且可以扩展支持其他格式的语音文件。
- 提供更多高级的函数，可以播放连续的语音消息，可以给打开的文件段或内存块分配消息编号。
- 和 ADI 服务的播放、录制函数联合应用。比如，ADI 服务提供了可兼容的播放器／录制器。

要在语音消息服务中使用 ADI 服务，需要在同一个 CTA 环境中把这两个服务都启动。参阅《ADI Service Developer's Manual》，了解有关 ADI 服务的播放和录制函数的信息。

下面的部分说明了：

- 语音消息服务的特征
- 获取语音句柄的函数
- 语音消息服务的系统限制

参看第九章，提供了如何使用语音消息服务中播放函数和录制函数的演示程序。

2.2　特　　征

在准备用语音消息服务创建一个应用程序时，必须掌握它的以下特征：

- 语音文件类型
- 语音编码格式
- 语音句柄
- 消息
- 列表
- 当前消息和当前位置
- 提示生成器
- 消息文本

2.2.1　语音文件类型

CT Access 中的语音消息服务支持下列类型的文件：

NMS 的 VOX 文件格式。这种文件格式可以在同一个文件中保存多种信息，可以对语音消息进行删除、添加、编辑和注释等操作。参阅附录 D，了解有关 VOX 文

件格式的更多信息。

- WAVE 文件格式。
- 平面文件格式，即不包含头部或没有格式的文件。

Voxinfo 程序显示了有关 VOX 文件的信息。参阅 9.3.3 章节，可了解更多相关信息。

2.2.2 语音编码格式

录制语音文件时必须选择一种编码格式。选择格式主要考虑的是相对于一定保真度的压缩率。压缩率越高，需要的硬盘空间就越小，并能减少主机到电话语音卡的加载时间。

一个过载系统中，主机和电话语音卡之间的吞吐量需求可能引起语音录制或播放的间断。这叫做欠载运行。要解决这个问题，就有必要选择更高的压缩率。

CTA 环境中的 ADI 服务决定了支持哪些语音编码类型。要了解有关 ADI 服务的信息，请参考《ADI Service Developer's Manual》。

AG 语音卡支持下列的编码格式：

- NMS 的 ADPCM
- G.726-compliant ADPCM
- OKI ADPCM（6kHz 和 8kHz）
- Mu-率和 A-率 PCM
- 11kHz，8 位和 16 位的线性 PCM
- 8 位，16 位的线性 PCM
- IMA ADPCM（6kHz 和 8kHz）

注意　如果你使用的是 AG 语音卡，请参考《ADI Service Developer's Manual》中的 DSP 文件，其中列举了每种编码格式所需的 DSP 文件。

每种编码格式都有其最小的数据块大小值，称为“帧”。在 NMS 的语音卡中，一帧大概能播放 10 毫秒或 20 毫秒的时间，具体由编码方式确定。同一个语音文件中所有信息的编码方式必须相同。

编码指的是从主机来或到主机去的数据，一般保存在语音文件中。除了 VCE_ENCODE_NMS_64 编码，主机编码和线路编码是互相独立的，使用 mu-率或 a-率，这取决于语音卡在初始化时的配置情况。

注意　ADI 服务的编码格式与 ADI-打头的编码格式是一样的。

下表列出的是 CT Access 的编码格式：

编码格式	描述	大小举例（bit）	频率举例（赫兹）	帧长（字节）	帧时间（毫秒）	数据率（字节/秒）
VCE_ENCODE_NMS_16	NMS ADPCM 16kbit/s	2	8000	42	20	2100
VCE_ENCODE_NMS_24	NMS ADPCM 24kbit/s	3	8000	62	20	3100
VCE_ENCODE_NMS_32	NMS ADPCM 32kbit/s	4	8000	82	20	4100
VCE_ENCODE_NMS_64	PCM 帧 64kb-it/s	8	8000	162	20	8100

（续表）

编码格式	描述	大小举例（bit）	频率举例（赫兹）	帧长（字节）	帧时间（毫秒）	数据率（字节/秒）
VCE_ENCODE_MULAW	mu-率 64kbit/s	8	8000	80	10	8000
VCE_ENCODE_ALAW	A-率 64kbit/s	8	8000	80	10	8000
VCE_ENCODE_PCM8M16	PCM8K 单一制 kbit/s	16	8000	160	10	16000
VCE_ENCODE_OKI_24	OKI ADPCM 24 kbit/s	4	6000	30	10	3000
VCE_ENCODE_OKI_32	OKI ADPCM 32 千赫 8bit	4	8000	40	10	4000
VCE_ENCODE_PCM11M8	PCM11 千赫 8bit(单声道)（Wave）	8	11000	110	10	11000
VCE_ENCODEPCM11M16	PCM 11 千赫 16-bit（单声道）(Wave 格式)	16	11000	220	10	22000
VCE_ENCODE_G726	G.726 ADPCM 32kbit/s	4	8000	40	10	4000
VCE_ENCODE_IMA_24	IMA ADPCM 24kbit/s	4	6000	36	10	3600
VCE_ENCODE_IMA_32	IMA ADPCM 32kbits/s	4	8000	46	10	4600

2.2.3 语音句柄

许多函数能获取（或返回）一个语音句柄（Voice Handles)。每个句柄标识一个打开的语音对象，它是一个文件或内存块。

如何访问一个语音对象：

1. 通过调用函数 vceOpenFile，vceCreateFile,vceOpenMemory 得到语音句柄，或通过 vceCreateMemory 打开或创建一个文件，或为内存块创建一个环境。

注意　如果文件（或设备）是调用 vceAssignHandle 打开的，可以直接使语音句柄与之相关连。

2. 语音句柄与消息编号结合，来标识一个语音段落，作为播放、录制或编辑的对象。

语音句柄与一个 CTA 环境关联。在指定了一个语音句柄后，对应的 CTA 环境如图 2-1 所示。

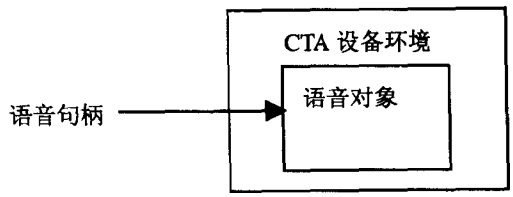

图 2-1 语音句柄和 CTA 环境的关联

注意 如果有多个线程，则每个线程都拥有自己的环境。当需要在内存载入提示消息时，每个线程必须拥有自己的语音句柄，与包含提示消息的内存块对应。

函数 VceClose 释放语音句柄。如果句柄是函数 vceOpenFile 或 vceCreateFile 创建的，则关闭相关的文件，释放由函数 vceCreateMemory 分配的内存。

2.2.4 消息

根据文件类型，一个语音对象可以包含 0、1 或多个消息。一个消息就是在逻辑上连续的语音块，由一个无符号的编号标识。通过录制或复制，可以在一个打开的文件或内存块中创建语音消息。非结构化的格式（平面文件或内存）在初建时只包含 0 个消息。另外的消息可以通过调用函数 vceDefineMessages 来定义。

消息的有效数量范围取决于对象的类型。

类型	消息的有效数量范围
VOX	0 到 32,767
平面	0 到 65,535
内存	0 到 65,535
WAVE	0

在有效数量范围内未使用（或已删除）的消息编号指的是零长度的消息，并且在引用时不会导致错误。

特殊的消息编号 VCE-ALL-MESSAGES ，可以使你把一个对象的所有消息作为一个消息来操作。使用此编号，可以对所有消息进行复制、播放或清除操作。但是不能对此消息编号进行录制或写入。

2.2.5 列表

一个列表是一个或多个消息。使用函数 vcePlayList 可以连续播放一组消息，而且每个消息间不存在延时。用该方法来播放一个提示串，这个提示串已经由函数 vceBuildPromptList 转化成一个消息编号序列。当然，列表中的消息编码格式必须一致。如果信息存放在不同的文件中，则调用函数 vceSetCurrentList ，再用函数 vcePlay 播放。

2.2.6 当前消息和当前位置

在播放一个消息或一个消息列表时，它就成为当前消息，可以进行暂停、继续和重定位等操作。当前消息和 CTA 环境句柄相关联。当前消息有一个当前位置。

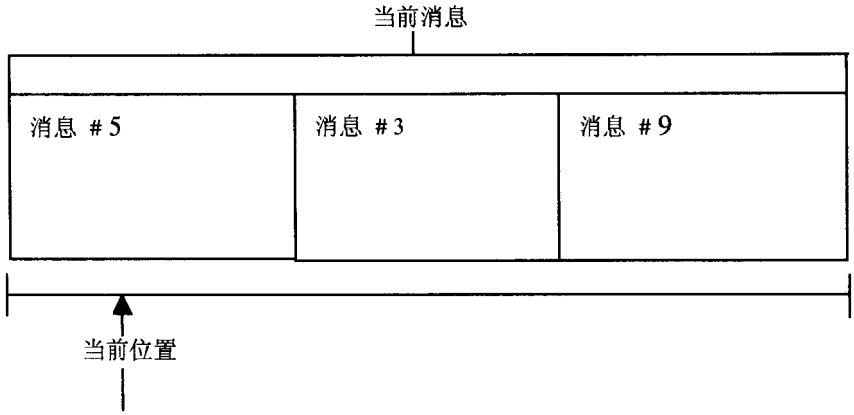

图 2-2 当前消息／当前位置

使用下面的函数来修改当前位置：

- vceErase
- vcePlay
- vceRead
- vceRecord
- vceSetPosition
- vceWrite

下面的函数用来定义当前消息：

- vceConvertMessage
- vceCopyMessage
- vceErase Message
- vcePlayList
- vcePlayMessage
- vceRecordMessage
- vceSetCurrentList
- vceSetCurrentMessage

参看第八章，了解有关这些函数的更多信息。

如果一个包含当前消息（列表）的语音对象被关闭，那其当前消息就不存在了。所有进行中的播放或录制操作都将中止。

注意 电话呼叫的挂断并不影响当前消息。

2.2.7 提示生成器

语音消息服务的提示生成器中，有把文本串转换成消息列表的函数。提示生成器使应用程序可以向用户通知日期、时间和费用数目等信息。一个文本串包含的是变量数据，比如 10/21/96，送到提示生成器后，按照标准的消息库的要求，根据提示规则表建立消息列表。消息编号的输出列表可以直接交付给函数 vcePlayList。

而且，提示生成器有一套规则，规定了文本串如何翻译为语音消息。一个相关的语音

文件包含了与这些规定对应的消息。例如，文本串 10翻译成语音消息“ ten”。

参阅第六章，了解更多关于使用提示生成器的信息。参考附录 E，其中描述了运行文件和创建自定义规则的命令。

2.2.8 消息文本

语音消息服务包含了复制、读出以及写入消息文本的函数。消息文本是一个描述性的字符串,可以附属到 NMS VOX 文件的信息上。

VOX 文件有一个与每个消息关联的文本串（或其他数据）。通常用来包含提示标识符。

下面的表格列出了在什么时候使用 CT Access 语音消息服务中可用的消息文本函数：

如果你需要...	使用函数...
把相关的数据从一个语音消息复制到另一个	**vceCopyMessageText**
从 VOX 文件中读出指定语音消息的消息文本	**vceReadMessageText**
保存 VOX 文件中指定语音消息的相关消息文本	**vceWriteMessageText**

参阅第八章，了解更多这些函数的信息。

2.3 语音句柄的获取

现在你对语音消息服务的一些重要特征已经有一定了解，可以使用这种服务创建应用程序了。

要使用语音对象，需要一个语音句柄。

如果你需要.....	使用函数.....
访问一个存在文件，进行播放或录制	**vceOpenFile**
根据指定类型，创建新文件	**vceCreateFile**
访问某个内存区间	**vceOpenMemory**
访问一个已打开的文件（或设备）	**vceAssignHandle**

参阅第八章，了解更多这些函数的信息。

当你获取了一个语音句柄后，就可以用该句柄和一个消息编号来标识语音对象中的一个语音段，使它可以被播放、录制和编辑。

2.4 系 统 限 制

下面是使用 CT Access 语音消息服务时需要注意的系统限制：

在给定的 CTA 环境中，播放和录制操作是互相排斥的。

一个语音句柄通常只和一个 CTA 环境关联。要访问一个在多 CTA 环境（共享一个提示文件）中的语音对象，必须为每个 CTA 环境单独地打开一个语音对象。

第三章　播放和录制

3.1 引　　言

本章介绍了如何用语音消息服务来完成语音文件的播放和录制。

3.2 播　　放

语音消息服务中有两种类型的播放函数：

- 面向消息的函数，vcePlayMessage 和 vcePlayList，对一个指定的消息或消息列表进行操作。它们获取一个语音句柄和消息的编号，或者是消息列表的编号。此类函数进行播放时是从第一个消息的首部开始的。
- 面向位置的播放函数，vcePlay，它对指定环境中的当前消息进行操作。此类函数播放时，是从最近访问过的消息或消息列表的当前位置开始的。

在语音消息服务中没有面向缓冲（内存）的播放函数。如果你想对缓冲进行直接访问，就必须使用 ADI 服务中的 adiPlayAsunc 或 adiStartPlaying 函数。请参考《ADI Service Developer's Manual》和《ADI Service Function Reference Manual》，了解更多的内容。

演示程序 *vceplay* 演示了如何对来自一个或多个语音文件的消息进行播放的过程。参阅 9.2.3 部分了解更多的信息。

3.2.1 播放函数

下面的表格列出了在什么时候使用 CT Access 语音消息服务中可用的播放函数：

如果你想要。。。	那么要调用函数。。。
从当前消息的当前位置开始播放	**vcePlay**
从消息列表的第一个消息的首部开始播放	**vcePlayList**
从指定的语音对象中播放一个消息	**vcePlayMessage**
把播放的速度调整到指定的大小	**vceSetPlaySpeed**
通过调节正在播放的消息的放大率或者增益来改变消息播放的音量	**vceSetPlayGain**

3.2.2 播放完成的条件

下面的表格列举了当播放结束时，语音消息服务返回的条件码。这些条件码在 VCEEVN_PLAY_DONE_EVENT 中声明。

注意　如果播放失败，则值域中将包含一个错误码。

如果。。。	则 DONE 事件包含。。。
已经到达当前消息或列表的末尾	CTA_REASON_FINISHED
函数 **vcePlay** 设定的播放时间到达	CTA_REASON_TIMEOUT

（续表）

如果。。。	则 DONE 事件包含。。。
调用函数 **vceStop** 使播放停止	CTA_REASON_STOPPED
接收到某个按键式的数字，使 DTMF 异常中止	CTA_REASON_DIGIT
参数的相应位置位，而使播放停止呼叫结束	CTA_REASON_RELEASED
发生语音识别事件，而使播放停止	CTA_REASON_RECOGNITION
相关的设备服务不支持你要播放的对象的编码方式	CTAERR_FUNCTION_NOT_AVAIL

3.2.3 速度和增益的调节

你可以在播放的任何时候设置音量（调用函数 vceSetPlayGain）和播放速度（调用函数 vceSetPlaySpeed）。新的音量和新的播放速度存贮在当前 CTA 环境中。如果是当前正在执行的播放，则上述调节会立即生效。如果想要让增益和速度的改变延至下一个播放函数，则需要把下一个播放的增益或速度参数设置到 VCE_CURRENT_VALUE 中。

播放的速度也可以通过一些编码方式进行调节。NMS ADPCM 编码方式（VCE_ENCODE_NMS_xx）和 OKI ADPCM编码方式（VCE_ENCODE_OKI_xx）都可以实现对播放速度的控制。

要提高播放的速度，就要改大默认的最大播放速度参数，一般为 100。当使用一个较高的最大播放速度值时，为了支持该播放速度，就要安装必要的数字信号处理（DSP）器件。通过改变速度参数的值，你就能用一个很高的速度进行播放（直到最大速度）。

如果要在 AG 板卡中用 NMS ADPCM 编码方式来提高播放速度，则要把 *ag.cfg* 文件中的 *voxp.dsp* 和 *voxr.dsp* 用 *voice.dsp* 代替。

注意 如果要以超过 100 的最大速度进行播放，必须要有附加的数字信号处理（DSP）器件，否则只能以普通的速度进行播放。要了解你的 AG 板卡和配置是否支持提速，请参阅《AG Runtime Configuration and Developer's Manual》。

调用函数 vceGetCurrentInfo ，可以得到当前增益和速度值。

3.2.4 参数

播放函数用一个指针指向参数结构。如果传递的是 NULL，那参数就取默认值。参阅附录 C，了解有关播放参数的更多信息。参阅《CT Access Developer's Reference Manual》了解有关参数的更多内容。

域名	默认值	描述
DTMFabort	0xffff	DTMF 异常中止（位屏蔽）
Gain(增益)	0	按分贝计的播放增益，或由 VCE_CURRENT_VALUE 以使用最近的增益设置。AG 板卡的增益有效范围是-54~24
播放速度	100	按百分比计的速度，或 VCE_CURRENT_VALUE 以使用最近的速度设置。AG 板卡的速度有效范围是 50~最大速度
最大速度	100	按百分比计的播放速度最大值。AG 板卡的最大速度有效范围是 100~200

3.3 录　　制

在语音消息服务系统中有两种类型的录制函数：

- 面向消息的函数 vceRecordMessage，对一个指定的消息编号进行操作。这个函数创建一个新的消息来代替原来存在的消息。
- 面向位置的函数 vceRecord，对指定环境中的当前消息进行操作。它在消息的当前位置处开始进行录制，该消息是最近被访问过的。

在语音消息服务中没有面向缓冲（内存）的录制函数。如果想对缓冲进行直接访问，就必须使用 ADI 服务中的函数 adiRecordAsunc 或函数 adiStartRecording。请参考《ADI Service Developer's Manual》和《ADI Service Function Reference Manual》，了解更多 ADI 服务函数的内容。

演示程序 *vceplay* 演示了如何对来自一个或多个语音文件的消息进行复制的过程。参阅 9.2.4 部分了解更多的信息。

3.3.1 录制函数

一般在发出蜂鸣声后，录制开始。录制函数可对录制的时间进行限定。录制过程中检测到的静音可引起录制的异常中断。

下面的表格列出了在什么情况下要调用语音消息服务中的哪个录制函数：

如果你想要...	则调用函数...
从当前消息的当前位置开始录制	**vceRecord**
把消息录制到指定的语音对象中	**vceRecordMessage**

3.3.2 录制完成条件

下面的表格列举了当录制结束时，语音消息服务返回的条件码。这些条件码在 VCEEVN_RECORD_DONE_EVENT 中声明。

注意　如果录制失败，则值域中将包含一个错误码。

如果...	DONE 事件将包含...
函数 **vceRecord** 设定的播放时间到达	CTA_REASON_TIMEOUT
语音对象中已没有可用空间，在覆盖模式下已经到达被覆盖消息的结尾	CTA_REASON_FINISHED
在录制开始时检测到静音	CTA_REASON_NO_VOICE
录制一段时间后，检测到静音	CTA_REASON_VOICE_END
由于调用函数 **vceStop**,使录制停止	CTA_REASON_STOPPED
由于接收到某个按键音数字，使 DTMF 异常中止参数的相应位置位，而使录制停止	CTA_REASON_DIGIT
呼叫结束	CTA_REASON_RELEASED
由于发生了语音识别事件，而使录制停止	CTA_REASON_RECOGNITION
由于相关的设备服务不支持录制对象的编码方式	CTA_REASON_NOT_AVAIL

3.3.3 消息大小的调整

调用函数 vceRecordMessage 替代一个消息，新的消息可能比原来的消息大或小。对消息大小调整的能力取决于语音对象和消息在对象中所处的位置。 VOX 文件中的消息或位于平面（未格式化）文件末尾的消息可以无限制地扩展（直到硬盘全满）。如果在文件末尾的消息被较小的消息代替，则文件会缩小（在文件关闭时）。函数 vceOpenMemory 指定了内存单元块的大小，内存块中的消息大小要受此限制。

在一个平面文件或一个内存块中一般只有一个消息。如果在一个平面文件或一个内存块中存在多个消息，当中间的消息要被替代时，新的消息不能比原有消息大。当新的消息超过原来消息的大小时，vceRecordMessage 会发出 CTA_REASON_FINISHED 消息。

3.3.4 参数

播放函数用一个指针指向参数结构。如果传递的是 NULL，那参数就取默认值。参阅附录 C，了解有关播放参数的更多信息。参阅《CT Access Developer's Reference Manual》了解有关参数的更多内容。

在来话业务（录制）中，在对语音进行压缩和存储前使用自动增益控制 (AGC)算 法，使保存的语音音量控制在目标范围内。

下面的参数是控制蜂鸣声、静音检测、录制增益(包括自动增益控制)和 DTMF 交互的：

域名	默认值	描述
DTMFAbort	0xffff	DTMF 异常中止（位屏蔽）
Gain	0 dB	动增益控制，这是编码开始时的初始增益。AG 板卡的有效范围是 -54到 24
Novoicetime	5000ms	在录制被 CTA_REASON_NO_VOICE 条件中止前，录制开始时保持静音的最大时间值。使用 0 值可取消此计时器。AG板卡的有效范围是 0 到 65535
Silencetime	3000ms	在录制被 CTA_REASON_VOICE_END 条件中止前，录制过程中静音状态可保持的最长时间。使用 0值可取消此计时器。AG 板卡的有效范围是 0 到 65535
Silenceampl	45dBm	低于此值，就认为是静音状态。AG板卡的有效范围是-51 到-15。
Beepfreg	1000Hz	录制时蜂鸣声的频率。使用 0 值可取消蜂鸣功能。AG 板卡的有效范围是 200到 3600
Beepampl	20dBm	蜂鸣声的音量。AG 板卡有效范围是:-54到 3
Beeptime	200ms	蜂鸣的持续时间。使用 0 值可取消蜂鸣功能。AG 板卡的有效范围是 0 到 65535
AGCenable		是否可以使用自动增益控制的标志

注意　当使用 AG 板卡时，上述表中的参数会覆盖 ADI.RECORD 类中的对应域。ADI.RECORD 包含了几个附加域，可以实现对自动增益控制（ AGC ）的控制。

图 4，图 5，图 6 说明了三种导致录制函数终止运行的情况：

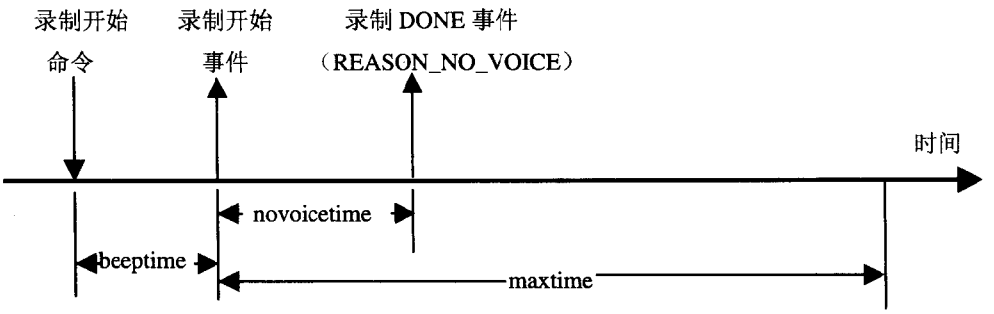

图 3-1　录制终止——无语音

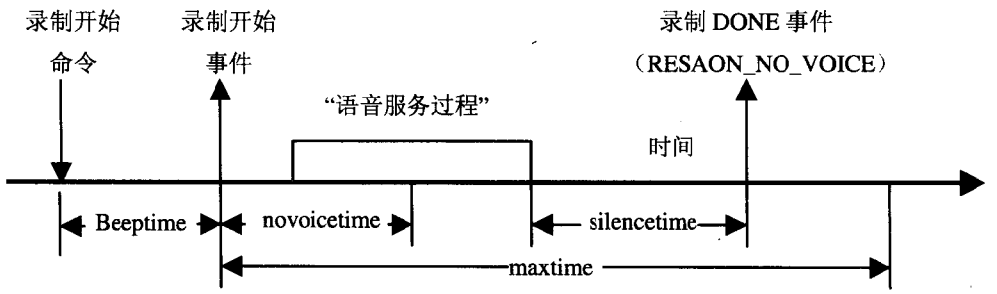

图 3-2　录制终止——语音服务结束

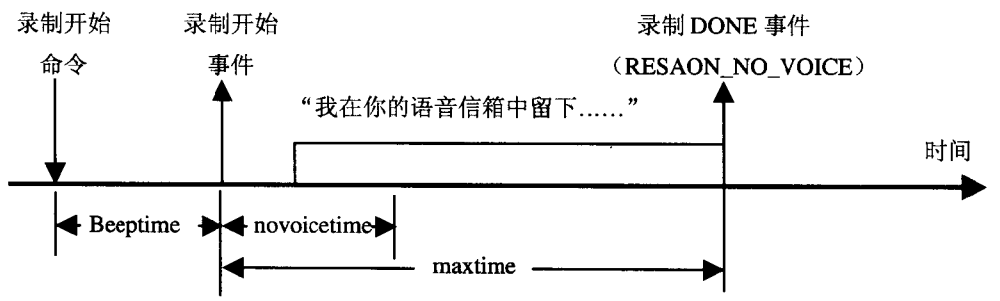

图 3-3　录制终止——超时

3.4　语音播放和语音录制状态

本段主要说明和解释了播放和录制过程中状态的转化。

空闲：播放和录制函数都未激活。

播放：应用程序调用播放函数（vcePlay,vcePlayList,vcePlayMessage），**开始进行播放，语音消息服务就进入了播放状态。**

录制：应用程序调用录制函数（vceRecord,vceRecordMessage），**开始进行录制，语音消息服务就进入了录制状态。**

播放停止：应用程序调用函数 vceStop，**停止播放。语音消息服务产生** VCEEV-N_PLAY_DONE **事件，并返回空闲状态。**

录制停止：应用程序调用函数 vceStop，停止录制。语音消息服务产生 VCEEVN RECORD_DONE 事件，并返回空闲状态。

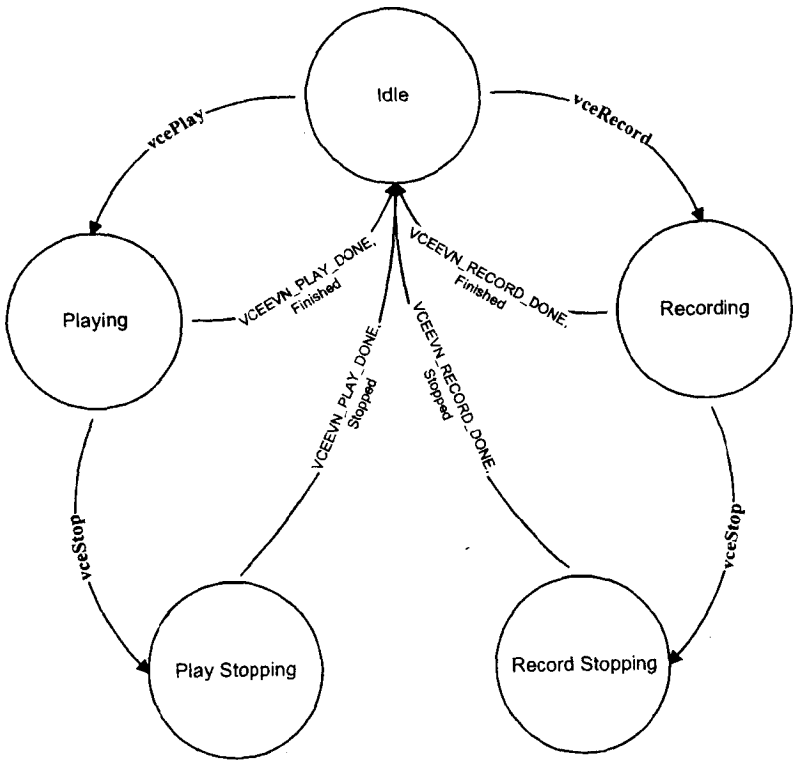

图 3-4 播放和录制的状态

播放和录制状态一直保持，直到：

接收到 VCEEVN_PLAY_DONE 事件或 VCEEVN_RECORD_DONE 事件。

在 3.2.2 或 3.3.2 段落中描述的终止条件出现。不包括 CTA_REASON_STOOPPED 条件。

3.5 停　　止

调用 vceStop 函数停止语音的播放或录制。播放或录制的停止将产生 DONE 事件。DONE 事件的值域中包含条件 CTA_REASON_STOPPED。

注意　当播放或录制的语音对象被函数 vceClose 关闭，将使播放或录制停止，这种情况下也会返回 CTA_REASON_STOPPED 条件。在播放一个消息列表时，任何一个包含列表中消息的对象被关闭，都将使播放停止。

3.6 DTMF 交互

初始化播放或录制时，如果远端的 DTMF 键被按下，应用程序可以执行某个函数去终止播放或录制。在默认情况下键入任何 DTMF 键，播放或录制都将被终止。在播放参数结构和录制参数结构中，提供了 DTMF 的异常中止屏蔽字，指定了哪个 DTMF 键可以终止函数的运行。

DTMF 异常屏蔽字是一个 16 位的入口，每一位都对应电话键盘上的一个键。在异常屏蔽字中设置了哪一位，则当对应的键被按下时，就会导致语音函数的终止。下表列出了DTMF异常屏蔽位对应的电话键盘上的键：

	比较重要的位											不太重要的位				
位的位置	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DTMF 键	D	C	B	A	#	*	9	8	7	6	5	4	3	2	1	0

举例，如果异常屏蔽字设置为 0x3FF，那么从键盘键入从 1到 9 的数字都将终止播放／录制函数的运行。头文件 *vcedef.h* 包含了每个数字和某些数字组的宏定义#defines（VCE DTMF xxx）。