

DECtalk™ Software

Reference Guide

October 1999

This guide provides an overview and reference listings of the DECtalk Software Application Programming Interface calls and in-line commands. It also contains a listing of all reference tables for phonemic symbols, stress and syntactic symbols, tone tables, and homographs.

Revision and Update Information: This guide supersedes the DECtalk Software Reference Guide, Version 4.5

Operating System: Microsoft Windows 95/98/NT/CE
Tru64 UNIX

Software Version: DECtalk Software Version 4.6

SMART Modular Technologies, Inc. and its subsidiaries
Fremont, California

October 1999

The information in this publication is subject to change without notice.

SMART MODULAR TECHNOLOGIES, INC. AND ITS SUBSIDIARIES SHALL NOT BE LIABLE FOR TECHNICAL OR EDITORIAL ERRORS OR OMISSIONS CONTAINED HEREIN, NOR FOR INCIDENTAL OR CONSEQUENTIAL DAMAGES RESULTING FROM THE FURNISHING, PERFORMANCE, OR USE OF THIS MATERIAL. THIS INFORMATION IS PROVIDED “AS IS” AND SMART MODULAR TECHNOLOGIES, INC. AND ITS SUBSIDIARIES DISCLAIM ANY WARRANTIES, EXPRESS, IMPLIED, OR STATUTORY, AND EXPRESSLY DISCLAIMS THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR PARTICULAR PURPOSE, GOOD TITLE, AND AGAINST INFRINGEMENT.

This publication contains information protected by copyright. No part of this publication may be photocopied or reproduced in any form without prior written consent from SMART Modular Technologies, Inc.

© 1999 SMART Modular Technologies, Inc.. All rights reserved. © 1997, 1998, 1999 Digital Equipment Corporation All rights reserved.

Printed in U.S.A.

The software described in this guide is furnished under a license agreement or nondisclosure agreement. The software may be used or copied only in accordance with the terms of the agreement.

The stylized “S” and “SMART” as well as “SMART Modular Technologies” are registered trademarks of SMART Modular Technologies, Inc.

The SMART logo is a trademark of SMART Modular Technologies, Inc.

DECTalk is a trademark of SMART Modular Technologies, Inc.

Compaq, the Compaq logo, and the Digital logo registered in U.S. Patent and Trademark Office. DEC, DIGITAL, and OpenVMS are trademarks of Compaq Computer Corporation.

Intel is a trademark of Intel Corporation.

Microsoft, Windows, Windows 95, and Windows NT are registered trademarks of Microsoft Corporation.

Motif is a registered trademark of Open Software Foundation, Inc.

RoboHELP is a registered trademark of Blue Sky Software Company. SoundBlaster is a registered trademark of Creative Labs, Inc.

UNIX and Tru64 UNIX are registered trademarks in the United States and other countries licensed exclusively through X/Open Company, Ltd.

Other product names mentioned herein may be trademarks and/or registered trademarks of their respective companies.

This document was prepared using Microsoft Word 97 to generate MS Word 6.0/95 compatible files. The online help system was prepared with Microsoft Word 97 and the RoboHELP help authoring tool.

Chapter 1 DECTalk Software API Calls	1-1
TextToSpeechAddBuffer	1-3
TextToSpeechCloseInMemory	1-5
TextToSpeechCloseLang	1-6
TextToSpeechCloseLogFile	1-7
TextToSpeechCloseWaveOutFile	1-8
TextToSpeechEnumLangs	1-9
TextToSpeechGetCaps	1-10
TextToSpeechGetFeatures	1-11
TextToSpeechGetLanguage (not supported)	1-12
TextToSpeechGetRate	1-13
TextToSpeechGetSpeaker	1-14
TextToSpeechGetStatus	1-15
TextToSpeechLoadUserDictionary	1-16
TextToSpeechOpenInMemory	1-17
TextToSpeechOpenLogFile	1-19
TextToSpeechOpenWaveOutFile	1-21
TextToSpeechPause	1-23
TextToSpeechReset	1-25
TextToSpeechResume	1-27
TextToSpeechReturnBuffer	1-28
TextToSpeechSelectLang	1-29
TextToSpeechSetLanguage (not supported)	1-30
TextToSpeechSetRate	1-31
TextToSpeechSetSpeaker	1-32

TextToSpeechShutdown	1-33
TextToSpeechSpeak	1-34
TextToSpeechStartLang	1-36
TextToSpeechStartup (Windows only)	1-37
TextToSpeechStartup (UNIX only)	1-39
TextToSpeechStartupEx	1-42
TextToSpeechSync	1-46
TextToSpeechTyping	1-47
TextToSpeechUnloadUserDictionary	1-48
TextToSpeechVersion	1-49
TextToSpeechVersionEx	1-50

Chapter 2 — DECTalk Software In-Line Commands2-1

In-Line Commands: Overview	2-1
Comma Pause [:comma]	2-4
Design Voice [:dv]	2-5
Dial Tones [:dial]	2-6
Error [:error]	2-7
Index Mark [:index mark]	2-8
Log [:log]	2-9
Mode [:mode]	2-10
Name [:name]	2-15
Period Pause [:period]	2-16
Phoneme Interpretation [:phoneme]	2-17
Pitch [:pitch]	2-19
Play Wave Files [:play]	2-20
Pronounce [:pronounce]	2-21
Punctuation [:punct]	2-22

Rate Selection [:rate]	2-23
Say [:say]	2-24
Skip [:skip]	2-25
Sync [:sync]	2-26
Tone [:tone]	2-27
Volume [:volume]	2-28
Monaural Volume Control	2-28
Stereo Volume Control	2-28
Chapter 3 — Using In-Line Commands	3-1
Changing Rhythm, Stress, and Intonation	3-2
Developing an Electronic Mail-Reading Application	3-2
Optimizing the Quality of Spoken Text	3-4
Index Marks for Speech Status	3-6
Speaking Rate	3-7
Adjusting Period and Comma Pause Durations	3-9
Text-Tuning Example	3-10
Original Version	3-10
Revised Version	3-11
Avoiding Common Errors	3-12
Chapter 4 — DECtalk Software Reference Tables	4-1
Phonemic Symbols	4-2
Stress and Syntactic Symbols	4-6
Phonemes Listed in Unicode Sequence	4-7
Pitch and Duration of Tones	4-11
Homographs	4-13
Homograph Phonetics - (A)	4-14
Homograph Phonetics - (B-C)	4-15
Homograph Phonetics - (D-G)	4-16
Homograph Phonetics - (I-L)	4-17
Homograph Phonetics - (M-P)	4-18

Homograph Phonetics - (R).....	4-19
Homograph Phonetics - (S-W).....	4-20
Supported SAPI Functions	4-21
 Chapter 5 — Customizing a DECtalk Software Voice.....	 5-1
Design Voice [:dv]	5-2
Definitions of DECtalk Software Voices.....	5-4
Changing Gender and Head Size	5-5
Sex, sx	5-5
Head Size, hs	5-6
Higher Formants, f4, f5, b4, and b5.....	5-6
Changing Voice Quality.....	5-8
Breathiness, br.....	5-8
Lax Breathiness, lx	5-8
Smoothness, sm	5-9
Richness, ri	5-9
Nopen Fixed, nf	5-9
Laryngealization, la.....	5-10
Changing Pitch and Intonation.....	5-11
Baseline Fall, bf	5-11
Hat Rise, hr.....	5-11
Stress Rise, sr.....	5-12
Assertiveness, as.....	5-13
Quickness, qu	5-13
Average Pitch, ap, and Pitch Range, pr	5-13
Changing Relative Gains and Avoiding Overloads.....	5-15
Loudness, g5	5-15
Sound Source Gains, gv, gh, gf, and gn	5-16
Cascade Vocal Tract Gains, g1, g2, g3, and g4.....	5-16
Saving Changes as Val's Voice	5-18
Save, save.....	5-18
Summary of Design Voice Options.....	5-19
 Chapter 6 Preprocessor Rules for Parsing	 6-1
Parsing email	6-1
Parsing Punctuation	6-1

Interpreting Punctuation Marks as Words.....	6-2
Interpreting Punctuation Marks as Punctuation	6-2
General Rules.....	6-3
German.....	6-3
Spanish (Castilian and Latin American).....	6-3
English (UK).....	6-4
English (US, UK).....	6-4
Index	1

Figures

Figure 4-1 DECTalk Software Singing “Happy Birthday”.....	4-11
---	------

Tables

Table 1-1 DECTalk Software API Calls.....	1-1
Table 2-1 DECTalk Software In-Line Commands	2-2
Table 2-2 DECTalk Interpretation of Special Characters	2-11
Table 4-1 Phonemic Symbols	4-3
Table 4-2 Stress Symbols	4-6
Table 4-3 Syntactic Symbols.....	4-6
Table 4-4 Phonemes in Unicode Sequence.....	4-7
Table 4-5 Phoneme Syntax for Singing	4-11
Table 4-6 Tone Table	4-12
Table 4-7 Homograph Phonetics - (A).....	4-14
Table 4-8 Homograph Phonetics - (B-C).....	4-15
Table 4-9 Homograph Phonetics - (D-G)	4-16
Table 4-10 Homograph Phonetics - (I-L).....	4-17
Table 4-11 Homograph Phonetics - (M-P)	4-18
Table 4-12 Homograph Phonetics - (R).....	4-19
Table 4-13 Homograph Phonetics - (S-W).....	4-20
Table 4-14 Supported Functions of the Microsoft Speech API	4-21
Table 5-1 [:dv] Command Options	5-2
Table 5-2 Speaker Definitions for All DECTalk Software Voices	5-4
Table 5-3 Head Size and Shape Options.....	5-5
Table 5-4 Voice Quality Options.....	5-8
Table 5-5 Fundamental Frequency Contour Options.....	5-11

Table 5-6 Internal Resonator Options 5-15

Purpose and Audience

This guide is written for the general user or programmer who wants a ready reference to DECTalk Software Application Programming Interface (API) calls, in-line commands, and reference tables. The information in this guide is accurate for Windows 95/98/NT/CE and Tru64 UNIX implementations of DECTalk Software. Use this guide in conjunction with the *DECTalk Software Programmer's Guide*.

Structure

The design of this guide gives you quick and easy access to information. Its organization can help you easily learn about new topics and perform specific tasks related to the use of the program applets for development of a DECTalk Software application.

The guide is organized as follows:

Chapter 1	DECTalk Software API Calls
Chapter 2	DECTalk Software In-Line Commands
Chapter 3	Using In-Line Commands
Chapter 4	DECTalk Software Reference Tables
Chapter 5	Customizing a DECTalk Software Voice
Chapter 6	Preprocessor Rules for Parsing
Glossary	Definitions of Terms Used in DECTalk Documentation

What's New in DECTalk Software V4.6?

DECTalk Software V4.6 contains the following new features and corrections:

- Installation tested on Windows 98.
- Support for the German language.
- Disabled license error pop-up window from DAPI engine.
- Significantly increased Microsoft Speech API (SAPI) compliance; see appendix3.txt.
- Enhanced Spanish Latin American and Spanish Castilian.

- CE version, which includes a static version of windic.exe for the Windows host to support a user-defined dictionary, and full registry support.
- Y2K compliance.
- Over 170 bugs fixed, including:
 - ❑ Various application or system hangs or crashes caused by data path, synchronization and dictionary failures corrected.
 - ❑ User dictionary compiler failures are corrected.
 - ❑ Wave file headers are corrected.
 - ❑ Use of extended ASCII characters are corrected.
 - ❑ New in-line command **Say filtered-letter** has been added.
 - ❑ Number processing changes automatically by language.
 - ❑ Memory leaks are corrected.
 - ❑ Main dictionary wordclass and function words have been added.
 - ❑ Word and number stressing are corrected.
 - ❑ Diphthong, phoneme, homograph and allophone processing are corrected.

What's New in DECtalk Software V4.5?

DECTalk Software V4.5 for Windows 95 and Windows NT contains the following new features and corrections:

- Distributed on CD-ROM.
- Year 2000 support speaks four-digit year specifications as four-digit year specifications, for example, the year 1998 is spoken as nineteen ninety eight, and speaks 2-digit year specifications as a 2-digit number without assuming the century, for example, the year 98 is spoken as ninety eight.
- United Kingdom English is added.
- SAPI mark syntax (\mrk=\ tag) is working. The TextData interface now accepts tagged text. The \mrk=\ tag is the only tag supported. Other tags are ignored.
- ITTSBufNotifySink: The BookMark and WordPosition interfaces are now functional. The timing of the TextDataStarted and TextDataDone notifications is corrected.

- ITTSNotifySink: The Visual interface is now functional; the timing of the AudioStart and AudioStop notifications is corrected.
- DECtalk Software V4.5 provides better indexing performance than previous versions.
- Multi-Language (ML) support is functional.
- Sample programs Speak and dtsample are updated to Multi-Language (ML) functionality. All sample applets are ML-compliant code. Some sample applets demonstrate ML-application programming.
- The User Dictionary compiler now handles accent characters (extended ASCII).
- OLE notification support, AudioStart and AudioStop, now returned.

The current DECtalk Software implementation of the Microsoft Speech API (SAPI) is described in the appendix3.txt file that is included with the DECtalk Software installation files.

What's New in DECtalk Software V4.4?

DECtalk Software V4.4 for the Windows operating system contains the following new features:

- Available in multiple languages: American English, Castilian Spanish, Latin-American Spanish, and German.
- Advanced preprocessing rules.
- Email parsing has been added to the Mode in-line command to expand abbreviations and initialisms as well as stripping unwanted header and footer information.
- The SPEAK.EXE demonstration program now shows highlighting.
- The **Sync** command, which previously had to be reversed, now works as documented.

What's New in DECtalk Software V4.3?

DECtalk Software V4.3 for the Windows operating system contains the following new features:

- Support for operation on the Windows operating system.
- Support for the Microsoft Speech API (SAPI).

- Support for OLE.
- Support for multi-threaded and multi-instance operation.
- Expanded main dictionary.
- A new document, the *DECtalk Software Reference Guide* has been added. This guide provides an overview and reference listings of the DECtalk in-line commands.
- User-dictionary suffix processing to help locate words in the user dictionary.
- Expanded pronunciation rules and improved pronunciation.
- Homograph processing.
- Improved in-line index-mark processing.
- Added the following in-line commands:
 - ❑ Play command to play wave audio files in line with text
 - ❑ Tone command to generate tones
 - ❑ Dial command to generate DTMF tones used to dial telephone numbers
 - ❑ Stereo volume control commands
- A new Text-to-Speech DDE (Dynamic Data Exchange) server that can be used to run DECtalk Software from within an application, such as MS Word.
- A speech-to-memory Sample program.
- An enhanced User Dictionary Builder that automatically translates text strings into phonemes.
- A new command-line applet named Say that runs DECtalk Software from the Windows command line.

Conventions

The following conventions are used in this guide:

Convention	Meaning
enter	<i>Enter</i> means type the required information and press the Enter key.
mouse	<i>Mouse</i> refers to any pointing device, such as a mouse, a puck, or a stylus.
MB1	<i>MB1</i> indicates the left mouse button.
click	<i>Click</i> means press and release MB1.
Double click	<i>Double click</i> means to press and release MB1 twice in rapid succession without moving the mouse.
drag	The phrase <i>drag</i> means to press and hold MB1, move the mouse, and then release MB1 when the pointer is in the desired position.
Ctrl/x	Press the Ctrl key while you press another key.
Menu → Command	The right arrow key indicates an abbreviated instruction for choosing a command from a menu. For example, <i>File → Exit</i> means pull down the File menu, move the pointer to the Exit command, and release MB1.
Courier type	Courier type indicates text that is typed or displayed on the screen. This is most often used for program code examples.
User Input	Boldface type in interactive examples indicates information you enter from the keyboard. For example: A:> SETUP
XX YY and XXn YYn	In DECtalk Software in-line command syntax, XX and YY indicate options and parameters. When more than one choice of options or parameters is allowed, the symbol XXn or YYn with n replaced by a numeral indicates each option or parameter in the symbolic representations, such as [:phoneme XX1 XX2 YY]. Note that the number of characters in the symbolic representation does NOT represent the number of characters allowed in the actual option or parameter name.
DD and DDn	In DECtalk Software in-line command syntax, DD indicates a decimal (base 10) value. When more than one decimal values are allowed, the symbol DDn with n replaced by a numeral represents each allowed value, such as [:volume XX DD1 DD2]. Note that the number of characters in the symbolic representation does NOT represent the number of characters allowed in the actual decimal value.

Conventions used in API calls

Italics Italic text emphasizes important information.

Unless you are instructed otherwise, press **Enter** after you type responses to command prompts.

Chapter 1 DECtalk Software API Calls

This chapter is an alphabetical listing of the DECtalk Software Application Programming Interface (API) calls.

Table 1-1 DECtalk Software API Calls

TextToSpeechAddBuffer
TextToSpeechCloseInMemory
TextToSpeechCloseLang
TextToSpeechCloseLogFile
TextToSpeechCloseWaveOutFile
TextToSpeechEnumLangs
TextToSpeechGetCaps
TextToSpeechGetFeatures
TextToSpeechGetLanguage (not supported)
TextToSpeechGetRate
TextToSpeechGetSpeaker
TextToSpeechGetStatus
TextToSpeechLoadUserDictionary
TextToSpeechOpenInMemory
TextToSpeechOpenLogFile
TextToSpeechOpenWaveOutFile
TextToSpeechPause
TextToSpeechReset
TextToSpeechResume
TextToSpeechReturnBuffer
TextToSpeechSelectLang
TextToSpeechSetLanguage (not supported)
TextToSpeechSetRate

TextToSpeechSetSpeaker
TextToSpeechShutdown
TextToSpeechSpeak
TextToSpeechStartLang
TextToSpeechStartup (Windows only)
TextToSpeechStartup (UNIX only)
TextToSpeechStartupEx
TextToSpeechSync
TextToSpeechTyping
TextToSpeechUnloadUserDictionary
TextToSpeechVersion
TextToSpeechVersionEx

The following formats are not supported in any TextToSpeech...() function call, because of a limitation in the Windows CE operating system:

- WAVE FORMAT 1M16
- WAVE FORMAT 08M08

TextToSpeechAddBuffer

The **TextToSpeechAddBuffer** call supplies a memory buffer to the text-to-speech system. This memory buffer stores speech samples while DECTalk is in the speech-to-memory mode.

Syntax `MMRESULT TextToSpeechAddBuffer` (*LPTTS_HANDLE_T* *phTTS*,
LPTTS_BUFFER_T *pTTSbuffer*)

Parameters	<i>LPTTS_HANDLE_T</i> <i>phTTS</i>	Specifies a text-to-speech handle.
	<i>LPTTS_BUFFER_T</i> <i>pTTSbuffer</i>	Points to a structure containing the memory buffers. Buffers are supplied by the application to be used while in speech-to-memory mode.

Return Value This call returns a value of type MMRESULT. The return value is zero if the call is successful. The return value is one of the following constants:

Constant	Description
MMSYSERR_NOERROR	Normal successful completion (zero).
MMSYSERR_INVALIDPARAM	Invalid parameter.
MMSYSERR_ERROR	Output to memory not enabled or unable to create a system object.
MMSYSERR_INVALIDHANDLE	The text-to-speech handle was invalid.

Comments

The application must call `TextToSpeechOpenInMemory` before calling `TextToSpeechAddBuffer`. The memory buffer is passed using the `TTS_BUFFER_T` structure. The user must allocate the structure and its associated elements (memory buffer, phoneme array, and index mark array). Refer to the *DECtalk Software Programmer's Guide, Speech-To-Memory Mode* section for more information on the `TTS_BUFFER_T` structure and its elements.

The text-to-speech system returns the buffer to the application when the memory buffer, phoneme array, or index mark array is full or when a TTS_FORCE was used in the TextToSpeechSpeak call. Refer to the *DECtalk Software Programmer's Guide, Callback Routines and Window Procedures* section for detail information on passing information back to the calling application.

See Also

Callback Routines and Window Procedures (*DECtalk Software Programmer's Guide*)

Speech-to-Memory Mode (*DECtalk Software Programmer's Guide*)

TextToSpeechOpenInMemory

TextToSpeechReturnBuffer

TextToSpeechStartup (Windows only)

TextToSpeechStartup (UNIX only)

TextToSpeechStartupEx

TextToSpeechCloseInMemory

The **TextToSpeechCloseInMemory** call terminates the speech-to-memory capability and returns to the startup state. The speech samples are then ignored or sent to an audio device, depending on the setting of the `dwDeviceOptions` parameter in the startup function.

Syntax `MMRESULT TextToSpeechCloseInMemory (LPTTS_HANDLE_T phTTS)`

Parameters `LPTTS_HANDLE_T phTTS` Specifies a text-to-speech handle.

Return Value This call returns a value of type `MMRESULT`. The return value is zero if the call is successful. The return value is one of the following constants:

Constant	Description
<code>MMSYSERR_NOERROR</code>	Normal successful completion (zero).
<code>MMSYSERR_ERROR</code>	Output to memory not enabled or unable to create a system object.
<code>MMSYSERR_INVALIDHANDLE</code>	The text-to-speech handle was invalid.

Comments `TextToSpeechOpenInMemory` must be called before calling `TextToSpeechCloseInMemory`.

See Also `TextToSpeechOpenInMemory`

TextToSpeechCloseLang

The **TextToSpeechCloseLang** call closes an instance for an installed language and attempts to unload it from the DECTalk Multi-Language (ML) engine.

Syntax	BOOL TextToSpeechCloseLang	<i>(char *lang)</i>
Parameters	<i>char *lang</i>	Specifies that the language being unloaded is passed as a null-terminated string containing the 2-character language ID.
Return Value	BOOL	Returns TRUE when a language is unloaded, or returns FALSE when the operation cannot be completed or more instances have the thread started.
Comments	<p>Call this function for each thread using the selected language. When a thread returns TRUE, the language is freed and can be uninstalled or upgraded.</p> <p>A return of TRUE may be a bad pass of the lang variable or more instances. If there are more instances, the function frees the current instance and returns a FALSE flag. After calling TextToSpeechCloseLang, assume that the language handle is no longer valid.</p>	

Example

```
BOOL stop_us (void) {
    if (TextToSpeechCloseLang ("us") == FALSE) {
        printf ("Another thread has the language\n") ;
        printf ("still loaded. \n") ;
        return FALSE ;
    }
    printf ("The language has been freed. \n" ) ;
    return TRUE ;
}
```

See Also

TextToSpeechEnumLangs
TextToSpeechSelectLang
TextToSpeechStartLang

TextToSpeechCloseLogFile

The **TextToSpeechCloseLogFile** call closes a log file opened by the **TextToSpeechOpenLogFile** call and returns to the startup state. The speech samples are then ignored or sent to an audio device, depending on the setting of the **dwDeviceOptions** parameter in the startup function.

Syntax **MMRESULT TextToSpeechCloseLogFile** (*LPTTS_HANDLE_T phTTS*)

Parameters *LPTTS_HANDLE_T phTTS* Specifies a text-to-speech handle.

Return Value This call returns a value of type **MMRESULT**. The return value is zero if the call is successful. The return value is one of the following constants.

Constants	Description
MMSYSERR_NOERROR	Normal successful completion (zero).
MMSYSERR_ERROR	Failure to wait for pending speech, unable to close the output file, or no output file is open.
MMSYSERR_INVALIDHANDLE	The text-to-speech handle was invalid.

Comments **TextToSpeechCloseLogFile** closes an open log file, even if it was opened with the **Log** command. The application must have previously called **TextToSpeechOpenLogFile** before calling **TextToSpeechCloseLogFile**.

See Also **TextToSpeechOpenLogFile**

TextToSpeechCloseWaveOutFile

The **TextToSpeechCloseWaveOutFile** call closes a wave file opened by the **TextToSpeechOpenWaveOutFile** call and returns to the startup state. The speech samples are then ignored or sent to an audio device, depending on the setting of the **dwDeviceOptions** parameter in the startup function.

Syntax MMRESULT **TextToSpeechCloseWaveOutFile** (*LPTTS_HANDLE_T phTTS*)

Parameters *LPTTS_HANDLE_T phTTS* Specifies a text-to-speech handle.

Return Value This call returns a value of type MMRESULT. The return value is zero if the call is successful. The return value is one of the following constants:

Constant	Description
MMSYSERR_NOERROR	Normal successful completion (zero).
MMSYSERR_ERROR	Failure to wait for pending speech, unable to update the wave file header, or unable to close the wave file.
MMSYSERR_INVALIDHANDLE	The text-to-speech handle was invalid.

Comments The application must call **TextToSpeechOpenWaveOutFile** before calling **TextToSpeechCloseWaveOutFile**.

See Also **TextToSpeechOpenWaveOutFile**

TextToSpeechEnumLangs

The **TextToSpeechEnumLangs** call retrieves information about what languages are available.

Syntax	DWORD TextToSpeechEnumLangs	<i>(LPLANG_ENUM *langs);</i>
Parameters	<i>LPLANG_ENUM *langs</i>	Specifies a LANG_ENUM struct structure.
Return Value	DWORD	Returns the size of the struct on success, and returns 0 on error. No further error information is available at this time.
Comments	TextToSpeechEnumLangs returns the default language in the registry as the first language of the array of LANG_ENTRY returned by langs.	

Example

```
if(TextToSpeechEnumLangs (&languageINFO) == 0)
{
    MessageBox(NULL, "Unable to allocate Memory", "Error", MB_OK);
    return(-1);
}
...
...
if(languageINFO->MultiLang==FALSE)
    do some none multi lang stuff;
else
    for(i=0;i<languageINFO->Languages;i++)//go through all languages...
    {
        languageINFO->Entries[i].lang_code; //short language name
        languageINFO->Entries[i].lang_name //long language name
    }
```

See Also	TextToSpeechCloseLang TextToSpeechSelectLang TextToSpeechStartLang
-----------------	--

TextToSpeechGetCaps

The **TextToSpeechGetCaps** call lists the current capabilities of the DECtalk Software by filling in the structure of type TTS_CAPS_T. The caller must have space allocated for this structure before calling TextToSpeechGetCaps.

Syntax MMRESULT **TextToSpeechGetCaps** (*LPTTS_CAPS_T lpTTScaps*)

Parameters *LPTTS_CAPS_T lpTTScaps* Specifies a structure of type TTS_CAPS_T. This structure returns the capabilities of the text-to-speech system.

Return Value This call returns a value of type MMRESULT. The return value is zero if the call is successful. The return value is one of the following constants.

Constant	Description
MMSYSERR_NOERROR	Normal successful completion (zero).
MMSYSERR_INVALIDHANDLE	The text-to-speech handle was invalid.
MMSYSERR_ERROR	The pointer to the TTS_CAPS_T structure was invalid.

Comments Information returned in the TTS_CAPS_T structure includes language and proper name pronunciation support, sample rate, minimum and maximum speaking rate, number of predefined speaking voices, character-set supported, and version number. See the DECtalk Software Programmer's Guide for more information on the TTS_CAPS_T structure.

TextToSpeechGetFeatures

The **TextToSpeechGetFeatures** call retrieves information, in the form of a bitmask, about the features of DECTalk.

Syntax `DWORD TextToSpeechGetFeatures (void);`

Parameters *void*

Return Value `DWORD` A bitmask of features supported by DECTalk, maskable to the list supplied in the header file TTSFEAT.H.

Comments Future implementation may involve calling `TextToSpeechSelectLang` to select the language for which to retrieve information. DECTalk ML always sets the ML bit to TRUE, as well as any feature bits returned from DECTalk.

Example

```
BOOL is_dectalk_ml (unsigned int language_handle) {
    unsigned long int feats ;

    TextToSpeechSelectLang (NULL, language_handle) ;
    feats = TextToSpeechGetFeatures() ;
    if (feats & TTS_FEATS_MULTILANG) {;
        printf ("DECTalk ML installed and running. \n" ) ;
        return TRUE ;
    }
    printf ("Multi-language DECTalk not found. \n") ;
    return FALSE ;
}
```

TextToSpeechGetLanguage (not supported)

Warning

The **TextToSpeechGetLanguage** call is not supported for DECTalk Software Version 4.6. Use of this call causes unpredictable operation and application linking errors.

This call has been replaced by TextToSpeechGetCaps.

The **TextToSpeechGetLanguage** call returns the current language.

Syntax	MMRESULT TextToSpeechGetLanguage (<i>LPTTS_HANDLE_T phTTS, LANGUAGE_T *pLanguage</i>)	
Parameters	<i>LPTTS_HANDLE_T phTTS</i>	Specifies a text-to-speech handle.
	<i>LANGUAGE_T *pLanguage</i>	Specifies a language. Refer to the <code>ttsapi.h</code> file for a list of valid languages, e.g. <code>TTS_AMERICAN_ENGLISH</code> .
Return Value	This call returns a value of type MMRESULT. The value is zero if the call is successful. The return value is one of the following constants:	
	Constant	Description
	MMSYSERR_NOERROR	Normal successful completion (zero).
	MMSYSERR_INVALIDHANDLE	The text-to-speech handle was invalid.
See Also	TextToSpeechSetLanguage	

TextToSpeechGetRate

The **TextToSpeechGetRate** call returns the current setting of the speaking rate.

Syntax `MMRESULT TextToSpeechGetRate` *(LPTTS_HANDLE_T phTTS,
LPDWORD pdwRate)*

Parameters *LPTTS_HANDLE_T phTTS* Specifies a text-to-speech handle.
LPDWORD pdwRate Points to a DWORD that is used to return the speaking rate. Valid values range from 75 to 600 words per minute.

Return Value This call returns a value of type MMRESULT. The return value is zero if the call is successful. The return value is one of the following constants:

Constant	Description
MMSYSERR_NOERROR	Normal successful completion (zero).
MMSYSERR_INVALIDHANDLE	The text-to-speech handle was invalid.

Comments The current setting of the speaking rate is returned even if the speaking rate change has not yet occurred. This may occur when the TextToSpeechSetRate call is used without the TextToSpeechSync call. The speaking-rate change occurs on clause boundaries.

See Also TextToSpeechSetRate

TextToSpeechGetSpeaker

The **TextToSpeechGetSpeaker** call returns the value of the identifier for the last voice that has spoken.

Syntax MMRESULT **TextToSpeechGetSpeaker** (*LPTTS_HANDLE_T phTTS, LPSPEAKER_T lpSpeaker*)

Parameters *LPTTS_HANDLE_T phTTS* Specifies a text-to-speech handle.
 LPSPEAKER_T lpSpeaker Points to a DWORD that returns a speaker value from the following list. These symbols are defined in the include file `ttsapi.h`.

Speaker	Description
PAUL	Default (male) voice
HARRY	Full male voice
FRANK	Aged male voice
DENNIS	Male voice
BETTY	Full female voice
URSULA	Aged female voice
WENDY	Whispering female voice
RITA	Female voice
KIT	Child's voice

Return Value This call returns a value of type MMRESULT. The return value is zero if the call is successful. The return value is one of the following constants:

Constant	Description
MMSYSERR_NOERROR	Normal successful completion (zero).
MMSYSERR_INVALIDHANDLE	The text-to-speech handle was invalid.

Comments Note that even after calling `TextToSpeechSetSpeaker`, `TextToSpeechGetSpeaker` returns the value for the previous speaking voice until the new voice actually speaks.

See Also `TextToSpeechSetSpeaker`

TextToSpeechGetStatus

The **TextToSpeechGetStatus** call returns the status of one or more text-to-speech system parameters.

Syntax	MMRESULT TextToSpeechGetStatus	(LPTTS_HANDLE_T phTTS, LPDWORD dwIdentifier[], LPDWORD dwStatus[], DWORD dwNumberOfStatusValues)
---------------	---------------------------------------	---

Parameters		
	<i>LPTTS_HANDLE_T</i> <i>phTTS</i>	Specifies a text-to-speech handle.
	<i>LPDWORD</i> <i>dwIdentifier[]</i>	Specifies an array of values of type <i>DWORD</i> that contains identifiers specifying the status values to return in the <i>dwStatus</i> array. These values can be any of the constants defined in include file <i>tsapi.h</i> .
	<i>LPDWORD</i> <i>dwStatus[]</i>	Specifies an array of type <i>DWORD</i> that contains the status values corresponding to each of the identifiers in the <i>dwIdentifier</i> array.
	<i>DWORD</i> <i>dwNumberOfStatusValues</i>	A <i>DWORD</i> that contains the number of entries to return.

Constant in ttsapi.h	Description
INPUT_CHARACTER_COUNT	Returns a count of characters that the text-to-speech system is currently processing.
STATUS_SPEAKING	The status value is TRUE if audio samples are playing and FALSE if no audio sample is playing.
WAVE_OUT_DEVICE_ID	The current wave output device ID is returned.

Return Value This call returns a value of type MMRESULT. The return value is zero if the call is successful. The return value is one of the following constants:

Constant	Description
MMSYSERR_NOERROR	Normal successful completion (zero).
MMSYSERR_INVALIDPARAM	An invalid parameter was passed.
MMSYSERR_ERROR	Error obtaining status values.
MMSYSERR_INVALIDHANDLE	The text-to-speech handle was invalid.

Comments	The STATUS_SPEAKING status identifier has no meaning if the application is sending speech to a wave file or sending speech to memory.
-----------------	---

TextToSpeechLoadUserDictionary

The **TextToSpeechLoadUserDictionary** call loads a user-defined pronunciation dictionary into the text-to-speech system.

Syntax MMRESULT **TextToSpeechLoadUserDictionary** (*LPTTS_HANDLE_T phTTS, LPSTR pszFileName*)

Parameters *LPTTS_HANDLE_T phTTS* Specifies a text-to-speech handle.
 LPSTR pszFileName Points to a NULL terminated string that specifies the name of the user dictionary file to be loaded.

Return Value This call returns a value of type MMRESULT. The return value is zero if the call is successful. The return value is one of the following constants:

Constant	Description
MMSYSERR_NOERROR	Normal successful completion (zero).
MMSYSERR_INVALIDHANDLE	The text-to-speech handle was invalid.
MMSYSERR_NOMEM	Unable to allocate memory for dictionary.
MMSYSERR_INVALIDPARAM	Dictionary file not found or an invalid dictionary file name.
MMSYSERR_ERROR	Illegal dictionary format or a dictionary is already loaded.

Comments This call loads a dictionary created by the **windict** or **userdict** applet. Any previously loaded user dictionary must be unloaded before loading a new user dictionary. Note that the text-to-speech system will automatically load a user dictionary, *user.dic*, at startup if it exists in the home directory.

See Also Dictionary Functions (*DECtalk Software Programmer's Guide*)
 TextToSpeechUnloadUserDictionary

TextToSpeechOpenInMemory

The **TextToSpeechOpenInMemory** call causes the text-to-speech system to enter into speech-to-memory mode. This mode indicates that the speech samples are to be written into memory buffers rather than sent to an audio device each time **TextToSpeechSpeak** is called. **TextToSpeechAddBuffer** supplies the text-to-speech system with the memory buffers that it needs. The text-to-speech system remains in the speech-to-memory mode until **TextToSpeechCloseInMemory** is called.

Syntax	MMRESULT TextToSpeechOpenInMemory	<i>(LPTTS_HANDLE_T phTTS, DWORD dwFormat)</i>
---------------	--	---

Parameters	<i>LPTTS_HANDLE_T</i> <i>phTTS</i>	Specifies a text-to-speech handle.
	<i>DWORD</i> <i>dwFormat</i>	Specifies an identifier that determines the audio sample format. It is one of the following constants defined in the include files <i>mmsystem.h</i> and <i>ttsapi.h</i> .

Constant	Description
WAVE_FORMAT_1M08	Mono, 8-bit 11.025 kHz sample rate
WAVE_FORMAT_1M16	Mono, 16-bit 11.025 kHz sample rate
WAVE_FORMAT_08M08	Mono, 8-bit μ -law, 8 kHz sample rate

Return Value This call returns a value of type MMRESULT. The return value is zero if the call is successful. The return value is one of the following constants:

Constant	Description
MMSYSERR_NOERROR	Normal successful completion (zero).
MMSYSERR_INVALIDPARAM	An invalid parameter or illegal wave output format was passed.
MMSYSERR_NOMEM	Unable to allocate memory.
MMSYSERR_ERROR	Illegal output state.
MMSYSERR_INVALIDHANDLE	The text-to-speech handle was invalid.

Comment

The text-to-speech system is in the speech-to-memory mode after successfully invoking this function. The memory buffer is passed using the structure `TTS_BUFFER_T`. The user must allocate the structure and its associated elements (memory buffer, phoneme array, and index mark array). Refer to the *DECtalk Software Programmer's Guide, Speech-To-Memory Mode* section for more information on the `TTS_BUFFER_T` structure and its elements.

The text-to-speech system returns the buffer to the application when the memory buffer, phoneme array, or index mark array is full or when a `TTS_FORCE` was used in the `TextToSpeechSpeak` call. Refer to the *DECtalk Software Programmer's Guide, Callback Routines and Window Procedures* section for detail information on passing information back to the calling application.

The startup function must be called to start the text-to-speech system before calling `TextToSpeechOpenInMemory`.

`TextToSpeechOpenInMemory` automatically resumes audio output if the text-to-speech system is in a paused state by a previously issued `TextToSpeechPause` call.

See Also

Callback Routines and Window Procedures (*DECtalk Software Programmer's Guide*)

Speech-to-Memory Mode (*DECtalk Software Programmer's Guide*)

`TextToSpeechAddBuffer`

`TextToSpeechCloseInMemory`

`TextToSpeechPause`

`TextToSpeechReset`

`TextToSpeechReturnBuffer`

`TextToSpeechSpeak`

`TextToSpeechStartup` (Windows only)

`TextToSpeechStartup` (UNIX only)

`TextToSpeechStartupEx`

TextToSpeechOpenLogFile

The **TextToSpeechOpenLogFile** call causes the specified log file to be opened and the text-to-speech system to enter into the log-file mode. This mode indicates that the speech samples are to be written as text, phonemes, or syllables into the log file each time **TextToSpeechSpeak** is called. The phonemes and syllables are written using the arpabet alphabet. The text-to-speech system remains in the log-file mode until **TextToSpeechCloseLogFile** is called.

Syntax	MMRESULT TextToSpeechOpenLogFile	<i>(LPTTS_HANDLE_T phTTS, LPSTR pszFileName, DWORD dwFlags)</i>
---------------	---	---

Parameter	Description
<i>LPTTS_HANDLE_T phTTS</i>	Specifies a text-to-speech handle.
<i>LPSTR pszFileName</i>	Points to a NULL terminated string that specifies the name of the log file to be opened.
<i>DWORD dwFlags</i>	Specifies the type of output. It can contain one or more of the following constants:

Constant	Description
LOG_TEXT	Log text
LOG_PHONEMES	Log phonemes
LOG_SYLLABLES	Log syllable structure

Return Value This call returns a value of type MMRESULT. The return value is zero if the call is successful. The return value is one of the following constants:

Constant	Description
MMSYSERR_NOERROR	Normal successful completion (zero).
MMSYSERR_INVALIDPARAM	An invalid parameter was passed.
MMSYSERR_NOMEM	Unable to allocate memory.
MMSYSERR_ALLOCATED	A phoneme file is already open.
MMSYSERR_ERROR	Unable to open the output file.
MMSYSERR_INVALIDHANDLE	The text-to-speech handle was invalid.

Comments

If more than one of the `dwFlags` are passed, then the logged output is mixed in an unpredictable fashion. If a log file is open already, this call returns an error. The **Log** command has no effect when a log file is open already.

The startup function must be called to start the text-to-speech system before calling `TextToSpeechOpenLogFile`.

`TextToSpeechOpenLogFile` automatically resumes audio output if the text-to-speech system is in a paused state by a previously issued `TextToSpeechPause` call.

See Also

Log-File Mode (*DECtalk Software Programmer's Guide*)

`TextToSpeechCloseLogFile`

`TextToSpeechPause`

`TextToSpeechReset`

`TextToSpeechSpeak`

`TextToSpeechStartup` (Windows only)

`TextToSpeechStartup` (UNIX only)

`TextToSpeechStartupEx`

TextToSpeechOpenWaveOutFile

The **TextToSpeechOpenWaveOutFile** call causes the specified wave file to be opened and the text-to-speech system to enter into wave-file mode. This mode indicates that the speech samples are to be written in wave format into the wave file each time **TextToSpeechSpeak** is called. The text-to-speech system remains in the wave-file mode until **TextToSpeechCloseWaveOutFile** is called.

Syntax **MMRESULT TextToSpeechOpenWaveOutFile** (*LPTTS_HANDLE_T phTTS*,
 LPSTR pszFileName,
 DWORD dwFormat)

Parameters	<i>LPTTS_HANDLE_T phTTS</i>	Specifies a text-to-speech handle.
	<i>LPSTR pszFileName</i>	Specifies a pointer to a file name.
	<i>DWORD dwFormat</i>	Determines the audio sample format. It can be one of the following constants that are defined in include files <i>mmsystem.h</i> and <i>ttsapi.h</i> :
Constant		Description
WAVE_FORMAT_1M08		Mono, 8-bit 11.025 kHz sample rate
WAVE_FORMAT_1M16		Mono, 16-bit 11.025 kHz sample rate
WAVE_FORMAT_08M08		Mono, 8-bit μ -law, 8 kHz sample rate

Return Value This call returns a value of type **MMRESULT**. The return value is zero if the call is successful. The return value is one of the following constants:

Constant	Description
MMSYSERR_NOERROR	Normal successful completion (zero).
MMSYSERR_INVALIDPARAM	An invalid parameter or an illegal wave output format was passed.
MMSYSERR_NOMEM	Memory allocation error.
MMSYSERR_ALLOCATED	A wave file is already open.
MMSYSERR_ERROR	Unable to open the wave file or unable to write to the wave file.
MMSYSERR_INVALIDHANDLE	The text-to-speech handle was invalid.

Comments This call automatically resumes audio output if the text-to-speech system is in a paused state by a previously issued **TextToSpeechPause** call.

The startup function must be called to start the text-to-speech system before calling `TextToSpeechOpenWaveOutFile`.

See Also

Wave-File Mode (*DECTalk Software Programmer's Guide*)

`TextToSpeechCloseWaveOutFile`

`TextToSpeechPause`

`TextToSpeechReset`

`TextToSpeechSpeak`

`TextToSpeechStartup` (Windows only)

`TextToSpeechStartup` (UNIX only)

`TextToSpeechStartupEx`

TextToSpeechPause

The **TextToSpeechPause** call pauses text-to-speech audio output.

Syntax `MMRESULT TextToSpeechPause` (*LPTTS_HANDLE_T phTTS*)

Parameters *LPTTS_HANDLE_T phTTS* Specifies a text-to-speech handle.

Return Value This call returns a value of type `MMRESULT`. The return value is zero if the call is successful. The return value is one of the following constants:

Constant	Description
<code>MMSYSERR_NOERROR</code>	Normal successful completion (zero).
<code>MMSYSERR_INVALIDHANDLE</code>	The specified device handle is invalid. The system is not speaking or the text-to-speech handle is invalid.

Comments

This call affects only the audio output and has no effect when writing log files, wave files, or when using the speech-to-memory capability of the text-to-speech system.

If the text-to-speech system owns the audio device (i.e. `OWN_AUDIO_DEVICE` was specified in the startup function), then the text-to-speech system remains paused until `TextToSpeechResume`, `TextToSpeechSync`, `TextToSpeechOpenInMemory`, `TextToSpeechOpenLogFile`, or `TextToSpeechOpenWaveOutFile` is called.

If the text-to-speech system does not own the audio device (i.e. `OWN_AUDIO_DEVICE` was NOT specified in the startup function) and `TextToSpeechPause` is called while the system is speaking, the text-to-speech system remains paused until the system has completed speaking.

In this case, the wave output device is released when `TextToSpeechReset` is called. It will also be released if `TextToSpeechSync()`, `TextToSpeechOpenInMemory()`, `TextToSpeechOpenLogFile`, or `TextToSpeechOpenWaveOutFile` is called AND the system has completed speaking.

Note that `TextToSpeechPause` will NOT resume audio output if the text-to-speech system is paused by `TextToSpeechPause`.

See Also

TextToSpeechOpenInMemory
TextToSpeechOpenLogFile
TextToSpeechOpenWaveOutFile
TextToSpeechReset
TextToSpeechResume
TextToSpeechSpeak
TextToSpeechSync

TextToSpeechReset

The **TextToSpeechReset** call flushes all previously queued text from the text-to-speech system and stops any audio output.

Syntax `MMRESULT TextToSpeechReset` *(LPTTS_HANDLE_T phTTS, BOOL bReset)*

Parameters *LPTTS_HANDLE_T phTTS* Specifies a text-to-speech handle.
 BOOL bReset Specifies one of the following Boolean values:

Value	Description
FALSE	Preserves the current mode of the text-to-speech system.
TRUE	The text-to-speech system is returned to the startup state and any open text-to-speech files are closed. The one exception is that this call will NOT resume the text-to-speech system if it has been paused by the TextToSpeechPause call.

Return Value The TextToSpeechReset call returns a value of type MMRESULT. The return value is zero if the call is successful. The return value is one of the following constants:

Constant	Description
MMSYSERR_NOERROR	Normal successful completion (zero).
MMSYSERR_NOMEM	Unable to allocate memory.
MMSYSERR_ERROR	Unable to flush the system.
MMSYSERR_INVALIDHANDLE	The text-to-speech handle was invalid.

Comments TextToSpeechReset flushes all previously queued text and stops all audio output. If the TextToSpeechOpenInMemory call has enabled writing the speech samples to memory, then all queued memory buffers are returned to the calling application. If the bReset flag is on and the text-to-speech system is in one of its special modes (log-file, wave-file or speech-to-memory mode), then all files will be closed and the text-to-speech system is returned to the startup state.

TextToSpeechReset should be called before calling TextToSpeechCloseInMemory. Failing to do this in a situation where the synthesizer is busy may result in a deadlock.

See Also

Special Text-To-Speech Modes (DECtalk Software Programmer's Guide)

TextToSpeechOpenInMemory

TextToSpeechOpenLogFile

TextToSpeechOpenWaveOutFile

TextToSpeechPause

TextToSpeechResume

The **TextToSpeechResume** call resumes text-to-speech output after it was paused by calling `TextToSpeechPause`.

Syntax `MMRESULT TextToSpeechResume` (*LPTTS_HANDLE_T phTTS*)

Parameters *LPTTS_HANDLE_T phTTS* Specifies a text-to-speech handle.

Return Value This call returns a value of type `MMRESULT`. The return value is zero if the call is successful. The return value is one of the following constants:

Constant	Description
<code>MMSYSERR_NOERROR</code>	Normal successful completion (zero).
<code>MMSYSERR_INVALIDHANDLE</code>	The system was not paused or the text-to-speech handle was invalid.

Comments This call affects only the audio output and has no effect when writing log files, writing wave files, or when writing speech samples to memory.

See Also `TextToSpeechPause`

TextToSpeechReturnBuffer

The **TextToSpeechReturnBuffer** call returns the current buffer when an application is using the speech-to-memory capability. The buffer can be empty or partially full when it is returned. The dwBufferLength element of the TTS_BUFFER_T structure contains the number of samples in the buffer. If no buffer is available, then a NULL pointer is returned in ppTTSbuffer.

Syntax

MMRESULT **TextToSpeechReturnBuffer** (LPTTS_HANDLE_T phTTS,
LPTTS_BUFFER_T *ppTTSbuffer)

Parameters

LPTTS_HANDLE_T phTTS

Specifies a text-to-speech handle.

LPTTS_BUFFER_T *ppTTSbuffer

Points to a structure containing the memory buffers. Buffers were supplied by the application to be used while in speech-to-memory mode.

Return Value

This call returns a value of type MMRESULT. The return value is zero if the call is successful. The return value is one of the following constants:

Constant	Description
MMSYSERR_NOERROR	Normal successful completion (zero).
MMSYSERR_INVALIDPARAM	Invalid parameter.
MMSYSERR_ERROR	Output to memory not enabled or unable to create a system object.
MMSYSERR_INVALIDHANDLE	The text-to-speech handle was invalid.

Comments

Most applications do not require this call because buffers are automatically returned when filled or when a TTS_FORCE flag is passed in the TextToSpeechSpeak call. The TextToSpeechReturnBuffer call is provided so an application can return a buffer before it is filled and, therefore, obtain more speech samples immediately. See the DECTalk Software Programmer's Guide for more information on the TTS_BUFFER_T structure.

TextToSpeechAddBuffer must be called before calling TextToSpeechReturnBuffer.

See Also

TextToSpeechAddBuffer

TextToSpeechSelectLang

The **TextToSpeechSelectLang** call selects a loaded language for a program thread.

Syntax	BOOL TextToSpeechSelectLang	<i>(LPTTS_HANDLE_T reserved, unsigned int lang)</i>
Parameters	<i>LPTTS_HANDLE_T reserved</i> <i>unsigned int lang</i>	Reserved; must be NULL. Specifies the language handle returned from TextToSpeechStartLang.
Return Value	BOOL	Returns TRUE on success, and returns FALSE on error.
Comments	None.	

Example

```
BOOL select_us (unsigned int us_handle) {  
    if (TextToSpeechSelectLang (NULL, us_handle) == FALSE) {  
        printf ("Select language failed. \n") ;  
        return FALSE ;  
    }  
    return TRUE ;  
}
```

See Also	TextToSpeechCloseLang TextToSpeechEnumLangs TextToSpeechStartLang
-----------------	---

TextToSpeechSetLanguage (not supported)

Warning

The **TextToSpeechSetLanguage** call is not supported for DECTalk Software version 4.6. Use of this call causes unpredictable operation and application linking errors.

For multi-language programming, see **TextToSpeechStartLang** to check for an installed language and to load that language into the DECTalk Multi-Language (ML) engine, and see **TextToSpeechSelectLang** to select a loaded language for a program thread.

The **TextToSpeechSetLanguage** call selects a language for the text-to-speech system to use as the default language.

Syntax	MMRESULT TextToSpeechSetLanguage (<i>LPTTS_HANDLE_T phTTS, LANGUAGE_T Language</i>)	
Parameters	<i>LPTTS_HANDLE_T phTTS</i>	Specifies a text-to-speech handle.
	<i>LANGUAGE_T Language</i>	Specifies a language. Refer to the ttsapi.h file for a list of valid languages, e.g. TTS_AMERICAN_ENGLISH.
Return Value	This call returns a value of type MMRESULT. The value is zero if the call is successful. The return value is one of the following constants:	
	Constant MMSYSERR_NOERROR MMSYSERR_INVALIDPARAM MMSYSERR_INVALIDHANDLE	Description Normal successful completion (zero). An invalid parameter was passed. The text-to-speech handle was invalid.
See Also	TextToSpeechGetLanguage	

TextToSpeechSetRate

The **TextToSpeechSetRate** call sets the text-to-speech speaking rate.

Syntax	MMRESULT TextToSpeechSetRate (<i>LPTTS_HANDLE_T</i> <i>phTTS</i> , <i>DWORD</i> <i>dwRate</i>)
---------------	--

Parameters	<i>LPTTS_HANDLE_T</i> <i>phTTS</i>	Specifies a text-to-speech handle.
	<i>DWORD</i> <i>dwRate</i>	Sets the speaking rate. Valid values range from 75 to 600 words per minute.

Return Value This call returns a value of type MMRESULT. The return value is zero if the call is successful. The return value is one of the following constants:

Constant	Description
MMSYSERR_NOERROR	Normal successful completion (zero).
MMSYSERR_INVALIDPARAM	An invalid parameter was passed.
MMSYSERR_INVALIDHANDLE	The text-to-speech handle was invalid.

Comments	The speaking rate change is not effective until the next phrase boundary. All the queued audio encountered before the phrase boundary is unaffected.
-----------------	--

See Also [TextToSpeechGetRate](#)

TextToSpeechSetSpeaker

The **TextToSpeechSetSpeaker** call sets the voice of the speaker the text-to-speech system will use.

Syntax

MMRESULT **TextToSpeechSetSpeaker** (*LPTTS_HANDLE_T phTTS*,
SPEAKER_T Speaker)

Parameters

LPTTS_HANDLE_T phTTS

Specifies a text-to-speech handle.

SPEAKER_T Speaker

Selects a speaker from the following list. These values are defined in include file ttsapi.h.

Speaker	Description
PAUL	Default (male) voice
HARRY	Full male voice
FRANK	Aged male voice
DENNIS	Male voice
BETTY	Full female voice
URSULA	Aged female voice
WENDY	Whispering female voice
RITA	Female voice
KIT	Child's voice

Return Value

This function returns a value of type MMRESULT. The return value is zero if the function is successful. The return value is one of the following constants:

Constant	Description
MMSYSERR_NOERROR	Normal successful completion (zero).
MMSYSERR_INVALIDPARAM	An invalid parameter was passed.
MMSYSERR_INVALIDHANDLE	The text-to-speech handle was invalid.

Comments

The change in speaking voice is not effective until the next phrase boundary. All queued audio encountered before the phrase boundary is unaffected.

See Also

TextToSpeechGetSpeaker

TextToSpeechShutdown

The **TextToSpeechShutdown** call shuts down the text-to-speech system and frees all system resources used by the text-to-speech system.

Syntax **MMRESULT TextToSpeechShutdown** (*LPTTS_HANDLE_T phTTS*)

Parameters *LPTTS_HANDLE_T phTTS* Specifies a text-to-speech handle.

Return Value This call returns a value of type **MMRESULT**. The return value is zero if the call is successful. The return value is one of the following constants:

Constant	Description
MMSYSERR_NOERROR	Normal successful completion (zero).
MMSYSERR_INVALIDHANDLE	The text-to-speech handle was invalid.

Comments TextToSpeechShutdown is called to close an application. Any user-defined dictionaries that were previously loaded are unloaded. All previously queued text is discarded, and the text-to-speech system immediately stops speaking.

See Also TextToSpeechStartup (Windows only)
TextToSpeechStartup (UNIX only)
TextToSpeechStartupEx

TextToSpeechSpeak

The **TextToSpeechSpeak** call queues a null-terminated string to the text-to-speech system. While in startup state, speech samples are routed to the audio device or ignored, depending on whether the `DO_NOT_USE_AUDIO_DEVICE` flag is set in the `dwDeviceOptions` parameter of the startup function. If the text-to-speech system is in one of its special modes (wave-file, log-file, or speech-to-memory modes), the speech samples are handled accordingly.

Syntax	MMRESULT TextToSpeechSpeak	(<i>LPTTS_HANDLE_T</i> <i>phTTS</i> , <i>LPSTR</i> <i>pszTextString</i> , <i>DWORD</i> <i>dwFlags</i>)
Parameters	<i>LPTTS_HANDLE_T</i> <i>phTTS</i>	Specifies a text-to-speech handle.
	<i>LPSTR</i> <i>pszTextString</i>	Points to a null terminated string of characters to be queued.
	<i>DWORD</i> <i>dwFlags</i>	Specifies whether the text is to be pushed through the text-to-speech system even if it does NOT end on a clause boundary. It can be set to one of the following constants defined in include file <code>ttsapi.h</code> :
Constant		Description
TTS_NORMAL		Insert characters in the text-to-speech queue.
TTS_FORCE		Insert characters in the text-to-speech queue and force all text to be output even if the text stream does NOT end on a clause boundary.

Return Values	This call returns a value of type <code>MMRESULT</code> . The return value is zero if the call is successful. The return value is one of the following constants:	
Constant		Description
MMSYSERR_NOERROR		Normal successful completion (zero).
MMSYSERR_NOMEM		Unable to allocate memory.
MMSYSERR_INVALIDHANDLE		The text-to-speech handle was invalid.

Comments

The speaker, speaking rate, and volume also can be changed in the text string by inserting commands as shown in the following example:

[:name paul] I am Paul. **[:nb]** I am Betty. **[:volume set 50]** The volume has been set to 50% of the maximum level. **[:rate 120]** I am speaking at 120 words per minute.

See Also

Special Text-To-Speech Modes (*DECtalk Software Programmer's Guide*)

TextToSpeechOpenInMemory

TextToSpeechOpenLogFile

TextToSpeechOpenWaveOutFile

TextToSpeechStartup (Windows only)

TextToSpeechStartup (UNIX only)

TextToSpeechStartupEx

TextToSpeechStartLang

The **TextToSpeechStartLang** call checks for an installed language and loads it into the DECTalk ML engine.

Syntax	unsigned int TextToSpeechStartLang	(<i>char *lang</i>)
Parameters	<i>char *lang</i>	Language being selected and passed as a null-terminated string containing the two character language ID.
Return Value	unsigned int	<p>This call returns a handle to the loaded language on success or sets the bit TTS_LANG_ERROR on failure.</p> <p>If the TTS_LANG_ERROR bit is set, the return can equal one of two values:</p> <ul style="list-style-type: none">• TTS_NOT_SUPPORTED – the application is not running DECTalk ML• TTS_NOT_AVAILABLE – the language selected is not installed
Comments	TextToSpeechStartLang must be called before a language can be selected and opened in a multi-language application.	

Example

```
BOOL start_us (void) {
    unsigned int handle ;

    handle = TextToSpeechStartLang ("us") ;
    if (handle & TTS_LANG_ERROR) {
        if (handle == TTS_NOT_SUPPORTED)
            printf ("DECTalk ML not found. \n") ;
        else if (handle == TTS_NOT_AVAILABLE)
            printf ("English is not currently installed \n")
;
        else
            printf ("An unknown error has occurred \n") ;
        return FALSE ;
    }
    return TRUE ;
}
```

See Also	TextToSpeechCloseLang
	TextToSpeechEnumLangs
	TextToSpeechSelectLang

TextToSpeechStartup (Windows only)

The **TextToSpeechStartup** call for Windows 95 and Windows NT initializes the text-to-speech system, defines the window procedure, checks for valid licenses, and loads the main and user pronunciation dictionaries. This function returns a value of type **MMRESULT**. This value is zero if initialization was successful.

A single process can run multiple instances of DECTalk.

Syntax	MMRESULT TextToSpeechStartup	<i>(HWND hWnd, LPTTS_HANDLE_T *phTTS, UINT uiDeviceNumber, DWORD dwDeviceOptions)</i>
Parameters	<i>HWND hWnd</i>	Specifies a handle used to send messages back to the window procedure. The window handle is used by DECTalk Software to inform the application when the buffer is full (if DECTalk Software in-memory calls are being used) or when TextToSpeechSpeak encounters an index mark. A value of NULL is passed if no window handle is desired.
	<i>LPTTS_HANDLE_T *phTTS</i>	Specifies a text-to-speech handle.
	<i>UINT uiDeviceNumber</i>	Specifies the device number of the wave output device. A value of WAVE_MAPPER can be used to select the first available device.
	<i>DWORD dwDeviceOptions</i>	Specifies how the wave output device is managed. It can be a combination of the following constants defined in include file ttsapi.h:

Constant	Description
OWN_AUDIO_DEVICE	The wave output device is open. No other process can allocate the wave output device until TextToSpeechShutdown is called. If OWN_AUDIO_DEVICE is NOT specified, the wave output device is opened after audio is queued by TextToSpeechSpeak. The wave output device is released when the text-to-speech system has completed speaking.
REPORT_OPEN_ERROR	If an attempt is made to open the wave output device while another process owns it, then an error message is sent to the calling application.
DO_NOT_USE_AUDIO_DEVICE	When this flag is set, speech samples are ignored until one of the text-to-speech special modes is set. The text-to-speech special modes can be used to write the speech

samples to a wave file, memory buffers, or log files. No error is returned if a wave output device is not present.

Return Values This call returns a value of type MMRESULT. The return value is zero if the call is successful. The return value is one of the following constants:

Constant	Description
MMSYSERR_NOERROR	Normal successful completion (zero).
MMSYSERR_NODRIVER	No wave output device present.
MMSYSERR_NOMEM	Memory allocation error.
MMSYSERR_ERROR	DECTalk dictionary not found.
MMSYSERR_BADDEVICE_ID	Device ID out of range.
MMSYSERR_ALLOCATED	License exists but no more units available.
WAVERR_BADFORMAT	Wave output device does not support request format.

Comments DECTalk Software sends a post message to the calling application if a Windows procedure is defined and one of the following events occurs:

- A buffer is filled while DECTalk Software is in speech-to-memory mode
- An error occurs
- An index mark is encountered

The default parameters are:

- *Language*: American English.
- *Speaking rate*: 200 words per minute.
- *Speaker*: Paul

If you build an application for the static version of DECTalk Software, you must include the winmm.lib file in the list of input files for the linker.

See Also: Callback Routines and Window Procedures (*DECTalk Software Programmer's Guide*)

Dictionary Functions (Windows only)

TextToSpeechLoadUserDictionary

TextToSpeechOpenInMemory

TextToSpeechOpenLogFile

TextToSpeechOpenWaveOutFile

TextToSpeechShutdown

TextToSpeechSpeak

TextToSpeechStartupEx

TextToSpeechUnloadUserDictionary

TextToSpeechStartup (UNIX only)

The **TextToSpeechStartup** call for Tru64 UNIX initializes the text-to-speech system, defines the callback routine, checks for valid licenses, and loads the main and user pronunciation dictionaries. This call returns a value of type MMRESULT. This value is zero if initialization was successful.

A single process can run multiple instances of DECtalk.

Syntax	MMRESULT TextToSpeechStartup	<i>(LPTTS_HANDLE_T *phTTS, UINT uiDeviceNumber, DWORD dwDeviceOptions, VOID (*DtCallbackRoutine)(), LONG dwCallbackParameter)</i>
Parameters	<i>LPTTS_HANDLE_T *phTTS</i>	Specifies a text-to-speech handle.
	<i>UINT uiDeviceNumber</i>	Specifies a device number of the wave output device. A value of WAVE_MAPPER can be used to select the first available device.
	<i>DWORD dwDeviceOptions</i>	Specifies how the wave output device is managed. It can be a combination of the constants defined in include file ttsapi.h.
	<i>VOID (*DtCallbackRoutine)()</i>	This parameter is used to specify a callback routine. The callback routine is used by DECtalk Software to inform the application when the buffer is full (if DECtalk Software in-memory calls are being used) or when the TextToSpeechSpeak() function encounters an index mark. Refer to the <i>DECtalk Software Programmer's Guide, Callback Routines and Window Procedures</i> section for information about the argument list for the callback routine. A value of NULL is passed if no callback routine is desired.
	<i>LONG dwCallbackParameter</i>	This is a pointer to a user-specified parameter. It is used to pass parameters in the callback routine. A value of NULL should be passed if no user-specified parameters are desired.

Constant in ttsapi.h	Description
OWN_AUDIO_DEVICE	The wave output device is opened. No other process can allocate the wave output device until TextToSpeechShutdown is called. If OWN_AUDIO_DEVICE is NOT specified, the wave output device is opened after audio is queued by the TextToSpeechSpeak() function. The wave output device is released when the text-to-speech system has completed speaking.
REPORT_OPEN_ERROR	If an attempt is made to open the wave output device while another process owns it, then a callback is sent to the callback routine that was passed to TextToSpeechStartup.
DO_NOT_USE_AUDIO_DEVICE	When this flag is set, speech samples are ignored until one of the text-to-speech special modes is set. The text-to-speech special modes can be used to write the speech samples to a wave file, memory buffers, or log files. No error is returned if a wave output device is not present.

Return Values This call returns a value of type MMRESULT. The return value is zero if the call is successful. The return value is one of the following constants:

Constant	Description
MMSYSERR_NOERROR	Normal successful completion (zero).
MMSYSERR_NODRIVER	No wave output device present.
MMSYSERR_NOMEM	Memory allocation error.
MMSYSERR_ERROR	DECtalk dictionary not found.
MMSYSERR_BADDEVICE_ID	Device ID out of range.
MMSYSERR_ALLOCATED	License exists but no more units available.
MMSYSERR_NOTENABLED	License does not exist.
WAVERR_BADFORMAT	Wave output device does not support request format.

Comments DECtalk Software sends a post message to the calling application if a Windows procedure is defined and one of the following events occurs:

- A buffer is filled while DECtalk Software is in speech-to-memory mode
- An error occurs
- An index mark is encountered

The default parameters are:

- *Language:* American English.
- *Speaking rate:* 200 words per minute.

See Also

- *Speaker*: Paul

Callback Routines and Window Procedures (*DECTalk Software Programmer's Guide*)

Dictionary Functions (UNIX only)

TextToSpeechLoadUserDictionary

TextToSpeechOpenInMemory

TextToSpeechOpenLogFile

TextToSpeechOpenWaveOutFile

TextToSpeechShutdown

TextToSpeechSpeak

TextToSpeechStartupEx

TextToSpeechUnloadUserDictionary

TextToSpeechStartupEx

The **TextToSpeechStartupEx** call initializes the text-to-speech system, defines the callback procedure, checks for valid licenses, and loads the main and user pronunciation dictionaries. This call returns a value of type MMRESULT. This value is zero if initialization was successful.

A single process can run multiple instances of DECtalk.

Syntax	MMRESULT TextToSpeechStartupEx	(LPTTS_HANDLE_T *phTTS, UINT uiDeviceNumber, DWORD dwDeviceOptions, VOID (*DtCallbackRoutine)(), LONG dwCallbackParameter)
Parameters	LPTTS_HANDLE_T *phTTS	Specifies a text-to-speech handle.
	UINT uiDeviceNumber	Specifies the device number of the wave output device. A value of WAVE_MAPPER can be used to select the first available device.
	DWORD dwDeviceOptions	Specifies how the wave output device is managed. It can be a combination of the constants defined in include file ttsapi.h.
	VOID (*DtCallbackRoutine)()	This parameter is used to specify a callback routine. The callback routine is used by DECtalk Software to inform the application when the buffer is full (if DECtalk Software in-memory calls are being used) or when the TextToSpeechSpeak call encounters an index mark. Refer to the <i>DECtalk Software Programmer's Guide, Chapter 3, Introduction to the DECtalk Software API, Callback Routines and Window Procedures</i> for information about the argument list for the callback routine.
		A value of NULL is passed if no callback routine is desired.
	LONG dwCallbackParameter	This is a pointer to a user-specified parameter. It is used to pass parameters in the callback routine.
		A value of NULL should be passed if no user-specified parameters are desired.

Constant in ttsapi.h	Description
OWN_AUDIO_DEVICE	The wave output device is open. No other process can allocate the wave output device until TextToSpeechShutdown is called. If OWN_AUDIO_DEVICE is NOT specified, the wave output device is opened after audio is queued by the TextToSpeechSpeak call. The wave output device is released when the text-to-speech system has completed speaking.
REPORT_OPEN_ERROR	If an attempt is made to open the wave output device while another process owns it, a callback is sent to the callback routine that was passed to TextToSpeechStartupEx.
DO_NOT_USE_AUDIO_DEVICE	When this flag is set, speech samples are ignored until one of the text-to-speech special modes is set. The text-to-speech special modes can be used to write the speech samples to a wave file, memory buffers, or log files. No error is returned if a wave output device is not present.

Return Values This call returns a value of type MMRESULT. The return value is zero if the call is successful. The return value is one of the following constants:

Constant	Description
MMSYSERR_NOERROR	Normal successful completion (zero).
MMSYSERR_NODRIVER	No wave output device present.
MMSYSERR_NOMEM	Memory allocation error.
MMSYSERR_ERROR	DECTalk dictionary not found.
MMSYSERR_BADDEVICE_ID	Device ID out of range.
MMSYSERR_ALLOCATED	License exists but no more units available.
MMSYSERR_NOTENABLED	License does not exist. (UNIX only)
WAVERR_BADFORMAT	Wave output device does not support request format.

Comments DECTalk Software sends a post message to the calling application if a Windows procedure is defined and one of the following events occurs:

- A buffer is filled while DECTalk Software is in speech-to-memory mode
- An error occurs
- An index mark is encountered

The default parameters are:

- *Language*: American English.
- *Speaking rate*: 200 words per minute.

- Speaker: Paul

If you build an application for the static version of DECtalk, you must include the winmm.lib file in the list of input files for the linker.

See Also

Callback Routines and Window Procedures (*DECTalk Software Programmer's Guide*)

Dictionary Functions (UNIX only)

Dictionary Functions (Windows only)

TextToSpeechLoadUserDictionary

TextToSpeechOpenInMemory

TextToSpeechOpenLogFile

TextToSpeechOpenWaveOutFile

TextToSpeechShutdown

TextToSpeechSpeak

TextToSpeechStartup (Windows only)

TextToSpeechStartup (UNIX only)

TextToSpeechUnloadUserDictionary

Note

Callback routines should not contain calls to any TextToSpeech...() functions. If callback routines contain TextToSpeech...() functions, an application crash may occur in the application calling DECTalk Software.

TextToSpeechSync

The **TextToSpeechSync** call blocks until all previously queued text is processed.

Syntax MMRESULT **TextToSpeechSync** (*LPTTS_HANDLE_T phTTS*)

Parameters *LPTTS_HANDLE_T phTTS* Specifies a text-to-speech handle.

Return Values This call returns a value of type MMRESULT. The return value is zero if the call is successful. The return value is one of the following constants

Constant	Description
MMSYSERR_NOERROR	Normal successful completion (zero).
MMSYSERR_ERROR	Unable to complete queued text.
MMSYSERR_INVALIDHANDLE	The text-to-speech handle was invalid.

Comments This call automatically resumes audio output if the text-to-speech system is in a paused state by a previously issued TextToSpeechPause call.

See Also TextToSpeechPause

TextToSpeechTyping

The **TextToSpeechTyping** call speaks a single letter as quickly as possible, aborting any previously queued speech. This is somewhat slower if TextToSpeechSpeak has been called since the last TextToSpeechTyping or TextToSpeechReset call.

This call is primarily useful with the Access32 versions of DECtalk Software. This call exists in non-Access32 versions, but is not fast.

Syntax

```
void TextToSpeechTyping((LPTTS_HANDLE_T phTTS,  
char cLetter))
```

Parameters	<i>LPTTS_HANDLE_T</i> <i>phTTS</i>	Specifies a text-to-speech handle.
	<i>char</i> <i>cLetter</i>	The letter to speak.

Return Value None.

Comments	This function should be called only when the application is synthesizing directly to an audio device (not to memory or to a file).
-----------------	--

TextToSpeechUnloadUserDictionary

The **TextToSpeechUnloadUserDictionary** call unloads a user dictionary. You must unload any previously loaded dictionary before you can load a new one. That is, only one user dictionary can be loaded at a time.

Syntax MMRESULT **TextToSpeechUnloadUserDictionary** (*LPTTS_HANDLE_T phTTS*)

Parameters *LPTTS_HANDLE_T phTTS* Specifies a text-to-speech handle.

Return Values This call returns a value of type MMRESULT. The return value is zero if the call is successful. The return value is one of the following constants:

Constant	Description
MMSYSERR_NOERROR	Normal successful completion (zero).
MMSYSERR_INVALIDHANDLE	The text-to-speech handle was invalid.

Comments A user dictionary is created using the User Dictionary Build tool.

If you build an application for the static version of DECTalk, you must include the `winmm.lib` file in the list of input files for the linker.

See Also Dictionary Functions (UNIX only)

Dictionary Functions (Windows only)

TextToSpeechLoadUserDictionary

TextToSpeechVersion

The **TextToSpeechVersion** call requests version information from DECTalk Software and enables an application to test for DAPI compatibility. Upon return, a text buffer can be set. A numerically encoded version number is returned as an unsigned long integer.

Syntax ULONG **TextToSpeechVersion** (*LPSTR *VersionStr*)

Parameters *LPSTR *VersionStr* The address of a pointer to an array of characters, or NULL for no text information.

Return Values This call returns an unsigned long integer (ULONG) encoded with both the DAPI build version and the DECTalk version number. The encoding is as follows:

Version	Bits Used
DECTalk Major Version	Bits 31-24
DECTalk Minor Version	Bits 23-16
DAPI Major Version	Bits 15-8
DAPI Minor Version	Bits 7-0

If DAPI Major Version is not the same as the DAPI Major Version the application was compiled with, the DAPI is no longer compatible and the application may easily crash during further calls into the DAPI.

If DAPI Minor Version is lower than the version of the DAPI the application was compiled with, some features that are expected may not be functional or present in the DAPI.

For safety, users should make the following check:

```
if (DAPI_Major_Version!=Build_Major_Version) Error();  
if (DAPI_Minor_Version<Build_Minor_Version) Error();  
success();
```

This allows your application to catch a majority of incompatibility bugs, which could arise from DECTalk version mismatching.

TextToSpeechVersionEx

The **TextToSpeechVersionEx** call returns information about the currently running version of DECTalk Software.

Syntax	ULONG TextToSpeechVersionEx	(LPVERSION_INFO *ver)
Parameters	LPVERSION_INFO *ver	The address of a pointer to an array of characters with version information. The VERSION_INFO structure is as follows: <div><div>DWORD</div><div>StructSize;</div><div>DWORD</div><div>StructVersion;</div><div>WORD</div><div>DLLVersion;</div><div>WORD</div><div>DTalkVersion;</div><div>LPSTR</div><div>VerString;</div><div>LPSTR</div><div>Language;</div><div>DWORD</div><div>Features;</div></div>

Return Value This call returns an unsigned long integer (ULONG) with the size of the VERSION_INFO structure. The return value is zero if the call is not successful. No other error information is available.

Example

```
BOOL IsDECTalkMLInstalled(void) {
    LPVERSION_INFO verinfo;

    TextToSpeechVersionEx(&verinfo);
    if (ver->Features & TTS_FEATS_MULTILANG) return TRUE;
    return FALSE;
}
```


Chapter 2 — *DECtalk Software In-Line Commands*

In-Line Commands: Overview

DECtalk Software includes in-line commands. In this documentation, in-line commands are referred to as commands. You can use these commands to perform simple operations, such as changing the speaking rate or speaking voice while DECtalk Software is speaking. Commands are inserted directly into the ASCII text that is sent to the synthesizer. Table 2-1 lists the DECtalk Software in-line commands and their associated functions.

With phoneme interpretation, it is possible to control intonation and stress and to create special effects, such as singing. These symbols and special effects can be added into the ASCII text stream. See the description of the **Phoneme Interpretation** command for more information.

When you use several commands together, they may interact with each other and affect the output. If incorrect syntax is used in a command, the right bracket (]) is ignored, because it might be considered part of the illegal string. To avoid this situation, insert an extra right bracket (]) in the command and use the Error command to enable the speaking of errors.

Note

Unique abbreviations of command names and option names work reliably. However, only 4-character abbreviations will be supported in future releases. A character abbreviation of less than four characters that works in the current release may not be unique in a future release. Only 4-character abbreviations, as shown in this documentation, will be supported for valid commands in future releases.

In addition to the commands fully described in this chapter, DECtalk Software has a **Design Voice** command that allows you to modify the characteristics of a voice. For

complete information on how to use the **Design Voice** command to change a voice, see Chapter 5.

Table 2-1 DECtalk Software In-Line Commands

Command	Syntax	Function
Comma Pause	[comma DD] or [:cp DD]	Inserts a comma pause into spoken text
Design Voice	[:dv XX YY]	Customizes a DECtalk Software voice by selecting and setting speaker-definition options
Dial Tones	[:dial YY]	Dials telephone numbers
Error	[:error XX]	Sets the error mode for a module
Index Mark	[:index mark DD]	Inserts marks, which are recognized by the application, into text
Log	[:log XX YY]	Sets logging modes for the module
Mode	[:mode XX YY]	Allows words and symbols to be interpreted for special use
Name	[:name XX] or [:nXX]	Selects the name of the DECtalk Software voice
Period Pause	[:period DD] or [:pp DD]	Inserts a pause equivalent to a period in a sentence into spoken text
Phoneme Interpretation	[:phoneme XX1 XX2 YY]	Allows everything within brackets to be interpreted as phonemic text
Pitch	[:pitch DD]	Raises by the value specified the frequency of uppercase letters spoken in typing mode
Play Wave Files	[:play <file>]	Plays wave files embedded in text strings
Pronounce	[:pronounce XX]	Speaks alternate, primary, or proper noun pronunciation of a word
Punctuation	[:punct XX]	Turns punctuation on and off
Rate Selection	[:rate DD]	Selects speed at which text is spoken
Say	[:say XX]	Allows DECtalk Software to speak words before they are queued
Skip	[:skip XX]	Allows users to skip specified parts of the test preprocessing
Sync	[:sync]	Synchronizes activity between DECtalk Software and an application program
Tone	[:tone DD, DD]	Creates tones of a specified length and frequency

Command	Syntax	Function
Volume	[:volume XX DD] or [:volume XX DD1 DD2]	Sets the volume

Note

Commands are not synchronous unless otherwise stated. To make a command synchronous, use the **[:sync]** command. See the **Sync** command for more information.

DECtalk Software inline commands are not supported in SAPI text buffers; they are ignored.

Comma Pause [:comma]

The **Comma Pause** command increases or decreases the length of the comma pause from the current value by the delta value specified, in milliseconds. This command is asynchronous. The comma pause can be increased and decreased. The **[:cp 0]** command resets the comma pause to its default state (approximately 160 ms). Comma pauses can be increased by 30,000 ms (30000) and decreased by 40 ms (-40). All values outside the legal range default to the nearest legal values.

SYNTAX:	[:comma DD]
ABBREVIATION:	[:comm DD]
ALTERNATE COMMAND:	[:cp DD] and [:cp 0]
OPTIONS:	none
PARAMETERS:	Pause time in milliseconds
DEFAULT:	160 ms
EXAMPLES:	[:comma 250]

Design Voice [:dv]

The **Design Voice** command customizes a DECtalk Software voice by selecting and setting speaker-definition options. This command is asynchronous. DECtalk Software voices provide an adequate selection for most applications. However, if you have a special application requiring a monotone or unusual voice, you can use the **Design Voice** command to modify any DECtalk Software voice. The speaker-definition options and parameters can be entered as a string or one at a time.

The **Design Voice** command options and parameters are documented and explained in Chapter 5.

Dial Tones [:dial]

The **Dial Tones** command generates tones called Dual Tone Multiple Frequency (DTMF) Tones or Touch-Tones™. The **Dial Tones** command is a synchronous command that can be used to dial a telephone. The tone characters are 0-9, #, *, and A, B, C, D (in uppercase only). A non-tone character generates a silent interval between dialed digits. White space characters (tabs, spaces) should not be used as dial tone characters.

SYNTAX:	[:dial YY]
ABBREVIATION:	none
ALTERNATE COMMAND:	none
OPTIONS:	none
PARAMETERS:	String of dial characters (0-9, A, B, C, D, #, *)
DEFAULT:	none
EXAMPLES:	[:dial 508-555-1212]

Error [:error]

The **Error** command asynchronously sets the error mode for the text-to-speech system. This command is useful for debugging an application. When opening a log file, using the **[:error text]** command, DECtalk Software checks to see if the system is in startup mode. If it is in one of the text-to-speech special modes (wave-file, log-file, or text-to-speech memory) instead, this command fails. See the Special Text-To-Speech Modes section in the *DECTalk Software Programmer's Guide* for more information.

In the default setting for DECtalk Software, the **Error** command has the **speak** option turned on. This means that DECtalk Software reports any command errors that it can detect. You can set the **[:error ignore]** command to avoid this problem.

SYNTAX:	[:error XX]	
ABBREVIATION:	[:erro XX]	
ALTERNATE COMMAND:	none	
OPTIONS:	text	Log all text to a file in the current directory called <i>log.txt</i>
	ignore	Ignore all errors
	speak	Speak error string in the current command
PARAMETERS:	none	
DEFAULT:	Error string is spoken	
EXAMPLES:	[:error speak]	

Index Mark [:index mark]

Index Mark commands report the progress of the text as it is spoken. Index marks are position markers; they do not modify heuristics or word pronunciations in any way. The index mark sequence inserts a flag into the text stream. When DECtalk Software encounters an **Index Mark** command, a message is sent to the calling application. Index marks cannot be put in the middle of a word. This command is synchronous.

- For more information on using index marks, refer to the Index Marks for Speech Status section.
- For more information about returning index marks to a calling application, see the Callback Routines and Window Procedures section in the *DECtalk Software Programmer's Guide*.

If a callback routine or window procedure is not specified in the startup function, index marks in the text are ignored.

SYNTAX:	[:index mark DD]
ABBREVIATION:	[:inde DD]
ALTERNATE COMMAND:	none
OPTIONS:	none
PARAMETERS:	Numeric index mark value
DEFAULT:	none
EXAMPLES:	[:index mark 01]

Note

This command is not recommended for use with the Microsoft Speech API (SAPI). This command was designed for the DECtalk Speech API (DAPI) only. Refer to the *DECtalk Software Programmer's Guide* for a detailed description of the DECtalk Speech API, also referred to as the DECtalk Software API.

Log [:log]

The **Log** synchronously logs text, phonemes, or syllables into a log file. The log file, called *log.txt*, can be found in the current directory. When opening a log file, DECtalk Software checks to see if the system is in startup mode. If it is in one of the text-to-speech special modes (wave-file, log-file, or text-to-speech memory) instead, this command fails. See the Special Text-To-Speech Modes section in the *DECTalk Software Programmer's Guide* for more information.

SYNTAX:	[:log XX YY]	
ABBREVIATION:	none	
ALTERNATE COMMAND:	none	
OPTIONS:	text	Log all text to the log file
	syllables	Log converted syllables to the log file
	Phonemes	Log converted phonemic text to the log file
PARAMETERS:	On	Turns on the specified log option
	Off	Turns off the specified log option
	Set	Turns on the specified log option while turning off all other log options
DEFAULT:	All of the log options are turned off	
EXAMPLES:	[:log phonemes on] The phonemes for this sentence will be stored in a file named log.txt [:log phonemes off]	

Mode [:mode]

The **Mode** command changes the mode for all text processed after this command. It remains in effect until the end of the file is reached or until the next **Mode** command is encountered. This is an asynchronous command. Refer to the description of the **Sync** command for information on how to make this command synchronous.

SYNTAX:	[:mode XX YY]	
ABBREVIATION:	None	
ALTERNATE COMMAND:	None	
OPTIONS:	math	Change interpretation of selected symbols
	europe	Select European cardinal pronunciation
	spell	Spell all words
	name	Pronounce all uppercase verbs as proper nouns (see also [:pronounce name] command)
	latin	Not supported
	email	Activates email parsing rules
PARAMETERS:	on	Turns on the specified mode option
	off	Turns off the specified mode option
	set	Turns on the specified mode option while turning off all other mode options
DEFAULT:	All of the mode options are turned off	
EXAMPLES:	[:mode spell on]	

Europe Mode Example:

When Mode is set to Europe, **[mode europe on]**, a comma (,) is the separator between the integer and fraction part of a number. A period (.) is the separator between 3-digit blocks.

1.255 (United States) = 1,255 (Europe)

125,873 (United States) = 125.873 (Europe)

Math Mode Example:

When Mode is set to Math, **[[:mode math on]]**, special symbols and characters are pronounced with mathematical meanings. Specifically, the characters in Table 2-1 are treated differently:

Table 2-1 DECtalk Interpretation of Special Characters

Symbol	Name	DECtalk Says...
+	plus	plus (no change from normal speech)
-	hyphen	minus
*	asterisk	multiplied by
/	slash	divided by
^	circumflex	to the power of
<	less than	less than
>	greater than	greater than
=	equal sign	equals
%	percent sign	percent
.	period	decimal point
xxE-xx	(spelled)	(scientific notation)

Name Mode Example:

When Mode is set to Name, **[[:mode name on]]**, uppercase words that occur in locations other than the beginning of a sentence are interpreted as special cases and pronounced as proper names.

Note

Do not enable the **[[:mode name]]** command except when pronouncing lists of names. This command interprets any uppercase word as a name. When finished, make sure that this mode is set to *off*. For the occasional use of this utility, use the **[[:pronounce name]]** command.

Email Mode Example:

When **Mode** is set to email, **[:mode email on]**, email parsing rules are activated to find email headers, to determine which email headers to speak, and to find email text.

- The specific text strings at the start of a line that initiate email header mode are, as follows:

```
From:
Return-Path:
%====Internet
Message-ID:
```

In email header mode, the DECtalk text preprocessor goes into line-by-line processing.

- In email header mode, the text lines saved for text preprocessing are the text lines that start with the following:

```
Sent:
Date:
Subject: Re:
Subject:
From:
To:
cc: or CC:
----- Forwarded Message
```

- In email header mode, each text line saved for text preprocessing gets a pause added at the end of the line.
- When DECtalk detects an empty line while still in email header mode, DECtalk goes into email text processing mode. An empty line is a line that has a <Return/Enter> only.
- In email text processing mode, DECtalk Software does the regular text preprocessing and checks for another possible email header that starts with %====Internet. If the text string %====Internet is found, DECtalk goes into email header mode.

The **[:mode email off]** command ends email processing mode.

An example of a UNIX email message with the Mode command for email is as follows:

```
[ :mode email on ]
```

From John Doe Wed Jun 10 18:07:28 EST
Return-Path:<john@node.com>
Received: from home.node.com ([127.0.0.1]) by smtp.node.com
Message-ID: <32FB6581.581A@smtp.node.com>
Date: Wed, 10 Jun 1998 18:07:28 EST
From: john@node.com (John Doe)
Organization: SMART Modular Technologies, Inc.
X-Mailer: ELM
MIME-Version: 1.0
To: jane@node.com
Subject: DECTalk Parsing
Content-Type: text/plain; charset=us-ascii
Content-Transfer-Encoding: 7bit
X-Mozilla-Status: 0001

Hi Jane,

At 11:52 EST on Wed Jun 10, I found a great web site. It's SMART Modular Technologies, Inc. web site. all about the DECTalk products. Take a look at URL:

<http://www.smartm.com/>

Let me know what you think by mailing me at john@aol.node.c or snail mail at: John Doe, 4321 St. James St.,Mt. View, CA 12345-6789, phone (123)297-4863. Or write to Dr. John Doe, 42nd St., Boston, MA 01234, phone 617-546-2345.

See ya! :-)

John

%=====Internet headers and postmarks (see
DECWRL::GATEWAY.DOC)=====

%Received: from smtp.node.com by node.com (5.6/rmc-22feb94)
idAA17792;Wed, 8 Sep 22:47:37 -0400

%Received: from node.com by node.com (8.7.5/UNIX 1.2/1.0/WV)
idWWA13939; Wed, 8 Sep 1998 22.35.28 -0400 (EDT)

%Received: from node.com(smtp.node.com[127.0.0.1]) by
worldaccess.com (8.6.10/8.6.10) with SMTPidTAA10463 for
<jane@node.com>;Wed, 8 Sep 1999 19:33:57 -0700

%Message-Id:<32094F06.4045@node.com>

%Date: Wed, 8 Sep 1999 19:20:54 -0700

%From: john Doe <John@node.com>

%Organization: SMART Modular Technologies, Inc.

%X-Mailer: ELM

%Mime-Version: 1.0

```
%To: "Jane Smith, jane@node.com"  
%Subject: Re: DECTalk Parsing  
%References: <9608071721.AA16334@mpde/com>  
%Content-Type: text/plain; charset=us-ascii  
%Content-Transfer-Encoding: 7bit
```

```
[ :mode email off]
```

The email header lines shown in bold are the lines saved for text preprocessing.

Some of the lines beginning with the % character in the example are shown wrapping to a second or third line. However, the actual text line is the line of text ended by a line terminator, such as <Return/Enter>.

Name [:name]

The **Name** command allows the current speaking voice to be changed to one of ten built-in DECTalk Software voices. XX represents the speaker name or letter variable for each voice. The letter variable is the first letter of the speaker name. This command is synchronous.

SYNTAX: [:name XX]

ABBREVIATION: none

ALTERNATE COMMAND: [:nXX]

OPTIONS:	Speaker	Variable	Description
	PAUL	p	Default male voice
	HARRY	h	Full male voice
	FRANK	f	Aged male voice
	DENNIS	d	Male voice
	BETTY	b	Full female voice
	URSULA	u	Aged female voice
	WENDY	w	Whispering female voice
	RITA	r	Female voice
	KIT	k	Child's voice
	VAL	v	Val's voice

PARAMETERS: none

DEFAULT: PAUL

EXAMPLES: [:name KIT] or [:nk]

Note

A user can change any of the voice characteristics of the current speaker by using the **Design Voice [:dv]** command. These changes are active only while the current speaker remains current. To save the voice changes, use the save option of the **Design Voice** command, which saves the changes as the voice of Val. For information on the individual characteristics of a speaker or details on how to change a voice using the **Design Voice** command, see Chapter 5.

Period Pause [:period]

The **Period Pause** command increases or decreases the length of the period pause from the current value by the delta value specified in milliseconds. The **[:pp 0]** command resets the period pause to its default state (approximately 640 ms). Period pauses can be increased by 30,000 ms (30000) and decreased by 380 ms (-380). All values outside the legal range default to the nearest legal values. This command is asynchronous.

SYNTAX:	[:period DD]
ABBREVIATION:	[:peri DD]
ALTERNATE COMMAND:	[:pp DD] and [:pp 0]
OPTIONS:	none
PARAMETERS:	Pause time in milliseconds
DEFAULT:	640 ms
EXAMPLES:	[:period 250]

Phoneme Interpretation [:phoneme]

When phoneme interpretation is set, the **Phoneme Interpretation** command allows everything within brackets to be interpreted as phonemic text. All phoneme interpretation of text can be silenced by using the **[:phoneme silent on]** command. By default, the text is spoken without phoneme interpretation. This command is asynchronous.

When you phonemicize text, put valid phoneme strings in brackets. A list of valid phonemic symbols can be found in Table 4-1.

Phoneme interpretation allows you to specify the preferred pronunciation of a word or phrase. It is important to note that this command sets the left bracket ([) and right bracket (]) characters as phoneme delimiters. When the user has the phoneme interpretation turned on [:phoneme on], all text and characters that appear between brackets are interpreted as phonemic text and is pronounced as such. For example, to say the word *associate*, simply embed the phonemic string [axs 'owshiyeyt] in the text string. Note that the pronunciation of the phonemic string is different depending on whether phoneme interpretation is on or off.

When phoneme interpretation is on, additional attributes can be associated with the phoneme text. For information on how to code a phoneme sequence to produce musical sounds, refer to Chapter 4. For a complete list of stress and syntactic symbols that can be used with phoneme text, see Table 4-1 and *Table 4-2*.

Note

Arpabet mode is a 2-character system. All single character symbols must be followed by a space so that faulty translations do not occur. Consider the phonemic representation of “whitehorse,” [* w 'ayt hxowr s]. The letter “t” in this phonemic representation must be followed by a space, so that it is not interpreted as part of the phonemic symbol [th] in the representation of “whitehorse.”

Some older versions of DECtalk Software supported single characters in arpabet mode. Application programs written for use with those versions may fail to function correctly when used with DECtalk Software V4.6 and subsequent versions.

SYNTAX:	[:phoneme XX1 XX2 YY] or [:phoneme arpabet speak on]
ABBREVIATION:	[:phon XX1 XX2 YY]

ALTERNATE COMMAND:	None	
OPTIONS:	arpabet	Set phonetic interpretation to arpabet alphabet. (Currently, this option is the only alphabet allowed.)
	speak	If phoneme interpretation is on, speak encountered phonemes. The speak option is ignored if phoneme interpretation is off.
	silent	If phoneme interpretation is on, do not speak encountered phonemes. The silent option is ignored if phoneme interpretation is off.
PARAMETERS:	On	Set phoneme interpretation on
	Off	Set phoneme interpretation off
DEFAULT:	Phonetic interpretation is off	
EXAMPLES:	[:phoneme arpabet speak on] [axs 'owshiyeyt] <i>associate</i> [:phoneme speak on] [axs 'owshiyeyt] <i>associate</i> [:phoneme on] [axs 'owshiyeyt] <i>associate</i> [:phoneme speak off] [axs 'owshiyeyt] <i>pronounced as axassociate</i> [:phoneme off] [axs 'owshiyeyt] <i>pronounced as axassociate</i> [:phoneme silent off] [axs 'owshiyeyt] <i>pronounced as axassociate</i> [:phoneme silent on] [axs 'owshiyeyt] <i>associate not spoken</i>	

Note

Make sure that you use a right bracket (]) to end the phonemic symbols. If you do not, any normal text appearing after the phonemic symbols sounds garbled. One right bracket is sufficient to close phonemic mode. It is sometimes useful to begin a text file with a right bracket (]) to ensure that text is not interpreted phonemically. A command sequence consisting of a left bracket followed by a colon ([:) is always interpreted as the beginning of a command.

Pitch [:pitch]

The **Pitch** command raises, by the value specified, the frequency of uppercase letters spoken in typing mode using the typing table (spoken one letter at a time). The default frequency difference between spoken lowercase and uppercase letters is 35 Hz. The frequency difference enables users to distinguish between uppercase and lowercase letters. You can return the pitch increment for uppercase letters to the default value by specifying the command **[:pitch 35]** or by restarting **Speak**. This command is asynchronous.

DECTalk adds the value of the argument, DD (in Hertz), as a pitch increment, to the uppercase letters in the next phoneme string it processes. However, the **Pitch** command is asynchronous. Place a **Sync** command in the character stream after the **Pitch** command to ensure that the **Pitch** command is processed before the letters that follow it in the buffer you are using.

SYNTAX:	[:pitch DD]
ABBREVIATION:	none
ALTERNATE COMMAND:	none
OPTIONS:	none
PARAMETERS:	frequency in hertz
DEFAULT VALUE:	35
 EXAMPLES:	 [:pitch 60] bBcCdD [:pitch 35] eEfFgGhH

Play Wave Files [:play]

Play Wave Files is a synchronous command that plays any wave file that is supported by your computer's audio system. When opening a wave file, DECtalk Software checks to see if the system is in startup mode. If it is in one of the text-to-speech special modes (wave-file, log-file, or text-to-speech memory) instead, this command fails. See the Special Text-To-Speech Modes section in the *DECtalk™ Software Programmer's Guide* for more information.

SYNTAX:	[;play <file>]
ABBREVIATION:	none
ALTERNATE COMMAND:	none
OPTIONS:	none
PARAMETERS:	A directory path and file name
DEFAULT:	none
EXAMPLES:	[;play bell.wav]

Pronounce [:pronounce]

The **Pronounce** command determines the type of pronunciation for the word immediately following this command. This command is synchronous.

Use the **[:pronounce alternate]** command to obtain an alternative pronunciation for a word. See the Homograph tables in Chapter 4 for examples of primary and alternate pronunciations of words. Using the word *wind* as an example, the primary pronunciation is w 'ihn d, as in 'the wind is blowing'. The alternate pronunciation, denoted by **[:pronounce alternate]** wind, is w 'ayn d, as in 'wind up the top'.

Use the **[:pronounce name]** command to pronounce a word as a proper name. First names, last names, street names, and place names are all examples of proper names.

SYNTAX:	[:pronounce XX]	
ABBREVIATION:	[:pron XX]	
ALTERNATE COMMAND:	none	
OPTIONS:	alternate	Uses the alternate pronunciation
	primary	Uses the primary pronunciation
	name	Uses the proper name pronunciation
PARAMETERS:	none	
DEFAULT:	Uses the primary pronunciation	
EXAMPLES:	Terry [:pronounce name] Doucette played [:pronounce primary] bass in the band.	

Punctuation [:punct]

The **Punctuation** command lets you specify how DECTalk software treats punctuation marks when it encounters them in text. This command is synchronous. The four options of the **Punctuation** command are:

- **none** – No punctuation is spoken.
- **some** – Text is read normally, and punctuation marks are used to mark pauses, changes in pitch, and so on.
- **all** – All punctuation is spoken, for example “,” is spoken as “comma.”
- **pass** – Turns off all special punctuation processing. For example, periods as part of file names are not spoken.

The pass option is useful in proofreading, as well as in applications where special characters are encountered, such as in a computer program. See Chapter 6 for more information on preprocessor parsing for treatment of punctuation.

Note

When the **[:punct none]** command is used, no punctuation is pronounced, although dollar amounts and percentages still are processed.

SYNTAX:	[:punct XX]	
ABBREVIATION:	[:punc XX]	
ALTERNATE COMMAND:	none	
OPTIONS:	none	Punctuation symbols and some other symbols are not spoken as words; all punctuation is treated as text breaks
	some	Text is read normally; clause boundary punctuation is not spoken, but all symbols such as \$ are spoken as words
	all	All punctuation symbols and other symbols are spoken as words
	pass	All special punctuation processing is turned off
PARAMETERS:	none	
DEFAULT:	[:punct some]	
EXAMPLES:	[:punct none]	

Rate Selection [:rate]

The **Rate Selection** command sets the speaking rate in DECtalk Software. The rate can range from 75 to 600 words per minute. All values outside the range of 75 to 600 default to the nearest legal value. For example, if you select a speaking rate of [:rate 880] or 880 words per minute, DECtalk Software defaults to 600 words per minute. The DECtalk synthesizer starts at a rate of 200 words per minute by default. This command is asynchronous.

SYNTAX:	[:rate DDD]
ABBREVIATION:	none
ALTERNATE COMMAND:	none
OPTIONS:	none
PARAMETERS:	Rate in words per minute
EXAMPLES:	[:rate 400]

Say [:say]

The **Say** command specifies when speaking begins. The **Say** command options are speak on end of clause (clause), speak on end of word (word), speak on end of letter (letter), and speak on end of line (line). This command is synchronous.

In DECtalk Software, each clause, word, or letter is spoken as it is queued. In word and letter mode, DECtalk Software does not need to wait for a clause terminator to begin speaking. Word mode is similar to letter mode except text is spoken a word at a time. A space after a character or string of characters causes that string to be spoken. This mode interacts with the rate selection command so you can increase or decrease the rate at which the text is spoken. In clause mode, speaking starts when DECtalk Software is sent a clause terminator (period, comma, exclamation point, or question mark) followed by a space. There is no time-out limit. This is the normal mode where text is spoken a phrase, clause, or sentence at a time. Clause mode is the default mode.

SYNTAX:	[:say XX]	
ABBREVIATION:	none	
ALTERNATE COMMAND:	none	
OPTIONS:	clause	Speak on end of clause.
	word	Speak on end of word.
	letter	Speak on end of letter.
	filtered letter	Speak on end of letter, ignoring control characters, such as "vertical tab" and "line feed"
	line	Speak on end of line.
PARAMETERS:	none	
DEFAULT:	[:say clause]	
EXAMPLES:	[:say word]	

Note

In letter mode, the left bracket is spoken only after the next character is entered because DECtalk Software needs to know if this is the beginning of a new command.

Skip [:skip]

The **Skip** command allows the user to skip various parts of the text preprocessing. It remains in effect until another **Skip** command is issued. The command allows only one value to be in effect at a time. This command is synchronous.

See Chapter 6 for information on the preprocessor rules for parsing.

SYNTAX:	[:skip XX]	
ABBREVIATION:	none	
ALTERNATE COMMAND:	none	
OPTIONS:	all	Skip all preprocessing
	cpg	Skip codepage translation
	none	Do not skip anything
PARAMETERS:	none	
DEFAULT:	The default is set to [:skip none] .	
EXAMPLES:	[:skip cpg]	
	[:skip all]	
	[:skip none]	
NOTES:	This command allows only one option to be in effect at a time; in the example, [:skip cpg] overrides [:skip all] .	

Sync [:sync]

The **Sync** command provides coordination between an application program and DECtalk Software. This command is synchronous.

An application program can send data to DECtalk Software faster than DECtalk Software can speak it. Therefore, if the user needs to carry on a dialogue with the application program, the application program must be notified that DECtalk Software has finished speaking the text sent to it.

When the program sends the **Sync** command, DECtalk Software finishes speaking any pending text before processing the next text command. This command also acts as a clause boundary, just the same as a comma, period, exclamation point, question mark, semicolon, or colon when followed by a space.

Some DECtalk inline commands are asynchronous. To ensure that these commands are processed before the text following them, place a **Sync** command after an asynchronous command that you want to synchronize. In the case of the **Pause** command, you need to place a **Sync** command before the **Pause** command to guarantee that all text preceding the **Pause** command is processed before the pause occurs.

SYNTAX:	[:sync]
ABBREVIATION:	none
ALTERNATE COMMAND:	none
OPTIONS:	none
PARAMETERS:	none
DEFAULT:	N/A
 EXAMPLES:	 My name is Bill S [:sync]

Tone [:tone]

The **Tone** command is a synchronous command that generates sounds of different frequencies and lengths based on the parameters you set. This command allows you to make a wide variety of sounds for purposes such as notification or warnings. Regular tones can also be used for a number of other purposes, such as indications of a margin bell. This command is synchronous.

SYNTAX:	[:tone DD, DD]	
ABBREVIATION:	none	
ALTERNATE COMMAND:	none	
OPTIONS:	none	
PARAMETERS:	Frequency:	Sets the frequency to the desired level
	Duration:	Tone duration in milliseconds
DEFAULT:	none	
EXAMPLES:	[:tone 500,500]	

Volume [:volume]

The **Volume** command is a synchronous command that changes the volume settings. DECTalk Software changes the audio system gain in increments from 0 to 99, in decibels (dB). Increments or decrements of 10 to 20 provide a perceptible increase or decrease in volume. The **Volume Set** option is an absolute command; **Volume Up** and **Volume Down** options are relative commands and increase or decrease the original value. This command does not affect the volume when the application writes wave files or uses the speech-to-memory capability, because scaling is not done to the speech samples based on the **Volume** command settings.

Monaural Volume Control

The following monaural volume commands are supported. DD must be in the range of 0 to 99.

SYNTAX:	[:volume XX DD]	
ABBREVIATION:	[:volu XX DD]	
ALTERNATE COMMAND:	none	
OPTIONS:	set	Sets the volume to the desired level
	up	Increases the volume by the desired amount
	down	Decreases the volume by the desired amount
PARAMETERS:	Volume amount	
DEFAULT:	none	
EXAMPLES:	[:volume up 30]	

Stereo Volume Control

The following stereo volume commands are supported. The value of DD1 or DD2 must be in the range of 0 to 99.

SYNTAX:	[:volume XX DD1 DD2]	
ABBREVIATION:	[:volu XX DD1 DD2]	
ALTERNATE COMMAND:	none	
OPTIONS:	lset	Sets the left channel to the desired level
	lup	Increases the left channel by the desired amount
	ldown	Decreases the left channel by the desired amount

rset	Sets the right channel to the desired level
rup	Increases the right channel by the desired amount
rdown	Decreases the right channel by the desired amount
sset	Sets the left channel to the DD1 amount and the right channel to the DD2 amount

PARAMETERS:

Volume amounts

DEFAULT:

none

EXAMPLES:

[:volume set 80 60]

Chapter 3 — *Using In-Line Commands*

This chapter provides an in-depth look at the DECTalk Software in-line commands, commands that can be used within a DECTalk Software text file or application. For information on using the **Design Voice** command see Chapter 5.

Topics include:

- Changing Rhythm, Stress, and Intonation
- Developing an Electronic Mail-Reading Application
- Optimizing the Quality of Spoken Text
- Index Marks for Speech Status
- Speaking Rate
- Adjusting Period and Comma Pause Duration
- Text-Tuning Example
- Avoiding Common Errors

Changing Rhythm, Stress, and Intonation

DECTalk Software uses stress and syntactic symbols to control aspects of rhythm, stress, and intonation patterns within a spoken text file. These symbols include punctuation marks such as commas, periods, and parentheses. Punctuation marks are recognized by DECTalk Software as indicating special phrasing requirements.

Table 4-2 list these symbols.

Developing an Electronic Mail-Reading Application

DECTalk Software supplies an email parser for Windows and for Tru64 UNIX (not for MS-DOS). See the email option of the **Mode** command for additional information.

If you wish to write your own electronic mail preprocessor, implement the following text conversions before sending the text to DECTalk Software:

- Parse the header boilerplate to remove extraneous information.
- Add the new paragraph symbol [+] to each blank line between paragraphs if DECTalk Software is speaking paragraphs of text. Refer to *Table 4-2* for the complete list of syntactic symbols.
- Create your own application-specific user dictionary for words that have an application-specific pronunciation.
- If DECTalk Software is connected to a database containing names, consider one of the following options:

Add the **Pronounce** command before the database word to force the language specific rules on the name. For example:

[:pronounce name] name

See the **Pronounce** command description in Chapter 2 for more information.

- ❑ Replace the database word with its phoneme text. For this option, you must have phoneme interpretation turned on. See the **Phoneme Interpretation** command description in Chapter 2 for more information.

- Scan the text for strings of numbers in a format understandable to your application but not to DECTalk Software. For example, if you can extract the time format from an electronic mail message, you can add code to your application to expand it to its “o’clock” form.

In many applications, the listener might want to write down number strings (such as prices or telephone numbers). Your application can scan the text for strings of numbers and, when they are found, send them to DECTalk Software in a way that includes pauses at critical locations. For example:

```
The number is, 1 (800) 5 5 5, 1 2 3 4. [:rate 120]
That is, [_<300>] 1 (800), [_<500>] 5 5 5,
[_<900>] 1 2 3 4. [:rate 180].
```

Refer to Table 4-1 for a complete list of phoneme symbols, including the silent underscore (_) symbol. See Chapter 4 for the syntax to add duration and pitch to phoneme text.

The spaces between the numbers ensure that “five five five” is spoken rather than “five hundred fifty five.” You can also use the **[:mode spell on]** command to produce the same results. The slower speaking rate, **[:rate 120]**, and the silence phonemes, **[_<300>]**, **[_<500>]**, **[_<900>]**, of specified duration, were carefully selected to allow enough time for the listener to write down the entire number. Silence phonemes were positioned after the commas (that is, **[_<300>] 1 (800)**, **[_<500>]**), to maintain appropriate intonation.

As another example, if your application is required to speak sums of money (such as bank balances or item costs), you might code the text to say:

```
Your balance is $244.05. That is, 2 4 4, [_<400>] point 0
5, [_<400>] dollars.
```

- When spelling an item, your application might need to distinguish the case of letters. Consider using the **Pitch** command (see Chapter 2) or different voices to distinguish between uppercase and lowercase letters. For example:

```
[:nf]Maynard [:nf]M[:nb]a y n a r d [:nf]Maynard.
```

Optimizing the Quality of Spoken Text

DECtalk Software can generally choose correct pronunciations by itself. For example, if you enter the following sentences:

He produced a lot of *REFUSE*. He *REFUSED* the produce.

He *INSERTS* 5 *INSERTS* per minute. He *DELIBERATED* *DELIBERATELY* for a long time.

Generally, DECTalk Software correctly selects the proper homograph. However, in certain unique contexts, the following user intervention may be needed:

- Replace the correct spelling of the word with a clever misspelling.

I red yesterday that . . .

- Spell the word phonetically.

I [r ' ehɪ] yesterday that . . .

Note

For words that have two pronunciations (homographs), see the Homograph tables found in Chapter 4.

Additionally, use the following steps to optimize spoken text.

1. If a word is a compound, use a hyphenated spelling to help DECTalk Software see the two parts of the compound.

The slide-show host . . .

2. Replace the text version by a phonemic string. Use the commands and phonemic symbols, but make sure to place the lexical stress pattern correctly.

Note

Sometimes, a word does not sound quite right even when the best phonemic representation is selected. Usually, such subtle pronunciation defects are not correctable.

3. Now that each word has been pronounced in the best possible way, listen to the total sentence rhythm and accent pattern. If it is not right, follow these steps.
 - (a) If it sounds as if there should be a short pause in a specific sentence location, but DECtalk Software says the sentence without a pause, insert a comma between the words in question.
 - (b) If the wrong word is emphasized in the sentence, emphasize the word that is supposed to take the emphasis with the correct stress symbols.

The ["] younger man is the trouble-maker, not the older one.

- (c) Use the stress symbols slash [/], backslash [\], and slash and backslash [/ \] to make final adjustments. Refer to Table 4-1 for a complete list of stress symbols.
- (d) If none of these actions gives you a satisfactory sentence, you can still specify duration and fundamental frequency motions for all phonemes with the **Design Voice** command, as explained in Chapter 5.

Index Marks for Speech Status

By embedding an **Index Mark** command in text, you can provide non-blocking synchronization. DECtalk Software can use index marks to track exactly when the text was spoken. The index marks bind themselves to the next speech sound, so you **MUST** always include a sound after the **Index Mark** command. Therefore, if you send, “Hello. [:index mark 5]”, DECtalk Software will wait until the next sound to send the mark to the application. Index marks cannot be put in the middle of a word.

Index Marks are handled differently depending on whether the text-to-speech system is in speech-to-memory mode. When an **Index Mark** command is encountered while not in this mode, the index mark is returned to the calling application with a message type of `TTS_MSG_INDEX_MARK`.

If the text-to-speech system is in the speech-to-memory mode, then the message type is `TTS_MSG_BUFFER` and the index marks are returned in the index mark array, if allocated, of the memory buffer structure. In addition to the index mark value, there is an index sample number also passed in the array, to allow you to determine which sample in the memory buffer corresponds to that index mark.

See the Chapter 3 in the *DECtalk Software Programmer’s Guide* for more information on returning index marks to the calling application.

Speaking Rate

The default speaking rate is 200 words per minute (WPM). DECTalk Software speaking rates range from 75 to 600 WPM. In the **Rate** command, valid speaking rates are between 75 and 600. Rates specified outside this range are limited to the nearest legal value. Speaking rates can be adjusted to very slow, very fast, or anywhere in between by using the following commands:

- `[:rate 120]`

Although the slowest possible rate is 75 WPM, 120 WPM is ideal for information such as phone numbers, which need to be copied down by a listener. Unless the listener is actually copying down each numeral, it might be frustrating to listen to extended speech at slow rates.

- `[:rate 160]`

This rate is moderate (160 WPM). It sounds a little slow, but is sometimes preferred when DECTalk Software is speaking math equations or long lists of acronyms.

- `[:rate 200]`

This is the default rate for DECTalk Software (200 WPM). This rate is ideal for listening to continuous text under optimal conditions.

- `[:rate 240]`

Experienced listeners may prefer to skim material at this rate (240 WPM). Inexperienced listeners may not understand every word at this rate.

- `[:rate 350]`

This rate (350 WPM) is too fast to follow, but can be used to quickly scan sections of text.

- `[:rate 550]`

This rate (550 WPM) is the fastest usable rate. It is too fast for most people to follow, but can be used to scan text very quickly.

Changes in the speaking rate influence the duration and the number of pauses in text, as well as the duration of individual phonemes. At rates below 140 WPM, DECTalk Software inserts pauses at all phrase boundaries and pauses, and inserts phonemes near

the ends of phrases. At rates faster than 240 WPM, DECtalk Software deletes all pauses and shortens phonemes.

Note

Near the beginning of a phrase, phonemes are fairly short at both slow and fast speaking rates.

Adjusting Period and Comma Pause Durations

At the default speaking rate of 200 WPM, DECTalk Software pauses about half a second after a period in the text and about a sixth of a second after a comma. When you change the speaking rate, the pause durations are automatically adjusted.

In some situations, you might prefer to change the pause after a period or a comma without changing the speaking rate. For example, to get DECTalk Software to read a list of words with a longer pause after each (to allow the listener to write them down), use the **Period Pause** command or the **Comma Pause** command.

- `[:period 4500] apple. banana. strawberry.`

This command adds a period pause of 4,500 ms (4.5 seconds) to the standard half-second pause that occurs after a period in text. The total pause between words is about five seconds. The accepted range for the period pause parameter is -380 to 30,000 ms. A negative value for this parameter shortens the standard period pause.

- `[:comma 4800] apple, banana, strawberry,`

This command adds a comma pause of 4,800 ms (4.8 seconds) to the standard sixth of a second pause that occurs after a comma in the text at normal speaking rate. The total pause between words separated by a comma is about five seconds. The accepted range for the comma pause parameter is -40 to 30,000 ms. Values specified outside this range are limited to the nearest legal value.

- `[:pp 0 :cp 0]`

This command resets the period pause and comma pause to their normal default values.

Text-Tuning Example

Although DECTalk Software allows for natural text-to-speech synthesis, the quality of speech can often be enhanced by giving it a more natural flow. Much of this tuning involves strategic placement of commas and periods, which tell the application to pause. The spoken language and written text are different, because spoken text generally does not contain information about pausing.

The text that follows is presented twice: the first time as originally written, and the second time after phonemic and textual fixes were applied. For a complete list of stress and syntactic symbols, refer to Table 4-1 and *Table 4-2*.

Original Version

```
[ :np] A California Shaggy Bear Tale for Seven DECTalk Software Voices  
by Dennis Klatt  
[ :np] Once upon a time, there were three bears.  
They lived in the great forest and tried to adjust to modern times.  
[ :nh] I'm papa bear. I love my family but I love honey best.  
[ :nb] I'm mama bear. Being a mama bear is a drag.  
[ :nk] I'm baby bear and I have trouble relating to all of the demands of  
older bears.  
[ :np] One day, the three bears left their condominium to search for honey.  
While they were gone, a beautiful young lady snuck into the bedroom through  
an open window.  
[ :nw] My name is Wendy. My purpose in entering this building should be  
clear. I am planning to steal the family jewels.  
[ :np] Hot on her trail was the famous police detective, Frank.  
[ :nf] Have you seen a lady carrying a laundry bag over her shoulder?  
[ :np] A woman kneeling with her left ear firmly placed against a large rock  
responded.  
[ :nu] No. No one passed this way. I've been listening for earthquakes all  
morning, but have only spotted three bears searching for honey.
```


Revised Version

In this section, text from the original example is enhanced with DECTalk Software embedded commands. Phoneme interpretation is turned on to allow the stress and syntactic symbols to be translated. See the Phoneme Interpretation command for more information.

Turn on phoneme interpretation

```
[ :phoneme arpabet speak on]
```

Add periods to add brief pauses after the title and author.

```
[ :np] A California Shaggy Bear Tale for Seven DECTalk Software Voices.  
By Dennis Klatt.
```

```
[ :np] Once upon a time, there were three bears.
```

```
They lived in the great forest and tried to adjust to modern times.
```

Add commas to increase pause length and quotation marks for emphatic stress.

```
[ :nh] I'm papa bear. I love my family, but I love ["]honey best.
```

```
[ :nb] I'm mama bear. Being a mama bear is a drag.
```

```
[ :nk] I'm baby bear and I have trouble relating to all of the demands of  
older bears.
```

```
[ :np] One day, the three bears left their condominium to search for honey.  
While they were gone, a beautiful young lady snuck into the bedroom through  
an open window.
```

```
[ :nw] My name is Wendy. My purpose in entering this building should be  
clear. I am planning to steal the family jewels.
```

Use a new paragraph symbol [+] to begin a new paragraph.

```
[ :np] [+] Hot on her trail was the famous police detective, Frank.
```

```
[ :nf] Have you seen a lady carrying a laundry bag over her shoulder?
```

Add commas to increase pause length and phrasing.

```
[ :np] A woman, kneeling with her left ear firmly placed against a large  
rock, responded.
```

Use pitch rise and fall symbols [/ \] and emphatic stress symbols ['] to add pitch control and emphatic stress.

```
[ :nu] [']No. No [ /]one passed this [ / \]way. I've been listening for  
[']earthquakes all morning, but have only spotted three bears searching for  
honey.
```

Avoiding Common Errors

When using DECTalk Software, try to avoid making common errors by doing the following:

- When you make voice-selection changes, always return to the default voice you have chosen. If you forget to return DECTalk Software to the default voice after using one of the other voices, all future text uses the currently selected voice.
- The default setting for the **Error** command is to have the speak option turned on. This means DECTalk Software reports any command errors that it can detect. Set the **[:error ignore]** command to avoid this action.
- Enter a right bracket (**]**) at the beginning of your text if you use the **Phoneme Interpretation** command.
- If the **[:phoneme arpabet speak on]** command is entered to allow phonemic input, it is possible for DECTalk Software to enter phonemic mode unintentionally.
 - ❑ If the text being spoken contains an unexpected left bracket (**[**), all text after the left bracket (**[**) is interpreted as phoneme text. In the following example, 'apple, banana, strawberry' is interpreted as phoneme text.

```
[ :phoneme arpabet speak on ] Here is the list [apple,  
banana, strawberry].
```
 - ❑ If you forget to enter a right (**]**) bracket after a phonemic entry, all text after the missing right bracket (**]**) is interpreted as phoneme text. In the following example, 'Ladies and Gentlemen' is interpreted as phoneme text.

```
[ :phoneme arpabet speak on Ladies and Gentlemen
```
- DECTalk Software inline commands are not supported in SAPI text buffers; they are ignored.

Chapter 4 — DECtalk Software Reference Tables

DECtalk Software reference tables include:

- Phonemic Symbols
- Stress and Syntactic Symbols
- Phonemes Listed in Unicode Sequence
- Pitch and Duration of Tones
- Homographs
- Supported SAPI Functions

Phonemic Symbols

Table 4-1 lists the phonemic symbols DECtalk Software uses, along with an example of each sound. Some dictionaries put the stress symbol after the vowel nucleus or at the start of the syllable. DECtalk Software requires that the stress symbol appear immediately before a syllable nucleus. Table 4-1 lists the supported stress symbols.

The phonemic symbol can be used to replace words that are spoken incorrectly. For more information on how to use phonemes, refer to the **Phoneme Interpretation** command description in Chapter 2. Phonemes can also be given duration and pitch attributes to create special effects, such as singing. See Table 4-1 for additional information.

Note

Arpabet mode is a 2-character system. All single character symbols must be followed by a space so that faulty translations do not occur. Consider the phonemic representation of “whitehorse,” [ˈ w ʰ aɪ t h x oʊ r s]. The letter “t” in this phonemic representation must be followed by a space, so that it is not interpreted as part of the phonemic symbol [th] in the representation of “whitehorse.”

Some older versions of DECtalk Software supported single characters in arpabet mode. Application programs written for use with those versions may fail to function correctly when used with DECtalk Software V4.6 and subsequent versions.

Table 4-1 Phonemic Symbols

ASCKY	DT index	DT internal	Example	Arpa- bet	Uni- code	Unicode Character Name
_	0	SIL	(silence)	_	U+5F	Low line
i	1	IY	bEAn	iy	U+69	Latin small letter I
l	2	IH	plt	ih	U+26A	Latin small letter Capital I
e	3	EY	bAY	ey	U+65	Latin small letter E
E	4	EH	pEt	eh	U+25B	Latin small letter open E
@	5	AE	pAt	ae	U+E6	Latin small letter AE
a	6	AA	pOt	aa	U+251	Latin small letter Alpha
A	7	AY	bUY	ay	U+61, U+26A	Latin small letter A + Latin small capital I
W	8	AW	brOW	aw	U+61, U+28A	Latin small letter A + Latin small capital Upsilon
^	9	AH	pUtt	ah	U+28C	Latin small letter turned V
c	10	AO	bOUght	ao	U+254	Latin small letter O
o	11	OW	nO	ow	U+6F, U+28A	Latin small letter O + Latin small letter Upsilon
O	12	OY	bOY	oy	U+254, U+26A	Latin small letter open O + Latin small letter capital I
U	13	UH	pUt	uh	U+28A	Latin small letter Upsilon
u	14	UW	bOOOn	uw	U+75	Latin small letter U
R	15	RR	anothER	rr	U+25A	Latin small letter Schwa with hook
Y	16	YU	cUte	yu	U+6A, U+75	Latin small letter J + Latin small letter U
x	17	AX	About	ax	U+259	Latin small letter Schwa
	18	IX	kissEs	ix	U+268	Latin small letter I with stroke
l	19	IR	pEEr	ir	U+69, U+2B4	Latin small letter I + modifier letter small turned R
R	20	ER	pAlr	er		
a	21	AR	bARn	ar	U+251, U+2B4	Latin small letter Alpha + modifier letter small turned

ASCKY	DT index	DT internal	Example	Arpa- bet	Uni- code	Unicode Character Name
						R
c	22	OR	bOrn	or	U+254, U+2B4	Latin small letter open O + modifier letter small turned R
U	23	UR	pOOr	ur	U+28A, U+2B4	Latin small letter Upsilon + modifier letter small turned R
w	24	W	Why	w	U+77	Latin small letter W
Y	25	Y	Yank	yx	U+6A	Latin small letter J
r	26	R	Rat	r	U+52	Latin capital letter R
l	27	LL	Lad	l	U+6C	Latin small letter L
h	28	HX	Had	hx	U+68	Latin small letter H
R	29	RX	coRe	rx	U+279	Latin small letter turned R with hook
l	30	LX	until	lx	U+26B	Latin small letter l with middle tilde
m	31	M	Mad	m	U+6D	Latin small letter M
n	32	N	Nat	n	U+6E	Latin small letter N
G	33	NX	baNG	nx	U+14B	Latin small letter Eng
L	34	EL	dangLe	el	U+6C, U+329	Latin small letter L combining vertical line below
D	35	DZ	wiDth	dz	U+64, U+32F	Latin small letter D + combining inverted breve below
N	36	EN	burdeN	en	U+6E, U+329	Latin small letter N + combining vertical line below
f	37	F	Fat	f	U+66	Latin small letter F
v	38	V	Vat	v	U+76	Latin small letter V
T	39	TH	THin	th	U+3B8	Greek small letter Theta
D	40	DH	THen	dh	U+F0	Latin small letter Eth
s	41	S	Sap	s	U+73	Latin small letter S
z	42	Z	Zap	z	U+7A	Latin small letter Z

ASCII	DT index	DT internal	Example	Arpa-bet	Uni-code	Unicode Character Name
S	43	SH	SHeep	sh	U+283	Latin small letter Esh
Z	44	ZH	meaSure	zh	U+292	Latin small letter Ezh
p	45	P	Pat	p	U+70	Latin small letter P
b	46	B	Bad	b	U+62	Latin small letter B
t	47	T	Tack	t	U+74	Latin small letter T
d	48	D	Dad	d	U+64	Latin small letter D
k	49	K	Cad	k	U+6B	Latin small letter K
g	50	G	Game	g	U+67	Latin small letter G
&	51	DX	riDer	dx	Internal use only	
Q	52	TX	baTTen	tx	U+74, U+294	Latin small letter T + Latin letter glottal stop
q	53	Q	we eat	q	U+294	Latin letter glottal stop
C	54	CH	CHeap	ch	U+2A7	Latin small letter Tesh digraph
J	55	JH	Jeep	jh	U+2A4	Latin small letter Dezh digraph
F	56	DF	wriTer	df	Internal use only	

Stress and Syntactic Symbols

Table 4-1 and Table 4-2 list the stress and syntactic symbols supported by DECtalk Software. Phoneme interpretation must be turned on for the stress and syntactic symbols to work. Refer to the **Phoneme Interpretation** command description in Chapter 2 for more information.

Table 4-1 Stress Symbols

Symbol	Name	Indicates	Unicode
'	Apostrophe	primary stress	U+27
`	Grave accent	secondary stress	U+60
"	Quotation mark	emphatic stress	U+22
/	Slash	pitch rise	U+2F
\	Backslash	pitch fall	U+5C

Table 4-2 Syntactic Symbols

Symbol	Name	Indicates	Unicode
-	Hyphen	syllable boundary	U+2D
*	Asterisk	morpheme boundary	U+2A
#	Number sign	compound nouns	U+23
(Open parenthesis	beginning of prepositional phrase	U+28
)	Close parenthesis	beginning of a verb phrase	U+29
,	Comma	clause boundaries	U+2C
.	Period	period	U+2E
?	Question mark	question mark	U+2F
!	Exclamation point	exclamation point	U+21
+	Plus sign	new paragraph	U+2B
	Space	word boundary	U+20

Phonemes Listed in Unicode Sequence

Table 4-1 Phonemes in Unicode Sequence

Unicode	Unicode Character Name	ASCKY	DT index	DT internal	Example	Arpabet
U+20	Space				Word boundary	<space>
U+21	Exclamation point					
U+22	Quotation mark	“			“Hello”	“
U+23	Number sign	#				
U+27	Apostrophe	“			r’ehd	
U+28	Left parenthesis	(
U+29	Right parenthesis)				
U+2A	Asterisk	*				
U+2B	Plus sign	+				
U+2C	Comma	,				
U+2D	Hyphen	-				
U+2E	Full stop	.			Syllable break	-
U+2F	Solidus	/				
U+3F	Question mark	?				
U+52	Latin capital letter R	R	26	R	Rat	r
U+5C	Reverse solidus	\				
U+5F	Low line	—	0	SIL	(silence)	—
U+61, U+26A	Latin small letter A + Latin small capital I	A	7	AY	bUY	ay
U+61, U+28A	Latin small letter A + Latin small capital I	W	8	AW	brOW	aw
U+62	Latin small letter B	b	46	B	Bad	b
U+64, U+32F	Latin small letter D + combining inverted breve below	D	35	DZ	WiDth	dz
U+64	Latin small letter D	d	48	D	Dad	d
U+65	Latin small letter E	e	3	EY	bAY	ey
U+66	Latin small letter F	f	37	F	Fat	f

Unicode	Unicode Character Name	ASCKY	DT index	DT internal	Example	Arpabet
U+67	Latin small letter G	g	50	G	Game	g
U+68	Latin small letter H	h	28	HX	Had	hx
U+69, U+2B4	Latin small letter I + modifier letter small turned R	I	19	IR	pEEr	ir
<u>U+69</u>	Latin small letter I	i	1	IY	bEAn	iy
U+6A, U+75	Latin small letter J + Latin small letter U	Y	16	YU	cUte	yu
U+6A	Latin small letter J	Y	25	Y	Yank	yx
U+6B	Latin small letter K	k	49	K	Cad	k
U+6C, U+329	Latin small letter L + combining vertical line below	L	34	EL	dangLe	el
U+6C	Latin small letter L	l	27	LL	Lad	l
U+6D	Latin small letter M	m	31	M	Mad	m
U+6E, U+329	Latin small letter N + combining vertical line below	N	36	EN	burdeN	en
U+6E	Latin small letter N	n	32	N	Nat	n
U+6F, U+28A	Latin small letter O + Latin small letter upsilon	o	11	OW	nO	ow
U+70	Latin small letter P	p	45	P	Pat	p
U+73	Latin small letter S	s	41	S	Sap	s
U+74	Latin small letter T	t	47	T	Tack	t
U+74, U+294	Latin small letter T + Latin letter glottal stop	Q	52	TX	baTTen	tx
U+75	Latin small letter U	u	14	UW	bOOOn	uw
U+76	Latin small letter V	v	38	V	Vat	v
U+77	Latin small letter W	w	24	W	Why	w
U+7A	Latin small letter Z	z	42	Z	Zap	z
U+E6	Latin small letter AE	@	5	AE	pAt	ae
U+F0	Latin small letter Eth	D	40	DH	THen	dh

Unicode	Unicode Character Name	ASCKY	DT index	DT internal	Example	Arpabet
U+14B	Latin small letter Eng	G	33	NX	baNG	nx
U+251, U+2B4	Latin small letter Alpha + modifier letter small turned R	a	21	AR	bARn	ar
U+251	Latin small letter Alpha	a	6	AA	pOt	aa
U+254, U+26A	Latin small letter open O + Latin small letter capital I	O	12	OY	bOY	oy
U+254, U+2B4	Latin small letter open O + modifier letter small turned R	c	22	OR	bOrn	or
U+254	Latin small letter O	c	10	AO	bOUght	ao
U+259	Latin small letter Schwa	x	17	AX	About	ax
U+25A	Latin small letter Schwa with hook	R	15	RR	anothER	rr
U+25B	Latin small letter open E	E	4	EH	pEt	eh
U+268	Latin small letter I with stroke		18	IX	kissEs	ix
U+26A	Latin small letter Capital I	l	2	IH	plt	ih
U+26B	Latin small letter I with middle tilde	l	30	LX	untiL	lx
U+279	Latin small letter turned R with hook	R	29	RX	coRe	rx
U+283	Latin small letter Esh	S	43	SH	SHeep	sh
U+28A, U+2B4	Latin small letter Upsilon + modifier letter small turned R	U	23	UR	pOOOr	ur
U+28A	Latin small letter Upsilon	U	13	UH	pUt	uh
U+28C	Latin small letter turned V	^	9	AH	pUtt	ah
U+292	Latin small letter Ezh	Z	44	ZH	meaSure	zh
U+294	Latin letter glottal stop	q	53	Q	we eat	q
U+2A4	Latin small letter Dezh digraph	J	55	JH	Jeep	jh

Unicode	Unicode Character Name	ASCKY	DT index	DT internal	Example	Arpabet
U+2A7	Latin small letter Tesh digraph	C	54	CH	CHeap	ch
U+2C8	Modifier letter vertical line	‘				
U+28CC	Modifier letter low vertical line	`				
U+3B8	Greek small letter Theta	T	39	TH	THin	th

Pitch and Duration of Tones

DECTalk Software can be used to sing songs or make various sounds associated with singing and musical tones. Table 4-2 provides the pitch numbers, associated notes, and frequencies you need to code a phonemic sequence to produce musical sounds.

Figure 4-1 is the code for the song, “Happy Birthday.” The command syntax for coding musical sequences is found in Table 4-1. You can use Table 4-1 to decode the phoneme symbols.

Table 4-1 Phoneme Syntax for Singing

SYNTAX:	[<i>phoneme</i> <duration, pitch number>]	
OPTIONS:	none	
PARAMETERS:	duration	Tone duration in milliseconds.
	pitch number	Pitch number from
DEFAULT:	none	
EXAMPLES:	See Figure 4-1	

Figure 4-1 DECTalk Software Singing “Happy Birthday”

```
[ :phoneme arpabet speak on]
[hxae<300,10>piy<300,10> brr<600,12>th<100>dey<600,10>
tuw<600,15> yu<1200,14>_<120>]
[hxae<300,10>piy<300,10> brr<600,12>th<100>dey<600,10>
tuw<600,17> yu<1200,15>_<120>]
[hxae<300,10>piy<300,10> brr<600,22>th<100>dey<600,19>
dih<600,15>r deh<600,14>ktao<600,12>k_<120>_<120>]
[hxae<300,20>piy<300,20> brr<600,19>th<100>dey<600,15>
tuw<600,17> yu<1200,15>]
```

Table 4-2 Tone Table

Pitch Number	Note	Pitch	Vocal Ranges				
1	C2	65					
2	C#	69					
3	D	73					
4	D#	77					
5	E	82	B				
6	F	87	A				
7	F#	92	S				
8	G	98	S	B			
9	G#	103		A			
10	A	110		R			
11	A#	116		I			
12	B	123		T			
13	C3	130		O	T		
14	C#	138		N	E		
15	D	146		E	N		
16	D#	155			O		
17	E	164			R		
18	F	174				A	
19	F#	185				L	
20	G	196				T	
21	G#	207				O	
22	A	220					
23	A#	233					
24	B	247					S
25	C4	261					O
26	C#	277					P
27	D	293					R
28	D#	311					A
29	E	329					N
30	F	348					O
31	F#	370					
32	G	392					
33	G#	415					
34	A	440					
35	A#	466					
36	B	494					
37	C5	523					

Homographs

Homographs are two or more words that have the same spelling but are pronounced differently. Homographs are often different in terms of which syllable is accented. For example, if *permit* is a noun, the accent is on the first syllable (*per*mit); if, however, the word is used as a verb, the accent is on the second syllable (per*mit*). This distinction often makes a great deal of difference in understanding DECtalk when it is speaking such words in connected discourse.

The default pronunciation is the more frequent form. In the event the alternate pronunciation is needed, you can insert the correct phonetics from the homograph index below. Use the **[pronounce alternate]** command to obtain an alternative pronunciation for a word. For example, the primary pronunciation of the word *bass* is b'ays, as in bass guitar, while the alternate pronunciation, denoted by **[pronounce alternate]**, is b'aes, as in the fish, bass.

Homograph Phonetics - (A)

Use the **[;pronounce alternate]** command before a word to obtain an alternative pronunciation for the word. Table 4-1 lists the homograph phonetics for words beginning with the letter A.

Table 4-1 Homograph Phonetics - (A)

Spelling	Primary	Alternate
abstract	' aeb s t r aek t	ae b s t r ' aek t
abuse	axb y ' u z	axb y ' u s
addict	ax d ' i h k t	' ae d i h k t
advocate	' aed v axk eyt	' aed v ax k axt
affix	' aef i h k s	axf ' i h k s
ally	' ael ay	axl ' ay
alternate	' aol t r m ax t	' ao l t r m ey t
animate	' aen i h m eyt	' aen i h m ax t
annex	' aen ehk s	axn ' ehk s
appropriate	axp r ' owp r i yaxt	axp r ' owp r i y eyt
arithmetic	axr ' i h t h m ax t i x k	aer i x t h m ' eht i x k
articulate	aar t ' i h k y xel eyt	aar t ' i h k y xel axt
associate	axs ' owshiyeyt	axs ' owshiyaxt
attribute	axt r ' i h byut	' aet r i x byut
august	' aog axs t	aog ' ahs t

Homograph Phonetics - (B-C)

Use the **[;pronounce alternate]** command before a word to obtain an alternative pronunciation for the word. Table 4-1 lists homograph phonetics for words beginning with the letters B and C.

Table 4-1 Homograph Phonetics - (B-C)

Spelling	Primary	Alternate
bass	b ' eys	b ' aes
baton	b ax t ' aon	b ' aet ax n
close	k l ' owz	k l ' ows
combat	k axm b ' aet	k ' aam b ae t
combine	k axm b ' ayn	k ' aam b ayn
compact	k axm p ' aek t	k ' aam pae k t
complex	k ' aam p l ehk s	k axm p l ' ehk s
compound	k ' aam paw n d	k axm p ' aw n d
compress	k ax m p r ' ehs	k ' aam p r ehs
concert	k ' aan s rrt	k axn s ' rrt
conduct	k axn d ' ahk t	k ' aa n d ahk t
confederate	k axn f ' ' ehd rrixt rreyt	k axn f ' ehd rriht
confine	k axn f ' ayn	k ' aan f ayn
conflict	k ' aan f l ixk t	k axn f l ' eyk t
conglomerate	k axn xg l ' aam rixt	k axn xg l ' aam rreyt
console	k ' aan s owl	k axn s ' owl
construct	k axn s t r ' ahk t	k ' aan s t r axk t
content	k ' aan t eh n t	k axn t ' eh n t
contest	k ' aan t ehs t	k axn t ' ehs t
contract	k ' aan t rae k t	k axn t r ' aek t
contrast	k ' aan t r aes t	k axn t r ' aes t
converse	k ' aan v rrs	k axn v ' rrs
convert	k axn v ' rrt	k ' aan v rrt
convict	kax n v ' ihk t	k ' aan vih k t
coordinate	k ow' aor d en eyt	kow' aor d ixn ax t

Homograph Phonetics - (D-G)

Use the **[;pronounce alternate]** command before a word to obtain an alternative pronunciation for the word. Table 4-1 lists homograph phonetics for words beginning with the letters D through G.

Table 4-1 Homograph Phonetics - (D-G)

Spelling	Primary	Alternate
decrease	d iyk r ' iys	d ' iyk r iys
defect	d ax f ' ehk t	d ' iyf ehk t
delegate	d ' ehl ixg axt	d ' ehl ixg ` ey t
deliberate	d axl ' ihb rraxt	d axl ' ihb rreyt
desert	d ' ehz rrt	d ixz ' rrt
desolate	d ' ehs el ixt	d ' eh sel yet
Diffuse	dix f ' yuw s	d ix f ' yuw z
digest	d ' ayjhehs t	d ayjh ' ehs t
discharge	d ix s ch ' arjh	d ' his charjh
discount	d ' ihs kaw n t	d his k ' awn t
dove	d ' owv	d ' ahv
duplicate	d ' uwp l ixk eyt	d ' uwp lixk axt
elaborate	axl ' aeb rraxt	axl ' aeb rreyt
estimate	' ehs tix m eyt	' ehs tix m axt
excerpt	' ehksrrpt	ehks'rrpt
excuse	ixk s k ` yuz	eh k s k ' yus
expatriate	ehk s p ' yet riy axt	ehk s p ' ey t riieyt
exploit	ixk s p l ' oyt	' ehk s p loy t
export	ehk s p ' ort	' ehk s por t
extract	ehk s t r ' aek t	' eh k s t raek t
ferment	frr m ' ehn t	f ' rrm eh n t
frequent	f r ' iyk wix n t	f riy k w ' eyn t
geminate	jh ' ehm ixn axt	jh ' ehm ixn eyt
graduate	g r ' aejhuweyt	g r ' aejhuwaxt

Homograph Phonetics - (I-L)

Use the **[;pronounce alternate]** command before a word to obtain an alternative pronunciation for the word. Table 4-1 lists homograph phonetics for words beginning with the letters I through L.

Table 4-1 Homograph Phonetics - (I-L)

Spelling	Primary	Alternate
impact	' ihm paek t	ixm p ' aek t
implant	ihm p l ' aen t	' ihm p l aen t
import	' ihm p ort	ihm p ' ort
imprint	' ihm p r ihnt	ihm p r ' ihn t
incense	ixn s ' ehns	' ihn s ehns
incline	ixn k l ' ayn	' ihn k l ayn
increase	ihn k r ' iys	' ihn k r iys
insert	ihn s ' rrt	' ihn s rrt
insult	ihn s ' ahl t	' ihn s axl t
interchange	' ihn t rr ch eyn jh	ihn t rr ch ' eyn jh
intimate	' ihn t axm axt	' ihn t axm eyt
invalid	ixn v ' ael ixd	' ihn v axl ixd
just	jh ixs t	jh ' ahs t
lead	l ' iyd	l ' ehd
live	l ' ihv	l ' ayv

Homograph Phonetics - (M-P)

Use the **[;pronounce alternate]** command before a word to obtain an alternative pronunciation for the word. Table 4-1 lists homograph phonetics for words beginning with the letters M through P.

Table 4-1 Homograph Phonetics - (M-P)

Spelling	Primary	Alternate
minute	m ' ih nix t	may n ' uwt
miscount	m ' ihs kaw n t	mih s k ' awn t
misprint	m ' ihs p r in t	mih s pr ' int
misuse	mix s ' yuz	mix s ' yus
moderate	m ' aad rraxt	m ' aad rreyt
object	' aa b jheht	ax b jh ' ehkt
overrun	' ow v rr rahn	ow v r rr'ahn
perfect	p ' rr f ixk t	prrf ' ehk t
permit	prr m ' iht	p ' rr miht
pervert	p rrv ' rrt	p ' rrv rrt
polish	p ' aal hish	p ' owl ixsh
postulate	p ' aas cheleyt	p ' aas chelaxt
predicate	p r ' ehd ixk eyt	p r ' ehd ixk axt
predominate	p r ixd ' aam ixn eyt	p r ixd ' aam ixn axt
present	p riy z ' ehn t	p r ' ehz axn t
proceed	p r axs ' iyd	p r ' ows iyd
produce	p r axd ' uws	p r ' aad uws
progress	p r ' aag r ehs	p rax g r ' eh s
project	p r ' aajh ehk t	p r axjh ' ehk t
protest	p r ' owt ehs t	p r owt ' ehs t

Homograph Phonetics - (R)

Use the **[;pronounce alternate]** command before a word to obtain an alternative pronunciation for the word. Table 4-1 lists homograph phonetics for words beginning with the letter R.

Table 4-1 Homograph Phonetics - (R)

Spelling	Primary	Alternate
read	r ' iyd	r ' ehd
reading	r ' iyd ixnx	r ' ehd ixnx
rebel	r ' ehb el	rix b ' ehl
recall	rix k ' aol	r ' iyk aol
recap	riy k ' aep	r ' iyk aep
recess	r ' iys ehs	r iys ' ehs
record	r ' ehk rrd	r ixk ' ord
recount	r iyk ' awn t	r ' iyk awn t
refill	r ' iyf ihl	r iyf ' ihl
refresh	r iyf r ' ehsh	r ' iyf r ehsh
refund	r iyf ' ahn d	r ' iyf ahn d
refuse	r ix f ' yuz	r ' ehf yus
reject	rixjh'ehkt	r'iyjh'ehkt
relapse	r ' iyl aep s	r ixl ' aep s
relay	r ' iyl ey	r ixl ' ey
remake	r ' iym eyk	r iym ' eyk
rerun	r ' iy * rahn	r iy * r ' ahn
research	r ' iys rrch	r iys ' rrch
resume	r iy z ' uwm	r ' ehz axm ey
retake	r iyt ' eyk	r ' iyt eyk
rewrite	r iy r ' ayt	r ' iy * r ayt

Homograph Phonetics - (S-W)

Use the **[;pronounce alternate]** command before a word to obtain an alternative pronunciation for the word. Table 4-1 lists homograph phonetics for words beginning with the letters S through W.

Table 4-1 Homograph Phonetics - (S-W)

Spelling	Primary	Alternate
segment	s ' ehg m ixn t	s ehg m ' ehn t
separate	s ' eh p axr eyt	s ' eh p axr axt
sow	s 'ow	s 'aw
subject	s ' ahb jhehk t	s axb jh ' ehk t
sublet	s axb l ' eht	s axb l ' eht
subordinate	s axb ' ord enaxt	s axb ' ord eneyt
survey	s ' rr vey	s rr v ' ey
suspect	s ' ahs peh k t	s ax s p ' eh k t
syndicate	s ' ihn dix kix t	s ' ihn dix key t
tear	t ' er	t ' ir
torment	t orm ' eh n t	t ' orm eh n t
transform	t r aen s f ' orm	t r ' aen s f orm
transplant	t r aen s p l ' aen t	t r ' aen s p l aen t
transport	t r aen s p ' ort	t r ' aen s p ort
upset	axp s ' eht	' ah p she t
use	y ' uwz	y ' uws
wind	w ' ihn d	w ' ayn d
wound	w ' awn d	w ' uwn d

Supported SAPI Functions

Table 4-1 shows the Microsoft SAPI functions that DECtalk supports. See the Microsoft documentation and the Microsoft web site for more information on the SAPI functions.

Table 4-1 Supported Functions of the Microsoft Speech API

SAPI Interface	Functions Supported	Functions With Limited Support	Unsupported Functions
ITTSAttributes			
	SpeedGet	RealTimeGet (always returns 1)	
	SpeedSet	RealTimeSet (value ignored)	
	VolumeGet		
	VolumeSet		
	PitchGet		
	PitchSet		
ITTSBufNotifySinc			
	TextDataStart		
	TextDataDone		
	BookMarks		
	WordPosition		
ITTSCentral			
	AudioPause		
	AudioReset		

SAPI Interface	Functions Supported	Functions With Limited Support	Unsupported Functions
	AudioResume		
	ModeGet		
	PosnGet		
	TextData \com=string\ \emp\ \mrk=number\ \pau=number\ \pit=number\ \rst\ \spd=number\ \vol=number\ \prm=string=string\	TextData \chr=string [[,string...] \ctx=string\ \eng[[:GUID]] : command\ \pro=number\ \prt=string\ \vce=character=value [[,character=value...]]	
	Register	Phoneme	
	Unregister		
	Inject		
	ToFileTime		
ITTSDialogs	All functions		None
ITTSEnum	All functions		None
ITTSFind	None		All functions
IllexPronounce	None		All functions
ITTSNotifySinc			
	AudioStart	AttribChanged	
	AudioStop		
laudio	All functions		None

SAPI Interface	Functions Supported	Functions With Limited Support	Unsupported Functions
audioDest	All functions		None
audioDestNotify	All functions		None
audioSourceNotifySink	None		All functions

Notes

- DECTalk inline commands in the SAPI interfaces are ignored. They are not supported.
- DECTalk Software for Windows CE does not support SAPI functions.
- SAPI tags embedded within a word (for example, th\mrk=1\is) are not supported. DECTalk does not hang, but it splits the word apart.
- The DECTalk SAPI interfaces return status indicating support for **VolumeSet** and **VolumeGet**. However, if the underlying audio-file destination object passed to the synthesizer does not support the **LevelSet** and **LevelGet** functions, DECTalk returns the status for handling a **VolumeSet** or **VolumeGet** call. The standard audio-file destination object returns E_NOTIMPL in this case.
- Both the ANSI and the UNICODE versions are supported, where applicable, unless otherwise noted in this table.
- Only the UNICODE version of **TextData** with CHARSET_IPAPHONETIC is supported.
- All dialog boxes that SAPI defines use English, even if the DECTalk synthesizer is running in another language, such as German.

Chapter 5 — Customizing a DECtalk Software Voice

The DECtalk Software built-in voices provide an adequate selection for most applications. However, if you have a special application requiring a monotone or unusual voice, you can use the **Design Voice** command to modify the options provided in this section to design your own voice. For information on all other commands, refer to Chapters 2 and 3.

Topics Include:

- Design Voice Command [:dv]
- Definitions of DECtalk Software Voices
- Changing Gender and Head Size
- Changing Voice Quality
- Changing Pitch and Intonation
- Changing Relative Gains and Avoiding Overloads
- Saving Changes as Val's Voice
- Summary of Design Voice Options

Design Voice [:dv]

The nine built-in voices of DECTalk Software are distinguished from one another by a large set of speaker-definition options. Note that there is a tenth voice, called Val. Val is initialized with the same voice as Paul, but can be used to save voice changes. Unlike the nine built-in voices that can be modified but not saved, Val can be used to store voice changes during a DECTalk Software session.

Speakers can differ in sex, age, head size and shape, larynx size and behavior, pitch range, pitch and timing habits, dialect, and emotional state. DECTalk Software cannot approximate all of these options. Therefore, the range of distinguishable voices is limited, even though DECTalk Software has many speaker-definition options that can be modified.

The **Design Voice [:dv]** command introduces the speaker-definition options and parameters that can be entered as a string or one at a time.

The following sections discuss speech production, acoustics, and perception. Some of the information is relatively technical, but the examples should make it possible for all developers to modify any option effectively and listen to the results.

Table 5-1 [:dv] Command Options

SYNTAX:	[:dv XX YY]	
OPTIONS:	ap	Average pitch, in Hz
	as	Assertiveness, in %
	b4	Fourth formant bandwidth, in Hz
	b5	Fifth formant bandwidth, in Hz
	bf	Baseline fall, in Hz
	br	Breathiness, in decibels (dB)
	f4	Fourth formant resonance frequency, in Hz
	f5	Fifth formant resonance frequency, in Hz
	g1	Gain of cascade formant resonator 1, in dB
	g2	Gain of cascade formant resonator 2, in dB
	g3	Gain of cascade formant resonator 3, in dB
	g4	Gain of cascade formant resonator 4, in dB
	g5	Loudness of the voice, in dB
	gf	Gain of frication source, in dB
	gh	Gain of aspiration source, in dB

gn	Gain of nasalization, in dB
gv	Gain of voicing source, in dB
hr	Hat rise, in Hz
hs	Head size, in %
la	Laryngealization, in %
lx	Lax breathiness, in %
nf	Number of fixed samples of open glottis
pr	Pitch range, in %
qu	Quickness, in %
ri	Richness, in %
sm	Smoothness, in %
sr	Stress rise, in Hz
sx	Sex 1 (male) or 0 (female)
save	Save the current speaker-definition options as Val's voice.

PARAMETERS: See the individual options for detailed information about valid parameter values

EXAMPLES: [:np][:dv ap 100] Change Paul's average pitch to be 100.

Definitions of DECtalk Software Voices

Table 5-1 Speaker Definitions for All DECtalk Software Voices

Param	Paul	Harry	Frank	Dennis	Betty	Ursula	Wendy	Rita	Kit
ap	122	89	155	110	208	240	200	106	306
as	100	100	65	100	35	100	50	65	65
b4	260	200	280	240	260	260	400	250	2048
b5	330	240	300	280	2048	2048	2048	2048	2048
bf	18	9	9	9	0	8	0	0	0
br	0	0	50	38	0	0	55	46	47
f4	3300	3300	3650	3200	4450	4450	4500	4000	2500
f5	3650	3850	4200	3600	2500	2500	2500	2500	2500
g1	68	71	63	75	69	67	69	69	69
g2	60	60	58	60	65	65	62	72	69
g3	48	52	56	52	50	51	53	48	52
g4	64	62	66	61	56	58	55	54	50
g5	86	81	86	84	81	80	83	83	73
gf	70	70	68	68	72	70	70	72	72
gh	70	70	68	68	70	70	68	70	70
gn	74	73	75	76	72	80	75	73	71
gv	65	65	63	63	65	65	51	65	65
hr	18	20	20	20	14	20	20	20	20
hs	100	115	90	105	100	95	100	95	80
la	0	0	5	0	0	0	0	4	0
lx	0	0	50	70	80	50	80	0	75
nf	0	10	0	10	0	10	10	0	0
pr	100	80	90	135	240	135	175	80	210
qu	40	10	0	50	55	30	10	30	50
ri	70	86	40	0	40	100	0	20	40
sm	3	12	46	100	4	60	100	24	5
sr	32	30	22	22	20	32	22	32	22
sx	1	1	1	1	0	0	0	0	0

Changing Gender and Head Size

Six speaker-definition options control the size and shape of the head. These options are listed in Table 5-1.

Table 5-1 Head Size and Shape Options

sx	Sex 1 (male) or 0 (female)
hs	Head size, in %
f4	Fourth formant resonance frequency, in Hz
f5	Fifth formant resonance frequency, in Hz
b4	Fourth formant bandwidth, in Hz
b5	Fifth formant bandwidth, in Hz

Sex, **sx**

Male and female voices differ in many ways, including head size, pharynx length, larynx mass, and speaking habits such as degree of breathiness, liveliness of pitch, choice of articulatory target values, and speed of articulation. Some of these differences are under the control of a single option, **sx**, the sex of the speaker. Speakers Paul, Harry, Frank, and Dennis are male (**sx** = 1), while speakers Betty, Rita, Ursula, Wendy, and Kit are female (**sx** = 0). Actually, Kit can be male or female because children of both sexes younger than 10 years old have similar voices.

Changing the Sex (**sx**) option causes DECTalk Software to access a different (male or female) table of target values for formant frequencies, bandwidths, and source amplitudes. The male and female tables are patterned after two individuals who were judged to have pleasant, intelligible voices. The built-in voices of DECTalk Software are simply scaled transformations of Paul and Betty, the two basic voices.

You can change the sex of any DECTalk Software voice by making the voice current and then modifying the **sx** option. For example, the following command gives Paul some of the speaking characteristics of a woman. (The **sx** option does not change the average pitch or breathiness, so a peculiar combination of simultaneous male and female traits results from this **sx** change.)

```
[ :np ] [ :dv sx 0 ] Am I a man or woman?
```

The **sx** option can also be specified as **m** or **f** with the commands **[:dv sx m]** or **[:dv sx f]**.

Note

If you change the sex of the voice, some phonemes might cause DECTalk Software's filters to overload, producing a squawk. The modification of certain options such as **f4**, **f5**, and **g1** can help to correct this problem.

Head Size, **hs**

The Head size (**hs**) option is specified as the average size for an adult man (if **sx** = 1) or an adult woman (if **sx** = 0). A head size of 100% is normal or average for a given sex, but people can differ significantly in this characteristic. Head size has a strong influence on a person's voice. Large musical instruments produce low notes, and humans with large heads tend to have low, resonant voices. For example, to make Paul sound like a larger man with a 15% longer vocal tract (and formant frequencies that are scaled down by a factor of about 0.85%), use the following command:

```
[ :np ][ :dv hs 115 ] Do I sound more like huge Harry this way?
```

Head size is one of the best variables to use if you want to make dramatic voice changes. For example, Paul has a head size of 100, while Harry's deep voice is caused in part by a head-size change to 115, or 15% greater than normal. Decreasing head size produces a higher voice, such as in a child or adolescent. Extreme changes in head size, as in the following examples, are somewhat difficult to understand.

```
[ :nh ][ :dv hs 135 ] Do I have a swelled head?
```

```
[ :nk ] I am about 10 years old.
```

```
[ :nk ][ :dv hs 65 ] Do I sound like a six year old?
```

Note

Extreme changes in head size can cause overloads, as well as difficulties in understanding the speech. The modification of certain options such as **f4**, **f5**, and **g1** can help to correct this problem.

Higher Formants, **f4**, **f5**, **b4**, and **b5**

A male voice typically has five prominent resonant peaks in the spectrum (over the range from 0 to 5 kHz), a female voice typically has only four (because of a smaller head size), and a child has three. If fourth and fifth formant resonances exist for a specific voice, they are fixed in frequency and bandwidth characteristics. These characteristics are specified in Hz by the options **f4**, **f5**, **b4**, and **b5**.

If a higher formant does not exist, the frequency and bandwidth of the speaker definition are set to special values that cause the resonance to disappear. To make a resonance disappear, the frequency is set to above 5500 Hz and the bandwidth is set to

5500 Hz. (This disables the formant filter.) This is what has been done to the fourth and fifth formants for Kit.

The permitted values for the **f4** and **f5** options have fairly complicated restrictions. Violating these restrictions can cause overloads and squawks. The following restrictions apply to cases where a higher formant exists:

- The **f5** option must be at least 300 Hz higher than **f4**.
- If **sx** is 1 (male), **f4** must be at least 3250 Hz.
- If **sx** is 0 (female), **f4** must be at least 3700 Hz.
- If **hs** is not 100, the preceding values should be multiplied by (**hs** / 100).

These higher formants produce peaks in the spectrum that become more prominent if the **b4** and **b5** options are smaller, and if the **f4** and **f5** options are closer together. The limits placed on the **b4** and **b5** options should ensure that no problems occur. However, smaller values for bandwidths may produce an overload in the synthesizer. You can correct these overloads by increasing the bandwidths or by changing the gain control, **g1**.

Changing Voice Quality

Six speaker-definition options control aspects of the output of the larynx, which, in turn, control voice quality. These options are listed in Table 5-1.

Table 5-1 Voice Quality Options

br	Breathiness, in decibels (dB)
lx	Lax breathiness, in %
sm	Smoothness, in %
ri	Richness, in %
nf	Number of fixed samples of open glottis
la	Laryngealization, in %

Breathiness, br

Some voices can be characterized as breathy. The vocal folds vibrate to generate voicing and breath noise simultaneously. Breathiness is a characteristic of many female voices, but it is also common under certain circumstances for male voices.

The range of the Breathiness (**br**) option is from 0 dB (no breathiness) to 70 dB (strong breathiness). By experimenting, you can learn what intermediate values sound like. For example, to turn Paul into a breathy, whispering speaker, use the following commands:

```
[ :np] [ :dv br 55 gv 56] Do I sound more like Dennis now?
```

This voice is not as loud as the others, because of the simultaneous decrease in the gain of voicing, **gv**, but it is intelligible and human sounding.

Lax Breathiness, lx

The **br** option creates simultaneous breathiness whenever voicing is turned on. Another type of breathiness occurs only at the ends of sentences and when going from voiced to voiceless sounds. This type of breathiness is controlled by the Lax breathiness (**lx**) option in percentage values.

A nonbreathy, tense voice would have the **lx** option set to 0, while a maximally breathy, lax voice would be set to 100. The difference between these two voices is not great, but you can hear it if you listen closely.

Smoothness, **sm**

The Smoothness (**sm**) option refers to vocal fold vibrations. The vocal folds meet at the midline, as they do in normal voicing, but they do not slam together forcefully to create a very sudden cessation of airflow.

DECTalk Software uses a variable-cutoff, gradual low-pass filter to model changes to smoothness. The range of **sm** is from 0% (least smooth and most brilliant) to 100% (most smooth and least brilliant). The voicing source spectrum is tilted so that energy at higher frequencies is attenuated by as much as 30 dB when smoothness is set to the maximum but is not attenuated at all when smoothness is set to 0.

Professional singing voices that are trained to sing above an orchestra are usually brilliant, while anyone who talks softly becomes breathy and smooth. To synthesize a breathy voice, having the **sm** option set to 50 or more is good. Changes to smoothness do not have a great effect on perceived voice quality.

Richness, **ri**

The Richness (**ri**) option is similar to smoothness and brilliance except that the spectral change occurs at lower frequencies. The spectral change difference is because of a different physiological mechanism. Brilliant, rich voices carry well and are more intelligible in noisy environments, while smooth, soft voices sound more friendly. For example, the following command produces a soft, smooth version of Paul's voice:

```
[ :np ] [ :dv ri 0 sm 70 ] Do I sound more mellow?
```

The following command produces a maximally rich and brilliant (forceful) voice:

```
[ :np ] [ :dv ri 90 sm 0 ] Do I sound more forceful?
```

Smoothness and richness are usually negatively correlated when a speaker dynamically changes laryngeal output. The **sm** and **ri** options do not influence the speaker's identity very much.

Nopen Fixed, **nf**

The number of samples in the open part of the glottal cycle is determined not only by the **ri** option, but also by a second option, **nf**. The **Nopen Fixed (nf)** option is the number of fixed samples in the open portion of the glottal cycle.

Most speakers adjust the open phase to be a certain fraction of the period, and this fraction is determined by the **ri** option. Other speakers keep the open phase fixed in duration when the overall period varies. To simulate this behavior, set the **ri** option to 100 and adjust the **nf** option to the desired duration of the open phase. The shortest

possible open phase is 10 (1 ms), and the longest is three quarters of the period duration (about 70 for a male voice).

Laryngealization, *la*

Many speakers turn voicing on and off irregularly at the beginnings and ends of sentences, which gives a querulous tone to the voice. This departure from perfect periodicity is called laryngealization or creaky voice quality.

The Laryngealization (***la***) option controls the amount of laryngealization, in the voice. A value of 0 results in no laryngealized irregularity, and a value of 100 (the maximum) produces laryngealization at all times. For example, to make Betty moderately laryngealized, type the following command:

```
[ :nb ][ :dv la 20 ]
```

The ***la*** option creates a noticeable difference in the voice, although it is not altogether a pleasant change.

Changing Pitch and Intonation

Seven speaker-definition options control aspects of the fundamental frequency (**f0**) contour of the voice. These options are listed in Table 5-1.

Table 5-1 Fundamental Frequency Contour Options

bf	Baseline fall, in Hz
hr	Hat rise, in Hz
sr	Stress rise, in Hz
as	Assertiveness, in %
qu	Quickness, in %
ap	Average pitch, in Hz
pr	Pitch range, in %

Baseline Fall, bf

The Baseline fall (**bf**) option in Hz determines one aspect of the dynamic fundamental frequency contour for a sentence. If the **bf** option is 0, the reference baseline fundamental frequency of a sentence begin and ends at 115 Hz. All rule-governed dynamic swings in **f0** are computed with respect to the reference baseline.

Some speakers begin a sentence at a higher **f0** and gradually fall as the sentence progresses. This falling baseline behavior can be simulated by setting the **bf** option to the desired fall in Hz. For example, setting the **bf** option to 20 Hz causes the **f0** pattern for a sentence to begin at 125 Hz (115 Hz plus half of **bf**) and to fall at a rate of 16 Hz per second until it reaches 105 Hz (115 Hz minus half of **bf**). The baseline remains at this lower value until it is reset automatically before the beginning of the next full sentence (right after a period, question mark, or exclamation point). The rate of fall (16 Hz per second) is fixed, regardless of the extent of the fall.

Whenever you include a [+] syntactic symbol in the text to indicate the beginning of a paragraph, the baseline is automatically set to begin slightly higher for the first sentence of the paragraph. While baseline fall differs among speakers, it is not a good cue for differentiating among them. As long as the fall is not excessive, its presence or absence is hardly noticeable. See Chapter 4 for a complete list of symbols.

Hat Rise, hr

The Hat rise (**hr**) option (nominal hat rises in Hz) and **sr** option (nominal stress impulse rises in Hz) determine aspects of the dynamic fundamental frequency contour

for a sentence. To modify these values selectively, you should understand how the **f0** contour is computed as a function of lexical stress pattern and syntactic structure of the sentence.

A sentence is first analyzed and broken into clauses with punctuation and clause-introducing words to determine the locations of clause boundaries. Within each clause, the **f0** contour rises on the first stressed syllable, stays at a high level for the remainder of the clause up to the last stressed syllable, and falls dramatically on the last stressed syllable. This rise-at-the-beginning and fall-at-the-end pattern has been called the hat pattern by linguists, using the analogy of jumping from the brim of a hat to the top of the hat and back down again.

The **hr** option indicates the nominal height, in Hz of a pitch rise to a plateau on the first stress of a phrase. A corresponding pitch fall is placed by rule on the last stress of the phrase. Some speakers use relatively large hat rises and falls, while others use a local impulse-like rise and fall on each stressed syllable. The default hr option value for Paul is 18 Hz, indicating that the **f0** contour rises a nominal 18 Hz when going from the brim to the top of the hat. To simulate a speaker who does not use hat rises and falls, use the command:

```
[ :dv hr 0 ]
```

Other aspects of the hat pattern are important for natural intonation but are not accessible by speaker-definition commands. For example, the hat fall becomes a weaker fall followed by a slight continuation rise if the clause is to be succeeded by more clauses in the same sentence. Also, if unstressed syllables follow the last stressed syllable in a clause, part of the hat fall occurs on the very last (unstressed) syllable of the clause. If the clause is long, DECTalk Software may break it into two hat patterns by finding the boundary between the noun phrase and the verb phrase.

If DECTalk Software is in phoneme input mode and you use the pitch rise [/] and pitch fall [\] symbols, the **hr** option determines the actual rise and fall in Hz. See Chapter 4 for a complete list of symbols.

Stress Rise, sr

The Stress rise (**sr**) option indicates the nominal height, in Hz, of a local pitch rise and fall on each stressed syllable. This rise-fall is added to any hat rise or fall that is also present. For example, Paul has the **sr** option set to 32 Hz, resulting in an **f0** rise-fall gesture of 32 Hz over a span of about 150 ms, which is located on the first and succeeding stressed syllables. However, DECTalk Software rules reduce the actual height of successive stress rises and falls in each clause and cause the last stress pulse to occur early so that there is time for the hat fall during the vowel.

If the **sr** option is set too low, the speech sounds monotone within long phrases. Great changes to the **hr** and **sr** options from their default values for each speaker are not necessary or desirable, except in unusual circumstances.

Assertiveness, as

The Assertiveness (**as**) option, in %, indicates the degree to which the voice tends to end statements with a conclusive final fall. Assertive voices have a dramatic fall in pitch at the end of utterances. Neutral or meek speakers often end a sentence with a slight questioning rise in pitch to deflect any challenges to their assertions. A value of 100 is very assertive, while a value of 0 is extremely meek.

Quickness, qu

The Quickness (**qu**) option, in percentage, controls the speed of response to a request to change the pitch. All hat rises, hat falls, and stress rises can be thought of as suddenly applied commands to change the pitch, but the larynx is sluggish and responds only gradually to each command. A smaller larynx typically responds more quickly, so while Harry has a quickness value of 10, Kit has a value of 50.

In engineering terms, a value of 10 implies a time constant (time to get to 70% of a suddenly applied step target) of about 100 ms. A value of 90% corresponds to a time constant of about 50 ms. Lower quickness values may mean that the **f0** never reaches the target value before a new command comes along to change the target.

Average Pitch, ap, and Pitch Range, pr

The Average pitch (**ap**) option (average pitch, in Hz) and the pitch range (**pr**) option (pitch ranges in % of normal range) modify the computed values of fundamental frequency, **f0**, according to the formula:

$$f0' = ap + ((f0 - 120) * pr) / 100$$

If the **ap** option is set to 120 Hz and the **pr** option to 100%, there is no change to the normal **f0** contour that is computed for a typical male voice. The effect of a change in the **ap** option is simply to raise or lower the entire pitch contour independently by a constant number of Hz, whereas the effect of the **pr** option is to expand or contract the swings in pitch about 120 Hz.

Normally, a smaller larynx simultaneously produces **f0** values that are higher in average pitch and higher in pitch range by about the same factor (the whole **f0** contour is multiplied by a constant factor). Observing the values assigned to the **ap** and **pr** options for each of the voices, you can see that the voices rank in average pitch from low (Harry) to high (Kit).

Rankings for the **pr** option are similar, except that Frank has a flat, nonexpressive pitch range as compared with his average pitch.

The best way to determine a good pitch range for a new voice is by trial and error. You can create a monotone or robot-like voice by setting the pitch range to 0. For example, to make Harry speak in a monotone at exactly 90 Hz, type the following command.

```
[ :nh ][ :dv ap 90 pr 0 ] I am a robot.
```

Reducing the pitch range reduces the dynamics of the voice, producing emotions such as sadness in the speaker. Increasing the pitch range while leaving the average pitch the same or setting it slightly higher suggests excitement.

Due to constraints involved in pitch-synchronous updating of other dynamically changing options, the fundamental frequency contour that is computed by the preceding formula is then checked for values that are outside the following limits.

```
f0 maximum = 500 Hz
```

```
f0 minimum = 50 Hz
```

Any value outside this range is limited to fall within the range.

To keep you from exceeding reasonable limits on the options that control pitch, certain constraints apply to the values selected. If the **Design Voice** command specifies values outside these limits, the value is limited to the nearest listed value before execution.

Changing Relative Gains and Avoiding Overloads

Eight speaker-definition options control the output levels of various internal resonators. These options are listed in Table 5-1.

Table 5-1 Internal Resonator Options

gv	Gain of voicing source, in dB
gh	Gain of aspiration source, in dB
gf	Gain of frication source, in dB
gn	Gain of nasalization, in dB
g1	Gain of cascade formant resonator 1, in dB
g2	Gain of cascade formant resonator 2, in dB
g3	Gain of cascade formant resonator 3, in dB
g4	Gain of cascade formant resonator 4, in dB
g5	Loudness of the voice, in dB

Loudness, g5

The Loudness of the voice (**g5**) option is set to about the same perceived loudness for each of the predefined voices. The values chosen are optimal for telephone conversation and are near the maximum value beyond which some phonemes would probably cause an overload squawk. A near-maximum value was selected for each predefined voice to maximize the signal-to-noise level of DECTalk Software.

If you want to decrease the loudness of a voice or temporarily increase a phrase that is known not to overload, determine the **g5** option value in dB for the voice in question. Then adjust the voice by using the following command:

```
[ :np ] [ :dv g5 76 ] I am speaking at about half my normal level.
```

Because the **g5** option value for Paul is 86, this command reduces loudness by 10 dB. Perceived loudness approximately doubles (or halves) for each 10 dB increment (or decrement) in the **g5** option.

Software control over loudness is useful in a loudspeaker application where the background noise level in the room might change. For example, a vocally handicapped, wheelchair-bound person does not want to appear to be shouting in a quiet interpersonal conversation, but he or she may want to be able to converse in a noisy room as well.

Note

DECtalk Software comes with volume control so that modification of the **g5** option should not be necessary. Using the **Volume** command or the volume control knob on the external loudspeaker is recommended.

Sound Source Gains, **gv**, **gh**, **gf**, and **gn**

Several types of sound sources are activated during speech production: voicing, aspiration, frication, and nasalization. The relative output levels of these sounds, in dB, are determined by the Gain of voicing source (**gv**) option, the Gain of aspiration source (**gh**) option, the Gain of frication source (**gf**) option, and the Gain of nasalization (**gn**) option, respectively. The default settings for these options are factory preset to maximize the intelligibility of each voice. However, changing the settings can be useful in debugging the system or in demonstrating aspects of the acoustic theory of speech production. You can change the level of one sound source globally. For example, turn off frication to hear just the output of the larynx. You might need to reduce these options to overcome certain kinds of overloads, but try the procedure described in the next section first.

Cascade Vocal Tract Gains, **g1**, **g2**, **g3**, and **g4**

Changes in head size or other options can sometimes produce overloads in the synthesizer circuits. If this occurs, make sure that the **f4** and **f5** options are set to reasonable values. If the squawk remains, you can adjust several gain controls in the cascade of formant resonators of the synthesizer to attenuate the signal at critical points. These gain controls are the Gain of cascade formant resonator (**g1** through **g4**) options. These gains can then be amplified back to desired output levels later in the synthesis.

Use the following procedure to correct an overload (typically indicated by a squawk during part of a word):

1. Synthesize the word or phrase several times to make sure the squawk occurs consistently. Use the same test word each time a change to a gain is made.
2. Determine the default values for the **g1** through **g4** options for the speaker that overloads.
3. Reduce the **g1** option by increments of three until the squawk goes away. When the squawk goes away, note the reduction that was needed. If more than a 10 dB decrement is required, some other option has probably been changed too much. If

the squawk does not go away at all, then you may need to reduce the **gv** option instead of the **g1** option.

4. Increase the **g5** option to return the output to its original level. For example, if the **g1** option was reduced by 6 dB, add 6 dB to the **g5** option (or to the **g4** option if the **g5** option is already at a maximum). If incrementing the **g5** option causes the squawk to return, then decrease the **g5** option slowly until the squawk goes away.

This procedure works in most cases, but using the **g2** option rather than the **g1** option can work better. If you can return the **g1** option to its factory-preset value and reduce the **g2** option instead to make the squawk go away, then the signal-to-quantization-noise level in the **g1** option remains maximized. If you can eliminate the squawk by using the **g3** or **g4** option rather than the **g2** option, more of the cascaded resonator system can be made immune to quantization noise accumulation.

Saving Changes as Val's Voice

A user can change any of the voice characteristics of the current speaker by using the options available in the **Design Voice** command. These changes are active only while the current speaker remains current. You can save a modified speaker definition in a buffer while synthesizing speech with other voices. To save voice changes for use after the current speaker has changed, use the save option of the **Design Voice** command. These voice changes are saved as the voice of Val. The Val voice **[:nv]** is either male or female, depending on what values are stored in the buffer. If you call Val before storing any values in the buffer, DECtalk Software initializes Val voice to be the same as that of Paul.

Save, save

The Save (**save**) option of the **Design Voice** command lets you save speaker-definition options as Val's voice. You can modify any of the predefined voices, but you can save the modifications only as Val's voice. The following commands store a modified Betty voice in Val and then recall the modified voice:

```
[ :nb ][ :dv sx m save ] Betty now sounds like a man. Val now has this voice.  
[ :nb ] Betty's voice is back to normal.  
[ :nv ] Val's voice sounds like Betty as a man.
```

Val's voice characteristics are retained until the **TextToSpeechShutdown** call is issued or a new save is done. You must reenter new voice characteristics for Val after successfully issuing a startup function.

Note

If you want to use the save option, leave a space between the command option and the trailing bracket; for example, **[:dv save]**.

Summary of Design Voice Options

Of the 28 options, only a few cause dramatic changes in the voice. The greatest effects are obtained with changes to the **hs**, **ap**, **pr**, and **sx** options, while moderate changes occur when modifying the **la** and **br** options. To some extent, DECtalk Software's nine predefined speakers cover most of the possible voices. However, you might easily find ways to slightly improve one of the standard voices.

Chapter 6 Preprocessor Rules for Parsing

The preprocessor parses text to ensure that DECtalk Software pronounces it correctly and efficiently with respect to its context. Users can suppress the parsing action of the preprocessor with the **Skip** inline command or modify it with the **Punctuation** inline command. Three sets of rules apply to the parsing process: email rules, punctuation rules, and general rules.

Parsing email

When the preprocessor parses an email message, it strips out much of the mail header, saving only:

- Sent:
- Date:
- Subject:
- Subject: Re:
- From:
- To:
- Forwarded Message:

Parsing Punctuation

When the preprocessor encounters punctuation, it interprets each punctuation mark (by default) as a guide to speaking the text normally, unless you use inline commands to

specify otherwise with the **Punctuation** command, **[[:punct]]** or the **Skip** command, **[[:skip]]**.

Interpreting Punctuation Marks as Words

For the **[[:punct all]]** command, the preprocessor interprets each punctuation mark as a word to be pronounced. For example, the symbol “~” is interpreted as the word “tilde,” and the symbol “,” is interpreted as the word “comma.”

For the **[[:punct none]]**, **[[:punct pass]]**, and **[[:skip all]]** commands, the preprocessor interprets the following symbols normally to modify text:

- .
- ,
- ;
- :
- ?
- !

All other punctuation marks are ignored.

Interpreting Punctuation Marks as Punctuation

For the **[[:punct some]]** command, the preprocessor applies the following rules:

- Multiple instances of identical punctuation marks are reduced to a single symbol. For example, ----- becomes -, and ***** becomes *.
- Doubly encapsulated items become singly encapsulated. For example, “(intelligent)” and ((intelligent)) become (intelligent).
- Hours and minutes are not altered. For example, 2:43pm becomes **two forty-three P M**.
- Numerals and decimal numbers are not altered. For example, -3.52 becomes **minus three point five two**.
- Currency values are interpreted appropriately. For example, -\$43.65 becomes **minus forty-three dollars and sixty-five cents**, and +\$123.21 becomes **plus one hundred and twenty-three dollars and twenty-one cents**.

- Uppercase single letters followed by periods are interpreted as single letters. For example, **U.S.A.** becomes **U S A**.
- **P.M.** and **p.m.** become **P M**.
- Doubled clause boundary symbols are reduced to the first clause boundary. For example, **boom!,** becomes **boom!**
- Commas and hyphens not followed by spaces are changed to be followed by spaces. For example **look,look** becomes **look, look**.

General Rules

Rules for parsing numbers and some other items vary according to the language being spoken.

German

Language-specific rules apply to:

- Hours and minutes
- Dates
- Currency
- Phone numbers
- Compound nouns

Spanish (Castilian and Latin American)

Language-specific rules apply to:

- Dates
- Currency
- Phone numbers
- Credit cards

English (UK)

Language-specific rules apply to:

- Dates
- Addresses

English (US, UK)

Language-specific rules apply to:

- Dates
- Hours and minutes
- Street, avenue, and drive
- Numbered street names; for example, **29 42 Street** becomes **twenty-nine forty-second street**
- Phone numbers are spoken as digits, with appropriate pauses
- **Dr.** becomes **doctor**
- **St.** becomes **saint**
- Two-letter state names are pronounced in full; for example **MA 01749** becomes **Massachusetts zero one seven four nine**
- Postal zip codes within a mail address are spoken one digit at a time
- URL addresses are spoken one character at a time
- File names are spoken one character at a time
- In compound words, prefixes may be broken apart from the second word
- Days of the week
- Directions on the compass are spoken in full; for example **30 W** becomes **thirty west**

- Roman numerals following a name are spoken as ordinal numbers; for example **John Doe III** becomes **John Doe the third**
- Credit card numbers are spoken appropriately; for example, **6011 4134 3621 4172** becomes **six zero one one, four one three four, three six two one, four one seven two.**
- In a word written with mixed uppercase and lowercase letters, each uppercase letter begins a new word; for example, **TextToSpeech** becomes **text to speech**
- Combinations of numbers and letters are broken into numbers and individual letters; for example **two34five** becomes **T W O thirty-four F I V E**; **XF302QB** becomes **XF three hundred and two QB**

Glossary

allophone

A positional or free variant of a phoneme.

applet

A small application that normally performs a very specific function and can be used with other larger applications.

arpabet

A special phonetic alphabet used to write phonemes and syllables.

clause boundary

The natural boundary between two or more clauses in a sentence that helps the listener easily separate the sentence into its component parts. Commas, periods, exclamation points, question marks, semi-colons, and colons are symbols used to indicate clause boundaries.

clause mode

The normal mode in which DECTalk Software speaks text a phrase, clause, or sentence at a time. In clause mode, speaking starts when DECTalk Software is sent a clause terminator (period, comma, exclamation point, question mark, semi-colon, or colon) followed by a space.

clause terminator

A symbol used to begin and terminate a clause boundary. Symbols can be periods, commas, exclamation points, question marks, semi-colon, or colon. Each of these symbols must be followed by a space.

comma pause

The pause DECTalk Software takes in speaking that is equivalent to inserting a comma in a sentence. Comma pause can be increased and decreased with the Comma Pause command.

.DIC file

The loadable dictionary file created by the User Dictionary Build Tool from a .tab source file.

dynamic engine

A text-to-speech engine that accesses .lib files using dynamic link libraries (DLLs). DLLs are software modules in Microsoft Windows operating environments that contain executable code and data that can be called and used by Windows applications or other DLLs. Functions and data in a DLL are loaded and linked at run time when they are referenced by a Windows application or other DLLs. DLLs can be unloaded when the code is no longer needed.

emphatic stress

The emphasis placed on a syllable of a word to give it more meaning.

falling intonation

A decrease in voice pitch.

flush

Process by which the Text-To-Speech system discards data in the system.

heuristic

A method or rule used to decide among several courses of action. Often called a “rule of thumb.” In the case of DECTalk Software, pronunciation heuristics govern the manner in which DECTalk Software pronounces words.

homograph

A pair of words that have the same spelling but which are pronounced differently, depending on which syllable is accented. For example, the pronunciation of *permit* as a noun and the pronunciation of *permit* as a verb.

index marker (flag)

A marker placed in the text stream to synchronize an external event. An index marker is inserted with the **Index Mark** command.

intonation

The manner in which a voice imparts extra meaning to speech by adjusting sound duration and voice pitch. For example, the emphasis and meaning of the sentence, *Bill, put in the edits.* can be changed by putting stronger emphasis on the name, Bill. *Bill! Put in the edits!*

letter mode

The state in which DECTalk Software speaks each letter as it is queued. In word and letter mode, DECTalk Software does not need to wait for a clause terminator to begin speaking. This command interacts with the rate selection command so that you can set both rate selection and letter mode for optimal output.

log file

A file that can contain text, phonemes, or syllables. The phonemes and syllables are written using the arpabet phoneme alphabet.

log-file mode

Log-file mode indicates that the speech samples are to be written as text, phonemes, or syllables into a log file rather than sent to an audio device. The TextToSpeechOpenLogFile() function enters the text-to-speech system into a log-file mode. The TextToSpeechCloseLogFile() function returns the text-to-speech system to the startup state.

morpheme

The minimum syntactic unit of a language that has an important role in determining pronunciations. For example, *spell* has only one morpheme, while *misspelling* is made up of three: *mis*, *spell*, and *ing*.

period pause

The pause DECTalk Software inserts when it finds a period that marks the end of the sentence. This pause imitates humans taking a breath. This pause is approximately half a second.

phoneme

The smallest unit of speech that distinguishes one word from another. Phonemes are divided into vowel and consonant phonemes. DECtalk Software interprets text within brackets as phonemes only after the phoneme arpabet command is used.

phoneme arpabet command

A command that causes all text within brackets to be treated as phonemic text.

phoneme string

Two or more phonemes together used to pronounce a special word or group of words.

phonemicize

To encode words as strings of phonemes.

phonemic mode

A mode DECtalk Software uses for speaking phoneme strings.

phonemic transcription

A word written the way it is pronounced is said to be in phonemic transcription or simply in phonemics. When DECtalk Software says a word or phrase not as you intended, you might need to use phonemic transcription to get the desired pronunciation. For example, *[r 'ehd]* is the phonemic transcription of the word *read*.

phrase boundary

A clause boundary formed by terminating punctuation (comma, period, exclamation point, question mark, semi-colon, colon) followed by a space.

pitch control symbols

Symbols used to override built-in DECtalk Software pitch control. Symbols include pitch rise [/], pitch fall [\], and pitch rise and fall [/ \].

primary stress

Most content words of English (nouns, verbs, adjectives, and adverbs) contain one primary stressed syllable. The primary stress symbol in DECtalk Software is the apostrophe ['].

proper name

First names, last names, street names, company names, and place names are all examples of proper names.

secondary stress

A symbol used to indicate a degree of stress that is between primary and unstressed (no stress). The secondary stress symbol is the grave accent [˘].

silence phonemes

Silences of specified durations inserted into text files in the same manner as you would insert a phoneme.

speech-to-memory mode

In speech-to-memory mode DECtalk Software writes speech samples into memory buffers instead of sending them to an audio device. The TextToSpeechAddBuffer() function supplies the text-to-speech system with the memory buffers that it needs. The TextToSpeechOpenInMemory() function enters the text-to-speech system into a speech-to-memory mode. The TextToSpeechCloseInMemory() function returns the text-to-speech system to the startup state.

startup function

Startup function refers to either the TextToSpeechStartup() functions or the TextToSpeechStartupEx().

startup state

Startup state indicates that the TextToSpeechStartup() function or TextToSpeechStartupEx() function has been successfully called and the text-to-speech system is *not* in one of the three special modes; wave-file, log-file, or speech-to-memory mode. While DECtalk Software is in the startup state, speech samples are sent to an audio device or ignored, depending on whether the DO_NOT_USE_AUDIO_DEVICE flag is set in the dwDeviceOptions parameter of the startup function. If the text-to-speech system is in one of its special modes, the speech samples are handled accordingly.

static engine

A text-to-speech engine that accesses .lib files without using dynamic link libraries (DLLs). See also **dynamic engine**.

syntactic function words

A set of words that are either unstressed or have secondary stress. They include prepositions, conjunctions, determiners, auxiliary verbs, pronouns, the question mark,

and clause introducers. DECtalk Software uses stress and syntactic symbols to control aspects of rhythm, stress, and intonation patterns. These symbols include punctuation marks such as commas, periods, question marks, and exclamation points.

.TAB file

The source file used to build a user dictionary.

user dictionary

The dictionary that you create for DECtalk Software to load and use with an application to control the pronunciation of specific words processed by the application.

user dictionary builder

A program applet included with DECtalk Software to build and compile user dictionaries.

voice-control command

A DECtalk Software command inserted into text strings and used to control basic and special Text-To-Speech attributes, such as speaking voice and speaking rate.

WAVE file

A Microsoft standard file format for storing waveform audio data. WAVE files have a .WAV file extension.

wave form output

The digitized reproduction of a sound wave form. DECtalk Software produces waveform output from the Speak program applet and the API, both of which allow you to save an ASCII text file to .WAV file format.

wave-file mode

Wave-file mode indicates that the speech samples are to be written to a wave file rather than sent to an audio device. The `TextToSpeechOpenWaveOutFile()` function enters the text-to-speech system into a wave-file mode. The `TextToSpeechCloseWaveOutFile()` function returns the text-to-speech system to the startup state.

word boundary

A white space character (space, tab, or carriage return) in the text that indicates a boundary between words. DECtalk Software uses word boundary symbols to select the word-beginning or word-ending allophone of a phoneme.

word mode

A text-processing mode where DECtalk Software speaks one word at a time. A blank space or equivalent after a character or string of characters causes that string to be spoken in word mode.

- [+] syntactic symbol, 5-11
- abbreviations, 2-1
- Access32, 1-48
- age, 5-2
- aged female voice, 2-15
- aged male voice, 2-15
- alternate pronunciations, 2-21
- ap option, 5-13
- API calls
 - TextToSpeechAddBuffer, 1-3
 - TextToSpeechCloseInMemory, 1-5
 - TextToSpeechCloseLang, 1-6
 - TextToSpeechCloseWaveOutFile, 1-8
 - TextToSpeechEnumLangs, 1-9
 - TextToSpeechGetCaps, 1-10
 - TextToSpeechGetFeatures, 1-11
 - TextToSpeechGetLanguage, 1-12
 - TextToSpeechGetRate, 1-13
 - TextToSpeechGetSpeaker, 1-14
 - TextToSpeechGetStatus, 1-15
 - TextToSpeechLoadUserDictionary, 1-16
 - TextToSpeechOpenInMemory, 1-17
 - TextToSpeechOpenLogFile, 1-19
 - TextToSpeechOpenWaveOutFile, 1-21
 - TextToSpeechPause, 1-23
 - TextToSpeechReset, 1-25
 - TextToSpeechResume, 1-27
 - TextToSpeechReturnBuffer, 1-28
 - TextToSpeechSelectLang, 1-29
 - TextToSpeechSetLanguage, 1-30
 - TextToSpeechSetRate, 1-31
 - TextToSpeechSetSpeaker, 1-32
 - TextToSpeechShutdown, 1-33
 - TextToSpeechSpeak, 1-34
 - TextToSpeechStartLang, 1-36
 - TextToSpeechStartup (UNIX only), 1-40
 - TextToSpeechStartup (Windows only), 1-37
 - TextToSpeechStartupEx, 1-43
 - TextToSpeechSync, 1-47
 - TextToSpeechTyping, 1-48
 - TextToSpeechUnloadUserDictionary, 1-49
 - TextToSpeechVersion, 1-50
 - TextToSpeechVersionEx, 1-51
- API function calls
 - TextToSpeechCloseLogFile, 1-7
- applet
 - userdict, 1-16
 - windict, 1-16
- Application development
 - electronic mail, 3-2
- arpabet alphabet, 1-19
- as option, 5-13
- aspiration, 5-16
- Assertiveness (as option), 5-13
- audio output, 1-23, 1-25
- audio system gain, 2-28
- Average pitch (ap option), 5-13
- background noise level, 5-15
- baseline, 5-11
- Baseline fall (bf option), 5-11
- bf option, 5-11
- bitmask, 1-11
- br option, 5-8
- Breathiness (br option), 5-8
- breathy, whispering speaker, 5-8
- brilliance, 5-9
- brilliant, rich voices, 5-9
- callback routine, 1-43
- Calls. *See* API calls
- characters, 2-11
- child's voice, 5-6
- child's voice, 2-15
- comma pause, 2-4
- Comma Pause [:comma] or [:cp] command, 2-4

- Comma Pause duration
 - control of, 3-9
- command names, 2-1
- commands
 - Comma Pause [:comma] or [:cp], 2-4
 - Design Voice, :dv, 2-5, 5-2
 - Dial Tones [:dial], 2-6
 - Error [:error], 2-7
 - Index mark [:index mark], 2-8
 - Log [:log], 2-9
 - Mode [:mode], 2-10
 - Name [:name], 2-15
 - Period Pause [:period] or [:pp], 2-16
 - Phoneme Interpretation [:phoneme], 2-17
 - Pitch [
 - pitch], 2-19
 - Play Wave Files [:play], 2-20
 - Pronounce [:pronounce], 2-21
 - Punctuation [:punct], 2-22
 - Rate Selection [:rate], 2-23
 - Say [:say], 2-24
 - Skip [:skip], 2-25
 - Sync [:sync], 2-26
 - Tone [:tone], 2-27
 - Volume [:volume], 2-28
- common errors, 3-12
- compatibility, 1-50
- contour, 5-12
- contour, limits, 5-14
- correct an overload, 5-16
- current buffer, 1-28
- DAPI compatibility, 1-50
- DECTalk
 - developing an application, 3-2
- DECTalk calls. *See* API calls
- DECTalk Multi-Language (ML) engine, 1-6, 1-36
- DECTalk Software API calls, 1-1
- DECTalk Software voices
 - aged female voice [:nu], 2-15
 - aged male voice [:nf], 2-15
 - child's voice [:nk], 2-15
 - default male voice [:np], 2-15
 - female voice [:nr], 2-15
 - full female voice [:nb], 2-15
 - full male voice [:nh], 2-15
 - male voice [:nd], 2-15
 - Val's voice [:nv], 2-15
 - whispering female voice [:nw], 2-15
- DECTalk Software voices, 2-15, 5-2
- deep voice, 5-6
- default male voice, 2-15
- default rate for DECTalk Software, 3-7
- default speaking rate, 2-23, 3-7
- delimiters, 2-17
- Design Voice [:dv] command, 2-5, 2-15, 5-2. *See also* speaker-definition options
- Dial Tones [:dial] command, 2-6
- dialect, 5-2
- dictionary
 - main, 1-37, 1-40
 - user pronunciation, 1-37, 1-40
- DO_NOT_USE_AUDIO_DEVICE flag, 1-34
- dramatic voice changes, 5-6
- duration and pitch attributes, 4-2
- dwBufferLength element, 1-28
- dwDeviceOptions parameter, 1-5, 1-7, 1-8, 1-34
- dynamic fundamental frequency contour, 5-12
- email
 - headers, 2-12
- email
 - parser, 3-2
 - parsing, 6-1
- email
 - text, 2-12
- emotional state, 5-2
- English UK,
 - parsing, 6-4
- English US, UK,
 - parsing, 6-4
- Error [:error] command, 2-7
- error mode, 2-7
- errors, 3-12
- excitement, 5-14

fastest usable rate, 3-7
 features of DECTalk, 1-11
 female voice, 2-15, 5-5, 5-6, 5-8
 formant, 5-6
 formant filter, 5-7
 formant resonances, 5-6
 frequencies, 4-11
 frequency contour, 5-11, 5-12
 frequency contour, limits, 5-14
 frication, 5-16
 full female voice, 2-15
 full male voice, 2-15
 Function calls. *See* API calls
 fundamental frequency, 5-11, 5-13
 fundamental frequency contour, 5-12
 fundamental frequency contour, limits, 5-14
 g1 through g4 options, 5-16
 g5 option, 5-15
 Gain of aspiration source (gh option), 5-16
 Gain of cascade formant resonator (g1 through g4 options), 5-16
 Gain of frication (gf option), 5-16
 Gain of nasalization (gn option), 5-16
 Gain of voicing source (gv option), 5-16
 German
 parsing, 6-3
 gf option, 5-16
 gh option, 5-16
 gn option, 5-16
 gv option, 5-16
 Hat rise (hr option), 5-12
 Head size (hs option), 5-6
 head size and shape, 5-2, 5-5
 headers, email, 2-12
 higher voice, 5-6
 homograph phonetics, 4-14
 for the letter A, 4-14
 for the letter R, 4-19
 for the letters B and C, 4-15
 for the letters D through G, 4-16
 for the letters I through L, 4-17
 for the letters M through P, 4-18
 for the letters S through W, 4-20
 homographs, 4-1, 4-13
 hr option, 5-12
 hs option, 5-6
 Index mark [:index mark] command, 2-8
 Index Mark command, 3-6
 index marks, 3-6
 information, version, 1-51
 in-line commands, 2-1. *See* commands
 interpretation, 2-1, 2-17
 intonation and stress, 2-1
intonation patterns, 3-2
 la option, 5-10
 language, 1-6
 Laryngealization (la option), 5-10
 larynx size and behavior, 5-2
 Lax breathiness (lx option), 5-8
 lexical stress pattern, 5-12
 Log [:log] command, 2-9
 log file, 1-19, 2-9
 log-file mode, 1-19
 Loudness of the voice (**g5** option), 5-15
 loudspeaker application, 5-15
 low-pass filter, 5-9
 lx option, 5-8
 main dictionary, 1-37, 1-40
 male voice, 2-15, 5-5, 5-6, 5-8, 5-13
 markers, 2-8
 math equations, 3-7
 ML engine, 1-6, 1-36
 mode
 error, 2-7
 log file, 1-19
 wave-file, 1-21
 Mode [:mode] command, 2-10
 Mode command options
 Email, 2-12
 Europe, 2-10
 Math, 2-11
 Name, 2-11
 monaural volume, 2-28
 monotone voice, 5-14

- musical sounds, 4-11
- musical tones, 4-11
- Name [:name] command, 2-15
- names, 2-1
- nasalization, 5-16
- nb voice, 2-15
- ndvoice, 2-15
- nf option, 5-9
- nf voice, 2-15
- nh voice, 2-15
- nk voice, 2-15
- noise level, 5-15
- nonbreathy, tense voice, 5-8
- Nopen Fixed (nf option), 5-9
- notes, 4-11
- np voice, 2-15
- nr voice, 2-15
- nu voice, 2-15
- nv voice, 2-15
- nw voice, 2-15
- option names, 2-1
- overload, 5-16
- overload squawk, 5-15
- parsing, 6-1
 - email, 6-1
 - English UK, 6-4
 - English US, UK, 6-4
 - punctuation, 6-1
 - rules, 6-3
 - Spanish, 6-3
- parsing,
 - German, 6-3
- pause, 2-4
- paused state, 1-21
- Period Pause [:period] command, 2-16
- Period Pause duration
 - control of, 3-9
- phone numbers, 3-7
- phoneme delimiters, 2-17
- phoneme interpretation, 2-1, 2-17
- Phoneme Interpretation [:phoneme] command, 2-17
- phonemes
 - listed in Unicode sequence, 4-1
- phonemic symbols, 4-1, 4-2
- phonemicizing text, 2-17
- phrasing requirements*, 3-2
- pitch
 - range, 5-2
- Pitch [
 - pitch] command, 2-19
- pitch and timing habits, 5-2
- pitch attributes, 4-2
- pitch numbers, 4-11
- pitch range, 5-14
- Play Wave Files [:play] command, 2-20
- pleasant, intelligible voices, 5-5
- position markers, 2-8
- pitch
 - range (pr option), 5-13
- ppTTSbuffer, 1-28
- pr option, 5-13
- preprocessor, 6-1
- preprocessor rules for parsing, 6-1
- primary pronunciations, 2-21
- prominent resonant peaks, 5-6
- Pronounce [:pronounce] command, 2-21
- pronunciation dictionary, 1-16
- pronunciations
 - alternate, 2-21
 - primary, 2-21
- punctuation
 - parsing, 6-1
- Punctuation [:punct] command, 2-22
- punctuation modes, 2-22
- qu option, 5-13
- queued text, 1-47
- Quickness (qu option), 5-13
- Rate command, 3-7
- Rate Selection [:rate] command, 2-23
- reference baseline, 5-11
- resonant peaks, 5-6
- rhythm patterns*, 3-2
- ri option, 5-9

- rich voices, 5-9
- Richness (ri option), 5-9
- right bracket (]), 2-1, 2-18
- robot-like voice, 5-14
- rules
 - parsing, 6-1
 - parsing, 6-3
- sadness, 5-14
- SAPI functions, 4-1
- Save (save) option, 5-18
- save option, 5-18
- save voice changes, 5-18
- Say [:say] command, 2-24
- sex, 5-2
- Sex (sx option), 5-5
- Silence phonemes, 3-3
- sing songs, 4-11
- singing tones, 4-1
- Skip [:skip] command, 2-25
- sm option, 5-9
- smooth, soft voices, 5-9
- Smoothness (sm option), 5-9
- smoothness and brilliance, 5-9
- smoothness and richness, 5-9
- soft voices, 5-9
- software voices. *See* DECTalk Software voices
- sounds, musical, 4-11
- Spanish,
 - parsing, 6-3
- speaker-definition options, 2-5, 5-2, 5-5, 5-8, 5-11, 5-15
 - Assertiveness (as option), 5-13
 - Average pitch (ap option), 5-13
 - Baseline fall (bf option), 5-11
 - Breathiness (br option), 5-8
 - formants, 5-6
 - Gain of aspiration source (gh option), 5-16
 - Gain of cascade formant resonators (g1 through g4 options), 5-16
 - Gain of friction source (gf option), 5-16
 - Gain of nasalization (gn option), 5-16
 - Gain of voicing source (gv option), 5-16
 - Hat rise (hr option), 5-12
 - Head size (hs option), 5-6
 - Laryngealization (la option), 5-10
 - Lax breathiness (lx option), 5-8
 - Loudness of the voice (g5 option), 5-15
 - Nopen Fixed (nf option), 5-9
 - pitch range (pr option), 5-13
 - Quickness (qu option), 5-13
 - Richness (ri option), 5-9
 - Save (save option), 5-18
 - Sex (sx option), 5-5
 - Smoothness (sm option), 5-9
 - Stress rise (sr option), 5-12
- speaking math equations, 3-7
- speaking rate, 1-13, 1-31, 2-1, 2-23, 3-7
- speaking voice, 2-1
- special characters, 2-11
- special modes, 1-34
- special phrasing requirements*, 3-2
- special symbols, 2-11
- spectral change, 5-9
- speech production, 5-16
- speech samples, 1-3
- speech-to-memory mode, 1-3, 1-17
- spoken language and written text, 3-10
- squawk, 5-15, 5-16
- sr option, 5-12
- startup function, 1-5, 1-7, 1-8, 1-34, 2-8
- startup state, 1-5
- status, 1-15
- stereo volume, 2-28
- store speech samples, 1-3
- store voice changes, 5-2
- stress and syntactic symbols*, 3-2, 4-1
- stress pattern, 5-12
- stress patterns*, 3-2
- Stress rise (sr option), 5-12
- stress symbols, 4-2, 4-6
- supported SAPI functions, 4-1
- sx option, 5-5
- symbols, 2-11, 4-2, 4-6
- Sync [:sync] command, 2-26

- syntactic structure, 5-12
- syntactic symbols*, 3-2, 4-1, 4-6
- synthesizer, 2-1
- system parameters, 1-15
- system resources, 1-33
- telephone conversation, 5-15
- tense voice, 5-8
- text tuning, 3-10
- TextToSpeechAddBuffer, 1-3
- TextToSpeechCloseInMemory, 1-5
- TextToSpeechCloseLang, 1-6
- TextToSpeechCloseLogFile, 1-7
- TextToSpeechCloseWaveOutFile, 1-8
- TextToSpeechEnumLangs, 1-9
- TextToSpeechGetCaps, 1-10
- TextToSpeechGetFeatures, 1-11
- TextToSpeechGetLanguage, 1-12
- TextToSpeechGetRate, 1-13
- TextToSpeechGetSpeaker, 1-14
- TextToSpeechGetStatus, 1-15
- TextToSpeechLoadUserDictionary, 1-16
- TextToSpeechOpenInMemory, 1-17
- TextToSpeechOpenLogFile, 1-19
- TextToSpeechOpenWaveOutFile, 1-21
- TextToSpeechPause, 1-23
- TextToSpeechReset, 1-25
- TextToSpeechResume, 1-27
- TextToSpeechReturnBuffer, 1-28
- TextToSpeechSelectLang, 1-29
- TextToSpeechSetLanguage, 1-30
- TextToSpeechSetRate, 1-31
- TextToSpeechSetSpeaker, 1-32
- TextToSpeechShutdown, 1-33
- TextToSpeechSpeak, 1-34
- TextToSpeechStartLang, 1-36
- TextToSpeechStartup (UNIX only), 1-40
- TextToSpeechStartup (Windows only), 1-37
- TextToSpeechStartupEx, 1-43
- TextToSpeechSync, 1-47
- TextToSpeechTyping, 1-48
- TextToSpeechUnloadUserDictionary, 1-49
- TextToSpeechVersion, 1-50
- TextToSpeechVersionEx, 1-51
- timing habits, 5-2
- Tone [:tone] command, 2-27
- tones, 2-6, 2-27, 4-11
- TTS_BUFFER_T structure, 1-28
- TTS_CAPS_T structure, 1-10
- TTS_MSG_BUFFER, 3-6
- TTS_MSG_INDEX_MARK, 3-6
- typical male voice, 5-13
- user dictionary, 1-49
- user pronunciation dictionary, 1-37, 1-40
- user-defined pronunciation dictionary, 1-16
- userdict applet, 1-16
- Val, 5-2, 5-18
- Val's voice, 2-15
- valid licenses, 1-43
- version information, 1-51
- vocal fold vibrations, 5-9
- voice changes, 5-2, 5-18
- voice contour, 5-11
- voice quality, 5-8
- voices, 5-5. *See* DECtalk Software voices
 - breathy, 5-8
 - brilliant, rich, 5-9
 - child, 5-6
 - female, 5-5, 5-6
 - male, 5-5, 5-6
 - monotone, 5-14
 - nonbreathy, 5-8
 - DECtalk, 5-2
 - pleasant, intelligible, 5-5
 - robot-like, 5-14
 - smooth, soft, 5-9
 - tense, 5-8
 - whispering, 5-8
- voice-selection, 3-12
- voicing, 5-16
- Volume command, 2-28
- Volume command, 5-16
 - option
 - Down, 2-28
 - Set, 2-28

Up, 2-28
volume control, 2-28
volume control knob, 5-16
volume settings, 2-28
wave file, 1-8, 1-21, 2-20
wave output device, 1-43

WAVE_MAPPER, 1-43
wave-file mode, 1-21
whispering female voice, 2-15
whispering speaker, 5-8
windict applet, 1-16