



# DECtalk<sup>®</sup> Software

Reference Guide 5.0



# DECtalk<sup>®</sup> Software Reference Guide

## October 2003

This guide provides reference descriptions of the DECtalk<sup>®</sup> Software API functions and in-line commands, followed by a guide to using the in-line commands. It also provides reference tables for phonemic symbols, stress and syntactic symbols, tones, and homographs; a guide to customizing DECtalk voices; and a guide to the DECtalk preprocessor's rules for parsing text.

## Revision / Update Information:

This document supersedes the *DECtalk Software Reference Guide, Version 4.6*.

## Operating System:

Microsoft Windows 95/98/ME/NT/2000/XP  
 Microsoft Windows CE/Pocket PC  
 Red Hat Linux Version 5.0 or higher

## Software Version:

DECtalk Software Version 5.00

## October 2003

The information in this publication is subject to change without notice. Fonix Corporation reserves the right to make changes without notice to this, or any of its products, to improve reliability, performance, or design.

FONIX CORPORATION SHALL NOT BE LIABLE FOR TECHNICAL OR EDITORIAL ERRORS OR OMISSIONS CONTAINED HERIN, NOR FOR INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES RESULTING FROM THE FURNISHING, PERFORMANCE, OR USE OF THIS MATERIAL. THIS INFORMATION IS PROVIDED "AS IS" AND FONIX CORPORATION EXPRESSLY DISCLAIM ANY AND ALL WARRANTIES, EXPRESS, IMPLIED, STATUTORY, OR OTHERWISE, INCLUDING WITHOUT LIMITATION, ANY EXPRESS, STATUTORY, OR IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT.

This publication contains information protected by copyright. This publication shall not be reproduced, transmitted, or stored in a retrieval system, nor its contents used for any purpose, without the prior written consent of Fonix Corporation. Fonix Corporation

assumes no responsibility for the use of any circuitry other than the circuitry that is part of a product of Fonix Corporation. Fonix Corporation does not convey to the purchaser of the product described herein any license under the patent rights of Fonix Corporation nor the rights of others.

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.

Copyright © 2000, 2001, 2002, 2003 by Fonix Corporation. Certain portions © 1997, 1998, 1999 Compaq Computer Corporation. All rights reserved.

The Fonix logo and DECtalk are trademarks of Fonix Corporation.

Compaq is a registered trademark of Compaq Computer Corporation.

Tru64 is a trademark of Compaq Information Technologies Group, L.P.

Intel is a trademark of Intel Corporation.

Linux is a registered trademark of Linus Torvalds.

Microsoft, Windows, Windows 95, Windows 98, Windows ME, Windows NT, Windows 2000, Windows XP and Windows CE are registered trademarks of Microsoft Corporation.

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

Red Hat is a registered trademark of Red Hat Software, Inc.

SoundBlaster is a registered trademark of Creative Labs, Inc.

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

# Contents

DECtalk® Software Reference Guide 1

Preface 5

Purpose and Audience 5

Structure 5

Conventions 5

DECtalk Software API Functions 7

DECtalk Software In-Line Commands 61

Overview 63

Comma Pause [:comma] 65

Design Voice [:dv] 65

Dial Tones [:dial] 66

Error [:error] 66

Index Mark [:index mark] 67

Log [:log] 68

Mode [:mode] 69

Name [:name] 72

Period Pause [:period] 74

Phoneme Interpretation [:phoneme] 74

Pitch [:pitch] 76

Play Wave Files [:play] 77

Pronounce [:pronounce] 77

Punctuation [:punct] 78

Rate Selection [:rate] 79

Say [:say] 80

Skip [:skip] 81

Sync [:sync] 82

Tone [:tone] 83

Volume [:volume] 83

## Using In-Line Commands 87

Changing Rhythm, Stress, and Intonation 89

Developing an Electronic  
Mail-Reading Application 90

Optimizing the Quality of Spoken Text 91

Index Marks for Speech Status 92

Speaking Rate 93

Adjusting Period and  
Comma Pause Durations 94

Text-Tuning Example 95  
    Original Version 95  
    Revised Version 96

Avoiding Common Errors 97

## DECTalk Software Reference Tables 99

Phonemic Symbols Listed By Language 101  
    U.S. English 103  
    U.K. English 106  
    Castilian Spanish 108  
    Latin American Spanish 110  
    German 112  
    French 114

Stress and Syntactic Symbols 115

Phonemes Listed in Unicode Sequence 116

Pitch and Duration of Tones 119

Homographs 121

Supported SAPI Functions  
(Windows 95/98/ME/NT/2000/XP Only) 129

Supported SAPI Version 5 Features (Windows 98/ME/NT/2000/XP Only) 131

## Customizing a DECtalk Software Voice 133

Design Voice [:dv] 135

Definitions of DECtalk Software Voices 137

Changing Gender and Head Size 138

Sex, sx 139

Head Size, hs 139

Higher Formants, f4, f5, b4, and b5 140

Changing Voice Quality 141

Breathiness, br 142

Lax Breathiness, lx 142

Smoothness, sm 142

Richness, ri 143

Nopen Fixed, nf 143

Laryngealization, la 144

Changing Pitch and Intonation 144

Baseline Fall, bf 145

Hat Rise, hr 145

Stress Rise, sr 146

Assertiveness, as 147

Quickness, qu 147

Average Pitch, ap, and Pitch Range, pr 148

Changing Relative Gains

and Avoiding Overloads 149

Loudness, g5 149

Sound Source Gains, gv, gh, gf, and gn 150

Cascade Vocal Tract Gains, g1, g2, g3, and g4 151

Saving Changes as Val's Voice 152

Save, save 152

Summary of Design Voice Options 153

## Preprocessor Rules for Parsing 155

Email Parsing Rules 157

Punctuation Parsing Rules 157

Interpreting Punctuation Marks as Words 158

Interpreting Punctuation Marks as Punctuation 158

General Parsing Rules 159

German 159

Spanish (Castilian and Latin American) 159

English (UK) 160

English (US, UK) 160

Glossary 163

Index 171



# Preface

## 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) functions, in-line commands, and reference tables. The information in this guide is accurate for Windows 95/98/ME/NT/2000/XP, Windows CE/Pocket PC, and Linux 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 applets for development of a DECtalk Software application.

The guide is organized as follows:

- Chapter 1** DECtalk Software API Functions
- 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

## Conventions

The following conventions are used in this guide:

Convention	Meaning
enter	Enter means type the required information and press the Enter key.
mouse	Mouse refers to any pointing device, such as a mouse, a puck, or a stylus.
MB1	MB1 indicates the left mouse button.

click	Click means to press and release MB1.
Double click	Double click means to press and release MB1 twice in rapid succession without moving the mouse.
drag	The phrase drag 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, File > Exit 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	<b>Boldface</b> type in interactive examples indicates information you enter from the keyboard. For example: A:> <b>SETUP</b>
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].  <b>NOTE:</b> 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].  <b>NOTE:</b> 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 functions

Italics                      Italic text emphasizes important information.

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

# DECtalk Software API Functions



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

**Table 1-1: DECTalk Software API Functions**

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()
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

## Function: TextToSpeechAddBuffer()

<b>Description</b>	The <b>TextToSpeechAddBuffer()</b> function 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.										
<b>Syntax</b>	MMRESULT <b>TextToSpeechAddBuffer</b> (LPTTS_HANDLE_T phTTS, LPTTS_BUFFER_T pTTSTbuffer)										
<b>Parameters</b>	<p><b>LPTTS_HANDLE_T phTTS</b> - Specifies an opened text-to-speech handle.</p> <p><b>LPTTS_BUFFER_T pTTSTbuffer</b> - Points to a structure containing the memory buffers.</p> <p>Buffers are supplied by the application to be used while in speech-to-memory mode.</p>										
<b>Return Value</b>	<p>This function returns a value of type MMRESULT. The return value is zero if the call is successful.</p> <p>The return value is one of the following constants:</p> <table> <tr> <th>Constant</th><th>Description</th></tr> <tr> <td>MMSYSERR_NOERROR</td><td>Normal successful completion (zero).</td></tr> <tr> <td>MMSYSERR_INVALIDPARAM</td><td>Invalid parameter.</td></tr> <tr> <td>MMSYSERR_ERROR</td><td>Output to memory not enabled or unable to create a system object.</td></tr> <tr> <td>MMSYSERR_INVALIDHANDLE</td><td>The text-to-speech handle was invalid.</td></tr> </table>	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.
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.										
<b>Comments</b>	<p>The application must call <b>TextToSpeechOpenInMemory()</b> before calling <b>TextToSpeechAddBuffer()</b>. 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 Speech-To-Memory Mode in the <i>DEctalk Software Programmer's Guide</i> for more information on the TTS_BUFFER_T structure and its elements.</p> <p>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 <b>TextToSpeechS-peak()</b> call. Refer to Callback Routines and Window Procedures in the <i>DEctalk Software Programmer's Guide</i> for detailed information on passing information back to the calling application.</p>										

**See Also**

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

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

"TextToSpeechOpenInMemory()" on page 25

"TextToSpeechReturnBuffer()" on page 36

"TextToSpeechStartup()" [Windows] on page 45

"TextToSpeechStartup()" [Linux] on page 48

"TextToSpeechStartupEx()" on page 51



## Function: TextToSpeechCloseInMemory()

<b>Description</b>	The <b>TextToSpeechCloseInMemory()</b> function 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 <b>dwDeviceOptions</b> parameter in the startup function.								
<b>Syntax</b>	MMRESULT <b>TextToSpeechCloseInMemory</b> (LPTTS_HANDLE_T phTTS)								
<b>Parameters</b>	<b>LPTTS_HANDLE_T phTTS</b> - Specifies an opened text-to-speech handle.								
<b>Return Value</b>	This function 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: <table> <tr> <th>Constant</th><th>Description</th></tr> <tr> <td>MMSYSERR_NOERROR</td><td>Normal successful completion (zero).</td></tr> <tr> <td>MMSYSERR_ERROR</td><td>Output to memory not enabled or unable to create a system object.</td></tr> <tr> <td>MMSYSERR_INVALIDHANDLE</td><td>The text-to-speech handle was invalid.</td></tr> </table>	Constant	Description	MMSYSERR_NOERROR	Normal successful completion (zero).	MMSYSERR_ERROR	Output to memory not enabled or unable to create a system object.	MMSYSERR_INVALIDHANDLE	The text-to-speech handle was invalid.
Constant	Description								
MMSYSERR_NOERROR	Normal successful completion (zero).								
MMSYSERR_ERROR	Output to memory not enabled or unable to create a system object.								
MMSYSERR_INVALIDHANDLE	The text-to-speech handle was invalid.								
<b>Comments</b>	<b>TextToSpeechOpenInMemory()</b> must be called before calling <b>TextToSpeechCloseInMemory()</b> .								
<b>See Also</b>	"TextToSpeechOpenInMemory()" on page 25								

## Function: TextToSpeechCloseLang()

<b>Description</b>	The <b>TextToSpeechCloseLang()</b> function closes an instance for an installed language and attempts to unload it from the DECTalk Multi-Language (ML) engine.
<b>Syntax</b>	BOOL <b>TextToSpeechCloseLang</b> (char *lang)
<b>Parameters</b>	<b>char *lang</b> - Specifies the language being unloaded; passed as a null-terminated string containing the 2-character language ID.
<b>Return Value</b>	BOOL - Returns TRUE when a language is successfully unloaded, or FALSE when the operation cannot be completed or more instances have the thread started.
<b>Comments</b>	<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 value of FALSE may indicate invalid passing of the lang variable or more instances of the language still loaded. If there are more instances, the function frees the current instance and returns FALSE. After calling <b>TextToSpeechCloseLang()</b>, assume that the language handle is no longer valid.</p>
<b>Example</b>	<pre> 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 ; } </pre>
<b>See Also</b>	<p>"TextToSpeechEnumLangs()" on page 17</p> <p>"TextToSpeechSelectLang()" on page 37</p> <p>"TextToSpeechStartLang()" on page 44</p>

## Function: TextToSpeechCloseLogFile()

<b>Description</b>	The <b>TextToSpeechCloseLogFile()</b> function closes a log file opened by the <b>TextToSpeechOpenLogFile()</b> function 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.									
<b>Syntax</b>	MMRESULT <b>TextToSpeechCloseLogFile</b> (LPTTS_HANDLE_T phTTS)									
<b>Parameters</b>	<b>LPTTS_HANDLE_T phTTS</b> - Specifies an opened text-to-speech handle.									
<b>Return Value</b>	This function 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.									
	<table><tr><th>Constant</th><th>Description</th></tr><tr><td>MMSYSERR_NOERROR</td><td>Normal successful completion (zero).</td></tr><tr><td>MMSYSERR_ERROR</td><td>Failure to wait for pending speech, unable to close the output file, or no output file is open.</td></tr><tr><td>MMSYSERR_INVALIDHANDLE</td><td>The text-to-speech handle was invalid.</td></tr></table>	Constant	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.	
Constant	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.									
<b>Comments</b>	<b>TextToSpeechCloseLogFile()</b> closes any open log file, even if it was opened with the Log command. The application must have called <b>TextToSpeechOpenLogFile()</b> before calling <b>TextToSpeechCloseLogFile()</b> .									
<b>See Also</b>	"TextToSpeechOpenLogFile()" on page 27									

## Function: TextToSpeechCloseWaveOutFile()

Description	The <b>TextToSpeechCloseWaveOutFile()</b> function closes a wave file opened by the <b>TextToSpeechOpenWaveOutFile()</b> function 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 <b>TextToSpeechCloseWaveOutFile</b> (LPTTS_HANDLE_T phTTS)									
Parameters	<b>LPTTS_HANDLE_T phTTS</b> - Specifies an opened text-to-speech handle.									
Return Value	This function 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:									
	<table><tr><th>Constant</th><th>Description</th></tr><tr><td>MMSYSERR_NOERROR</td><td>Normal successful completion (zero).</td></tr><tr><td>MMSYSERR_ERROR</td><td>Failure to wait for pending speech, unable to update the wave file header, or unable to close the wave file.</td></tr><tr><td>MMSYSERR_INVALIDHANDLE</td><td>The text-to-speech handle was invalid.</td></tr></table>	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.	
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 <b>TextToSpeechOpenWaveOutFile()</b> before calling <b>TextToSpeechCloseWaveOutFile()</b> .									
See Also	"TextToSpeechOpenWaveOutFile()" on page 29									

## Function: TextToSpeechEnumLangs()

<b>Description</b>	The <b>TextToSpeechEnumLangs()</b> function retrieves information about what languages are available in the system.
<b>Syntax</b>	DWORD <b>TextToSpeechEnumLangs</b> (LPLANG_ENUM *langs);
<b>Parameters</b>	<b>LPLANG_ENUM *langs</b> - Specifies a LANG_ENUM struct structure to be used to return the language information.
<b>Return Value</b>	DWORD - Returns the size of the struct on success, or 0 on error. No further error information is available at this time.
<b>Comments</b>	<b>TextToSpeechEnumLangs()</b> returns the default language in the registry as the first language of the array of LANG_ENTRY returned by the langs parameter.
<b>Example</b>	<pre> if(TextToSpeechEnumLangs (&amp;languageINFO) == 0) {     MessageBox(NULL, "Unable to allocate Memory", "Error", MB_OK);     return(-1); } - - if(languageINFO-&gt;MultiLang==FALSE)     /* perform nonML processing */ else     for(i=0;i&lt;languageINFO-&gt;Languages;i++)//go through         all languages_     {         languageINFO-&gt;Entries[i].lang_code; //short language name         languageINFO-&gt;Entries[i].lang_name //long language name     } </pre>
<b>See Also</b>	<p>"TextToSpeechCloseLang()" on page 14</p> <p>"TextToSpeechSelectLang()" on page 37</p> <p>"TextToSpeechStartLang()" on page 44</p>



## Function: TextToSpeechGetFeatures()

<b>Description</b>	The <b>TextToSpeechGetFeatures()</b> function retrieves information, in the form of a bitmask, about the features of DECTalk Software.
<b>Syntax</b>	DWORD <b>TextToSpeechGetFeatures</b> (void) ;
<b>Parameters</b>	void
<b>Return Value</b>	<b>DWORD</b> - A bitmask of features supported by DECTalk, maskable to the list supplied in the header file TTSFEAT.H.
<b>Comments</b>	If the DECTalk Multi-Language (ML) engine is running, the ML bit is set to TRUE, as well as any feature bits returned from DECTalk. Future implementation may involve calling <b>TextToSpeechSelectLang()</b> to select the language for which to retrieve information.
<b>Example</b>	<pre>BOOL is_dectalk_ml (unsigned int language_handle) {     unsigned long int feats ;      TextToSpeechSelectLang (NULL, language_handle) ;     feats = TextToSpeechGetFeatures() ;     if (feats &amp; TTS_FEATS_MULTILANG) {;         printf ("DECTalk ML installed and running. \n" ) ;         return TRUE ;     }     printf ("Multi-language DECTalk not found. \n") ;     return FALSE ; }</pre>

Function: `TextToSpeechGetLanguage()` [not supported]

### Warning

The **`TextToSpeechGetLanguage()`** function is not supported for DECtalk Software Version 4.5 or higher. Use of this function causes unpredictable operation and application linking errors.

This function has been replaced by **`TextToSpeechGetCaps()`**.

<b>Description</b>	The <b>TextToSpeechGetLanguage()</b> function returns the current language.							
<b>Syntax</b>	MMRESULT <b>TextToSpeechGetLanguage</b> (LPTTS_HANDLE_T phTTS, LANGUAGE_T *pLanguage)							
<b>Parameters</b>	<b>LPTTS_HANDLE_T phTTS</b> - Specifies a text-to-speech handle. <b>LANGUAGE_T *pLanguage</b> - Specifies a language. Refer to the ttsapi.h file for a list of valid languages, e.g. TTS_AMERICAN_ENGLISH.							
<b>Return Value</b>	<table><tr><th>Constant</th><th>Description</th></tr><tr><td>MMSYSERR_NOERROR</td><td>Normal successful completion (zero).</td></tr><tr><td>MMSYSERR_INVALIDHANDLE</td><td>The text-to-speech handle was invalid.</td></tr></table>		Constant	Description	MMSYSERR_NOERROR	Normal successful completion (zero).	MMSYSERR_INVALIDHANDLE	The text-to-speech handle was invalid.
Constant	Description							
MMSYSERR_NOERROR	Normal successful completion (zero).							
MMSYSERR_INVALIDHANDLE	The text-to-speech handle was invalid.							
<b>See Also</b>	"TextToSpeechSetLanguage() [not supported]" on page 38							



## Function: TextToSpeechGetRate()

<b>Description</b>	The <b>TextToSpeechGetRate()</b> function returns the current setting of the speaking rate.						
<b>Syntax</b>	MMRESULT <b>TextToSpeechGetRate</b> (LPTTS_HANDLE_T phTTS, LPDWORD pdwRate)						
<b>Parameters</b>	<b>LPTTS_HANDLE_T phTTS</b> - Specifies an opened text-to-speech handle. <b>LPDWORD pdwRate</b> - Specifies a pointer to a DWORD that is used to return the speaking rate. Valid values range from 50 to 600 words per minute.						
<b>Return Value</b>	<p>This function returns a value of type MMRESULT. The return value is zero if the call is successful.</p> <p>The return value is one of the following constants:</p> <table><tr><th>Constant</th><th>Description</th></tr><tr><td>MMSYSERR_NOERROR</td><td>Normal successful completion (zero).</td></tr><tr><td>MMSYSERR_INVALIDHANDLE</td><td>The text-to-speech handle was invalid.</td></tr></table>	Constant	Description	MMSYSERR_NOERROR	Normal successful completion (zero).	MMSYSERR_INVALIDHANDLE	The text-to-speech handle was invalid.
Constant	Description						
MMSYSERR_NOERROR	Normal successful completion (zero).						
MMSYSERR_INVALIDHANDLE	The text-to-speech handle was invalid.						
<b>Comments</b>	The current setting of the speaking rate is returned even if the speaking rate change has not yet occurred. This may occur when the <b>TextToSpeechSetRate()</b> function is used without the <b>TextToSpeechSync()</b> function. The speaking-rate change occurs on clause boundaries.						
<b>See Also</b>	"TextToSpeechSetRate()" on page 39						

## Function: TextToSpeechGetSpeaker()

<b>Description</b>	The <b>TextToSpeechGetSpeaker()</b> function returns the value of the identifier for the last voice that has spoken.																				
<b>Syntax</b>	MMRESULT <b>TextToSpeechGetSpeaker</b> (LPTTS_HANDLE_T phTTS, LPSPEAKER_T lpSpeaker)																				
<b>Parameters</b>	<p><b>LPTTS_HANDLE_T phTTS</b> - Specifies an opened text-to-speech handle.</p> <p><b>LPSPEAKER_T lpSpeaker</b> - Specifies a pointer to a DWORD that returns a speaker value from the following list. These symbols are defined in the include file <code>tsapi.h</code>.</p> <table> <tr> <th>Speaker</th><th>Description</th></tr> <tr> <td>PAUL</td><td>Default (male) voice</td></tr> <tr> <td>HARRY</td><td>Full male voice</td></tr> <tr> <td>FRANK</td><td>Aged male voice</td></tr> <tr> <td>DENNIS</td><td>Male voice</td></tr> <tr> <td>BETTY</td><td>Full female voice</td></tr> <tr> <td>URSULA</td><td>Aged female voice</td></tr> <tr> <td>WENDY</td><td>Whispering female voice</td></tr> <tr> <td>RITA</td><td>Female voice</td></tr> <tr> <td>KIT</td><td>Child's voice</td></tr> </table>	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
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																				
<b>Return Value</b>	<p>This function returns a value of type MMRESULT. The return value is zero if the call is successful.</p> <p>The return value is one of the following constants:</p> <table> <tr> <th>Constant</th><th>Description</th></tr> <tr> <td>MMSYSERR_NOERROR</td><td>Normal successful completion (zero).</td></tr> <tr> <td>MMSYSERR_INVALIDHANDLE</td><td>The text-to-speech handle was invalid.</td></tr> </table>	Constant	Description	MMSYSERR_NOERROR	Normal successful completion (zero).	MMSYSERR_INVALIDHANDLE	The text-to-speech handle was invalid.														
Constant	Description																				
MMSYSERR_NOERROR	Normal successful completion (zero).																				
MMSYSERR_INVALIDHANDLE	The text-to-speech handle was invalid.																				
<b>Comments</b>	Even after calling <b>TextToSpeechSetSpeaker()</b> , <b>TextToSpeechGetSpeaker()</b> returns the value for the previous speaking voice until the new voice actually speaks.																				
<b>See Also</b>	"TextToSpeechSetSpeaker()" on page 40																				

## Function: TextToSpeechGetStatus()

<b>Description</b>	The <b>TextToSpeechGetStatus()</b> function returns the status of one or more text-to-speech system parameters.										
<b>Syntax</b>	MMRESULT <b>TextToSpeechGetStatus</b> (LPTTS_HANDLE_T phTTS, LPDWORD dwIdentifier[ ], LPDWORD dwStatus[ ], DWORD dwNumberOfStatusValues)										
<b>Parameters</b>	<p><b>LPTTS_HANDLE_T phTTS</b> - Specifies an opened text-to-speech handle.</p> <p><b>LPDWORD dwIdentifier[ ]</b> - Specifies an array of values of type DWORD that specify TTS parameters for which status values are to be returned in the dwStatus array. These values can be any of the constants defined in include file ttsapi.h and listed below.</p> <p><b>LPDWORD dwStatus[ ]</b> - Specifies an array of type DWORD that is to return status values corresponding to each of the identifiers in the dwIdentifier array.</p> <p><b>DWORD dwNumberOfStatusValues</b> - Specifies the number of entries to return.</p> <table> <tr> <th>Constant in ttsapi.h</th><th>Description</th></tr> <tr> <td>INPUT_CHARACTER_COUNT</td><td>Returns a count of characters that the text-to-speech system is currently processing.</td></tr> <tr> <td>STATUS_SPEAKING</td><td>The status value is TRUE if audio samples are playing and FALSE if no audio sample is playing.</td></tr> <tr> <td>WAVE_OUT_DEVICE_ID</td><td>The current wave output device ID is returned.</td></tr> </table>	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.		
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.										
<b>Return Value</b>	<p>This function returns a value of type MMRESULT. The return value is zero if the call is successful.</p> <p>The return value is one of the following constants:</p> <table> <tr> <th>Constant</th><th>Description</th></tr> <tr> <td>MMSYSERR_NOERROR</td><td>Normal successful completion (zero).</td></tr> <tr> <td>MMSYSERR_INVALIDPARAM</td><td>An invalid parameter was passed.</td></tr> <tr> <td>MMSYSERR_ERROR</td><td>Error obtaining status values.</td></tr> <tr> <td>MMSYSERR_INVALIDHANDLE</td><td>The text-to-speech handle was invalid.</td></tr> </table>	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.
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.										
<b>Comments</b>	The STATUS_SPEAKING status identifier has no meaning if the application is sending speech to a wave file or sending speech to memory.										

## Function: TextToSpeechLoadUserDictionary()

<b>Description</b>	The <b>TextToSpeechLoadUserDictionary()</b> function loads a user-defined pronunciation dictionary into the text-to-speech system.												
<b>Syntax</b>	MMRESULT <b>TextToSpeechLoadUserDictionary</b> (LPTTS_HANDLE_T phTTS, LPSTR pszFileName)												
<b>Parameters</b>	<b>LPTTS_HANDLE_T phTTS</b> - Specifies an opened text-to-speech handle. <b>LPSTR pszFileName</b> - Specifies a pointer to a NULL-terminated string that specifies the name of the user dictionary file to be loaded.												
<b>Return Value</b>	This function 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: <table> <tr> <th>Constant</th><th>Description</th></tr> <tr> <td>MMSYSERR_NOERROR</td><td>Normal successful completion (zero).</td></tr> <tr> <td>MMSYSERR_INVALIDHANDLE</td><td>The text-to-speech handle was invalid.</td></tr> <tr> <td>MMSYSERR_NOMEM</td><td>Unable to allocate memory for dictionary.</td></tr> <tr> <td>MMSYSERR_INVALIDPARAM</td><td>Dictionary file not found or an invalid dictionary file name.</td></tr> <tr> <td>MMSYSERR_ERROR</td><td>Illegal dictionary format or a dictionary is already loaded.</td></tr> </table>	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.
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.												
<b>Comments</b>	This function loads a dictionary created by the <b>windict</b> or <b>user-dict</b> applet (Linux) or the windic applet (Windows). 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, <i>user.dic</i> (or <i>udict_langcode.dic</i> for Linux), at startup if it exists in the home directory.												
<b>See Also</b>	Dictionary Functions ( <i>DECTalk Software Programmer's Guide</i> ) "TextToSpeechUnloadUserDictionary()" on page 57												

## Function: TextToSpeechOpenInMemory()

Description	<p>The <b>TextToSpeechOpenInMemory()</b> function 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 <b>TextToSpeechSpeak()</b> is called. <b>TextToSpeechAddBuffer()</b> 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 <b>TextToSpeechCloseInMemory()</b> is called.</p>												
Syntax	<b>MMRESULT TextToSpeechOpenInMemory</b> (LPTTS_HANDLE_T phTTS, DWORD dwFormat)												
Parameters	<p><b>LPTTS_HANDLE_T phTTS</b> - Specifies an opened text-to-speech handle.</p> <p><b>DWORD dwFormat</b> - Specifies an identifier that determines the audio sample format. It is one of the following constants defined in the include files <code>mmsystem.h</code> and <code>ttsapi.h</code>.</p> <table><tr><th>Constant</th><th>Description</th></tr><tr><td>WAVE_FORMAT_1M08</td><td>Mono, 8-bit 11.025 kHz sample rate</td></tr><tr><td>WAVE_FORMAT_1M16</td><td>Mono, 16-bit 11.025 kHz sample rate</td></tr><tr><td>WAVE_FORMAT_08M08</td><td>Mono, 8-bit ?-law, 8 kHz sample rate</td></tr></table>	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 ?-law, 8 kHz sample rate				
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 ?-law, 8 kHz sample rate												
Return Value	<p>This function returns a value of type <b>MMRESULT</b>. The return value is zero if the call is successful.</p> <p>The return value is one of the following constants:</p> <table><tr><th>Constant</th><th>Description</th></tr><tr><td>MMSYSERR_NOERROR</td><td>Normal successful completion (zero).</td></tr><tr><td>MMSYSERR_INVALIDPARAM</td><td>An invalid parameter or illegal wave output format was passed.</td></tr><tr><td>MMSYSERR_NOMEM</td><td>Unable to allocate memory.</td></tr><tr><td>MMSYSERR_ERROR</td><td>Illegal output state.</td></tr><tr><td>MMSYSERR_INVALIDHANDLE</td><td>The text-to-speech handle was invalid.</td></tr></table>	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.
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.												

**Comments**

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 Speech-To-Memory Mode in the *DECtalk Software Programmer's Guide* 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 Callback Routines and Window Procedures in the *DECtalk Software Programmer's Guide* for details 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()" on page 11

"TextToSpeechCloseInMemory()" on page 13

"TextToSpeechPause()" on page 31

"TextToSpeechReset()" on page 33

"TextToSpeechReturnBuffer()" on page 36

"TextToSpeechSpeak()" on page 42

"TextToSpeechStartup()" [Windows] on page 45

"TextToSpeechStartup()" [Linux] on page 48

"TextToSpeechStartupEx()" on page 51

## Function: TextToSpeechOpenLogFile()

<b>Description</b>	The <b>TextToSpeechOpenLogFile()</b> function opens the specified log file and causes 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 <b>TextToSpeechSpeak()</b> is called. The phonemes and syllables are written using the arpabet alphabet. The text-to-speech system remains in the log-file mode until <b>TextToSpeechCloseLogFile()</b> is called.														
<b>Syntax</b>	MMRESULT <b>TextToSpeechOpenLogFile</b> (LPTTS_HANDLE_T phTTS, LPSTR pszFileName, DWORD dwFlags)														
<b>Parameters</b>	<p><b>LPTTS_HANDLE_T phTTS</b> - Specifies an opened text-to-speech handle.</p> <p><b>LPSTR pszFileName</b> - Specifies a pointer to a NULL terminated string that specifies the name of the log file to be opened.</p> <p><b>DWORD dwFlags</b> - Specifies the type of output. It can contain one or more of the following constants:</p> <table> <tr> <th>Constant</th><th>Description</th></tr> <tr> <td>LOG_TEXT</td><td>Log text</td></tr> <tr> <td>LOG_PHONEMES</td><td>Log phonemes</td></tr> <tr> <td>LOG_SYLLABLES</td><td>Log syllable structure</td></tr> </table>	Constant	Description	LOG_TEXT	Log text	LOG_PHONEMES	Log phonemes	LOG_SYLLABLES	Log syllable structure						
Constant	Description														
LOG_TEXT	Log text														
LOG_PHONEMES	Log phonemes														
LOG_SYLLABLES	Log syllable structure														
<b>Return Value</b>	<p>This function returns a value of type MMRESULT. The return value is zero if the call is successful.</p> <p>The return value is one of the following constants:</p> <table> <tr> <th>Constant</th><th>Description</th></tr> <tr> <td>MMSYSERR_NOERROR</td><td>Normal successful completion (zero).</td></tr> <tr> <td>MMSYSERR_INVALIDPARAM</td><td>An invalid parameter was passed.</td></tr> <tr> <td>MMSYSERR_NOMEM</td><td>Unable to allocate memory.</td></tr> <tr> <td>MMSYSERR_ALLOCATED</td><td>A Log file is already open.</td></tr> <tr> <td>MMSYSERR_ERROR</td><td>Unable to open the output file.</td></tr> <tr> <td>MMSYSERR_INVALIDHANDLE</td><td>The text-to-speech handle was invalid.</td></tr> </table>	Constant	Description	MMSYSERR_NOERROR	Normal successful completion (zero).	MMSYSERR_INVALIDPARAM	An invalid parameter was passed.	MMSYSERR_NOMEM	Unable to allocate memory.	MMSYSERR_ALLOCATED	A Log file is already open.	MMSYSERR_ERROR	Unable to open the output file.	MMSYSERR_INVALIDHANDLE	The text-to-speech handle was invalid.
Constant	Description														
MMSYSERR_NOERROR	Normal successful completion (zero).														
MMSYSERR_INVALIDPARAM	An invalid parameter was passed.														
MMSYSERR_NOMEM	Unable to allocate memory.														
MMSYSERR_ALLOCATED	A Log 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, the logged output is mixed in an unpredictable fashion. If a log file is open already, this function returns an error. The Log voice-control command also 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()" on page 15

"TextToSpeechPause()" on page 31

"TextToSpeechReset()" on page 33

"TextToSpeechSpeak()" on page 42

"TextToSpeechStartup() [Windows]" on page 45

"TextToSpeechStartup() [Linux]" on page 48

"TextToSpeechStartupEx()" on page 51



## Function: TextToSpeechOpenWaveOutFile()

<b>Description</b>	The <b>TextToSpeechOpenWaveOutFile()</b> function opens the specified wave file and causes 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 <b>TextToSpeechSpeak()</b> is called. The text-to-speech system remains in the wave-file mode until <b>TextToSpeechCloseWaveOutFile()</b> is called.														
<b>Syntax</b>	MMRESULT <b>TextToSpeechOpenWaveOutFile</b> (LPTTS_HANDLE_T phTTS, LPSTR pszFileName, DWORD dwFormat)														
<b>Parameters</b>	<p><b>LPTTS_HANDLE_T phTTS</b> - Specifies an opened text-to-speech handle.</p> <p><b>LPSTR pszFileName</b> - Specifies a pointer to a wave file name.</p> <p><b>DWORD dwFormat</b> - Determines the audio sample format. It can be one of the following constants that are defined in include files <code>mmsystem.h</code> and <code>ttsapi.h</code>:</p> <table> <tr> <th>Constant</th><th>Description</th></tr> <tr> <td>WAVE_FORMAT_1M08</td><td>Mono, 8-bit 11.025 kHz sample rate</td></tr> <tr> <td>WAVE_FORMAT_1M16</td><td>Mono, 16-bit 11.025 kHz sample rate</td></tr> <tr> <td>WAVE_FORMAT_08M08</td><td>Mono, 8-bit ?-law, 8 kHz sample rate</td></tr> </table>	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 ?-law, 8 kHz sample rate						
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 ?-law, 8 kHz sample rate														
<b>Return Value</b>	<p>This function returns a value of type MMRESULT. The return value is zero if the call is successful.</p> <p>The return value is one of the following constants:</p> <table> <tr> <th>Constant</th><th>Description</th></tr> <tr> <td>MMSYSERR_NOERROR</td><td>Normal successful completion (zero).</td></tr> <tr> <td>MMSYSERR_INVALIDPARAM</td><td>An invalid parameter or an illegal wave output format was passed.</td></tr> <tr> <td>MMSYSERR_NOMEM</td><td>Memory allocation error.</td></tr> <tr> <td>MMSYSERR_ALLOCATED</td><td>A wave file is already open.</td></tr> <tr> <td>MMSYSERR_ERROR</td><td>Unable to open the wave file or unable to write to the wave file.</td></tr> <tr> <td>MMSYSERR_INVALIDHANDLE</td><td>The text-to-speech handle was invalid.</td></tr> </table>	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.
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.														
<b>Comments</b>	<p>This function automatically resumes audio output if the text-to-speech system is in a paused state by a previously issued <b>TextToSpeechPause()</b> call.</p> <p>The startup function must be called to start the text-to-speech system before calling <b>TextToSpeechOpenWaveOutFile()</b>.</p>														

**See Also**

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

"TextToSpeechCloseWaveOutFile()" on page 16

"TextToSpeechPause()" on page 31

"TextToSpeechReset()" on page 33

"TextToSpeechSpeak()" on page 42

"TextToSpeechStartup() [Windows]" on page 45

"TextToSpeechStartup() [Linux]" on page 48

"TextToSpeechStartupEx()" on page 51

## Function: TextToSpeechPause()

<b>Description</b>	The <b>TextToSpeechPause()</b> function pauses text-to-speech audio output.						
<b>Syntax</b>	MMRESULT <b>TextToSpeechPause</b> (LPTTS_HANDLE_T phTTS)						
<b>Parameters</b>	<b>LPTTS_HANDLE_T phTTS</b> - Specifies an opened text-to-speech handle.						
<b>Return Value</b>	<p>This function returns a value of type MMRESULT. The return value is zero if the call is successful.</p> <p>The return value is one of the following constants:</p> <table> <tr> <th>Constant</th><th>Description</th></tr> <tr> <td>MMSYSERR_NOERROR</td><td>Normal successful completion (zero).</td></tr> <tr> <td>MMSYSERR_INVALIDHANDLE</td><td>The specified device handle is invalid. The system is not speaking or the text-to-speech handle is invalid.</td></tr> </table>	Constant	Description	MMSYSERR_NOERROR	Normal successful completion (zero).	MMSYSERR_INVALIDHANDLE	The specified device handle is invalid. The system is not speaking or the text-to-speech handle is invalid.
Constant	Description						
MMSYSERR_NOERROR	Normal successful completion (zero).						
MMSYSERR_INVALIDHANDLE	The specified device handle is invalid. The system is not speaking or the text-to-speech handle is invalid.						
<b>Comments</b>	<p>This function affects only the audio output and has no effect when writing log files or wave files, or when using the speech-to-memory capability of the text-to-speech system.</p> <p>If the text-to-speech system owns the audio device (that is, <b>OWN_AUDIO_DEVICE</b> was specified in the startup function), then the text-to-speech system remains paused until <b>TextToSpeechResume()</b>, <b>TextToSpeechSync()</b>, <b>TextToSpeechOpenInMemory()</b>, <b>TextToSpeechOpenLogFile()</b>, or <b>TextToSpeechOpenWaveOutFile()</b> is called.</p> <p>If the text-to-speech system does not own the audio device (<b>OWN_AUDIO_DEVICE</b> was NOT specified in the startup function) and <b>TextToSpeechPause()</b> is called while the system is speaking, the text-to-speech system remains paused until the system has completed speaking.</p> <p>In this case, the wave output device is released when <b>TextToSpeechReset()</b> is called. It will also be released if <b>TextToSpeechSync()</b>, <b>TextToSpeechOpenInMemory()</b>, <b>TextToSpeechOpenLogFile()</b>, or <b>TextToSpeechOpenWaveOutFile()</b> is called AND the system has completed speaking.</p> <p><b>NOTE:</b> <b>TextToSpeechPause()</b> will NOT resume audio output if the text-to-speech system is paused by <b>TextToSpeechPause()</b>.</p>						

**See Also**

“TextToSpeechOpenInMemory()” on page 25  
“TextToSpeechOpenLogFile()” on page 27  
“TextToSpeechOpenWaveOutFile()” on page 29  
“TextToSpeechReset()” on page 33  
“TextToSpeechResume()” on page 35  
“TextToSpeechSpeak()” on page 42  
“TextToSpeechSync()” on page 55

## Function: TextToSpeechReset()

<b>Description</b>	The <b>TextToSpeechReset()</b> function flushes all previously queued text from the text-to-speech system and stops any audio output.										
<b>Syntax</b>	MMRESULT <b>TextToSpeechReset</b> (LPTTS_HANDLE_T phTTS, BOOL bReset)										
<b>Parameters</b>	<p><b>LPTTS_HANDLE_T phTTS</b> - Specifies an opened text-to-speech handle.</p> <p><b>BOOL bReset</b> - Specifies one of the following Boolean values:</p> <table> <tr> <th>Value</th><th>Description</th></tr> <tr> <td>FALSE</td><td>Preserves the current mode of the text-to-speech system.</td></tr> <tr> <td>TRUE</td><td>The text-to-speech system is returned to the startup state and any open text-to-speech files are closed. However, this function will NOT resume the text-to-speech system if it has been paused by the <b>TextToSpeechPause()</b> function.</td></tr> </table>	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. However, this function will NOT resume the text-to-speech system if it has been paused by the <b>TextToSpeechPause()</b> function.				
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. However, this function will NOT resume the text-to-speech system if it has been paused by the <b>TextToSpeechPause()</b> function.										
<b>Return Value</b>	<p>The <b>TextToSpeechReset()</b> function 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:</p> <table> <tr> <th>Constant</th><th>Description</th></tr> <tr> <td>MMSYSERR_NOERROR</td><td>Normal successful completion (zero).</td></tr> <tr> <td>MMSYSERR_NOMEM</td><td>Unable to allocate memory.</td></tr> <tr> <td>MMSYSERR_ERROR</td><td>Unable to flush the system.</td></tr> <tr> <td>MMSYSERR_INVALIDHANDLE</td><td>The text-to-speech handle was invalid.</td></tr> </table>	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.
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.										
<b>Comments</b>	<p><b>TextToSpeechReset()</b> flushes all previously queued text and stops all audio output. If the <b>TextToSpeechOpenInMemory()</b> function has enabled writing speech samples to memory, all queued memory buffers are returned to the calling application. If the <b>bReset</b> 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), all files are closed and the text-to-speech system is returned to the startup state.</p> <p><b>TextToSpeechReset()</b> should be called before calling <b>TextToSpeechCloseInMemory()</b>. Failing to do this in a situation where the synthesizer is busy may result in a deadlock.</p>										

**See Also**

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

"TextToSpeechOpenInMemory()" on page 25

"TextToSpeechOpenLogFile()" on page 27

"TextToSpeechOpenWaveOutFile()" on page 29

"TextToSpeechPause()" on page 31

## Function: TextToSpeechResume()

<b>Description</b>	The <b>TextToSpeechResume()</b> function resumes text-to-speech output after it was paused by calling <b>TextToSpeechPause()</b> .	
<b>Syntax</b>	MMRESULT <b>TextToSpeechResume</b> (LPTTS_HANDLE_T phTTS)	
<b>Parameters</b>	<b>LPTTS_HANDLE_T phTTS</b> - Specifies an opened text-to-speech handle.	
<b>Return Value</b>	This function 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:	
	<b>Constant</b>	<b>Description</b>
	MMSYSERR_NOERROR	Normal successful completion (zero).
	MMSYSERR_INVALIDHANDLE	The system was not paused or the text-to-speech handle was invalid.
	MMSYSERR_INVALIDPARAM	The system was not paused.
<b>Comments</b>	This function affects only audio output and has no effect when writing log files or wave files or when writing speech samples to memory.	
<b>See Also</b>	"TextToSpeechPause()" on page 31	

## Function: TextToSpeechReturnBuffer()

<b>Description</b>	The <b>TextToSpeechReturnBuffer()</b> function 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 <b>dwBufferLength</b> element of the TTS_BUFFER_T structure contains the number of samples in the buffer. If no buffer is available, a NULL pointer is returned in <b>ppTTSbuffer</b> .								
<b>Syntax</b>	MMRESULT <b>TextToSpeechReturnBuffer</b> (LPTTS_HANDLE_T phTTS, LPTTS_BUFFER_T *ppTTSbuffer)								
<b>Parameters</b>	<p><b>LPTTS_HANDLE_T phTTS</b> - Specifies an opened text-to-speech handle.</p> <p><b>LPTTS_BUFFER_T *ppTTSbuffer</b> - Specifies a pointer to a pointer to a structure containing the memory buffers. Buffers were supplied by the application to be used while in speech-to-memory mode.</p>								
<b>Return Value</b>	<p>This function returns a value of type MMRESULT. The return value is zero if the call is successful.</p> <p>The return value is one of the following constants:</p> <table> <tr> <th>Constant</th><th>Description</th></tr> <tr> <td>MMSYSERR_NOERROR</td><td>Normal successful completion (zero).</td></tr> <tr> <td>MMSYSERR_ERROR</td><td>Output to memory not enabled or unable to create a system object.</td></tr> <tr> <td>MMSYSERR_INVALIDHANDLE</td><td>The text-to-speech handle was invalid.</td></tr> </table>	Constant	Description	MMSYSERR_NOERROR	Normal successful completion (zero).	MMSYSERR_ERROR	Output to memory not enabled or unable to create a system object.	MMSYSERR_INVALIDHANDLE	The text-to-speech handle was invalid.
Constant	Description								
MMSYSERR_NOERROR	Normal successful completion (zero).								
MMSYSERR_ERROR	Output to memory not enabled or unable to create a system object.								
MMSYSERR_INVALIDHANDLE	The text-to-speech handle was invalid.								
<b>Comments</b>	<p>Most applications do not require this function, because buffers are automatically returned when filled or when a TTS_FORCE flag is passed in the <b>TextToSpeechSpeak()</b> function. The <b>TextToSpeechReturnBuffer()</b> function is provided so that an application can return a buffer before it is filled and, therefore, obtain more speech samples immediately. See the <i>DECTalk Software Programmer's Guide</i> for more information on the TTS_BUFFER_T structure.</p> <p><b>TextToSpeechAddBuffer()</b> must be called before calling <b>TextToSpeechReturnBuffer()</b>.</p>								
<b>See Also</b>	"TextToSpeechAddBuffer()" on page 11								



## Function: TextToSpeechSelectLang()

<b>Description</b>	The <b>TextToSpeechSelectLang()</b> function selects a loaded language for a program thread.
<b>Syntax</b>	BOOL <b>TextToSpeechSelectLang</b> (LPTTS_HANDLE_T reserved, unsigned int lang)
<b>Parameters</b>	<b>LPTTS_HANDLE_T reserved</b> - Reserved; must be NULL. <b>unsigned int lang</b> - Specifies the language handle returned from <b>TextToSpeechStartLang()</b> .
<b>Return Value</b>	Return Value This function returns a value of type BOOL: <ul style="list-style-type: none"> <li>• TRUE if the call is successful</li> <li>• FALSE if the call failed</li> </ul> <p><b>NOTE:</b> The <b>TextToSpeechStartLang()</b> and <b>TextToSpeechSelectLang()</b> functions do not return MMRESULT status values in the manner of the <b>TextToSpeechSetLanguage()</b> function they replace. See the example of <b>TextToSpeechSelectLang()</b> error checking provided below.</p>
<b>Comments</b>	None
<b>Example</b>	<pre> BOOL select_us (unsigned int us_handle) {     if (TextToSpeechSelectLang (NULL, us_handle) ==         FALSE) {         printf ("Select language failed. \n") ;         return FALSE ;     }     return TRUE ; } </pre>
<b>See Also</b>	<p>"TextToSpeechCloseLang()" on page 14</p> <p>"TextToSpeechEnumLangs()" on page 17</p> <p>"TextToSpeechStartLang()" on page 44</p>

Function: `TextToSpeechSetLanguage()` [not supported]

### Warning

The **`TextToSpeechSetLanguage()`** function is not supported for DECtalk Software Version 4.5 or higher. Use of this function causes unpredictable operation and application linking errors.

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

<b>Description</b>	The <b><code>TextToSpeechSetLanguage()</code></b> function selects a language for the text-to-speech system to use as the default language.	
<b>Syntax</b>	<b><code>MMRESULT TextToSpeechSetLanguage</code></b> ( <b><code>LPTTS_HANDLE_T phTTS</code></b> , <b><code>LANGUAGE_T Language</code></b> )	
<b>Parameters</b>	<b><code>LPTTS_HANDLE_T phTTS</code></b> - Specifies a text-to-speech handle. <b><code>LANGUAGE_T Language</code></b> - Specifies a language. Refer to the <code>ttsapi.h</code> file for a list of valid languages, e.g. <code>TTS_AMERICAN_ENGLISH</code> .	
<b>Return Value</b>	This function returns a value of type <code>MMRESULT</code> . The value is zero if the call is successful. The return value is one of the following constants:	
	<b>Constant</b>	<b>Description</b>
	<code>MMSYSERR_NOERROR</code>	Normal successful completion (zero).
	<code>MMSYSERR_INVALIDPARAM</code>	An invalid parameter was passed.
	<code>MMSYSERR_INVALIDHANDLE</code>	The text-to-speech handle was invalid.
<b>See Also</b>	“ <code>TextToSpeechGetLanguage()</code> [not supported]” on page 20	

## Function: TextToSpeechSetRate()

<b>Description</b>	The <b>TextToSpeechSetRate()</b> function sets the text-to-speech speaking rate.								
<b>Syntax</b>	MMRESULT <b>TextToSpeechSetRate</b> (LPTTS_HANDLE_T phTTS, DWORD dwRate)								
<b>Parameters</b>	<b>LPTTS_HANDLE_T phTTS</b> - Specifies an opened text-to-speech handle. <b>DWORD dwRate</b> - Sets the speaking rate. Valid values range from 50 to 600 words per minute.								
<b>Return Value</b>	This function 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: <table data-bbox="729 743 1435 1003"> <tr> <th>Constant</th><th>Description</th></tr> <tr> <td>MMSYSERR_NOERROR</td><td>Normal successful completion (zero).</td></tr> <tr> <td>MMSYSERR_INVALIDPARAM</td><td>An invalid parameter was passed.</td></tr> <tr> <td>MMSYSERR_INVALIDHANDLE</td><td>The text-to-speech handle was invalid.</td></tr> </table>	Constant	Description	MMSYSERR_NOERROR	Normal successful completion (zero).	MMSYSERR_INVALIDPARAM	An invalid parameter was passed.	MMSYSERR_INVALIDHANDLE	The text-to-speech handle was invalid.
Constant	Description								
MMSYSERR_NOERROR	Normal successful completion (zero).								
MMSYSERR_INVALIDPARAM	An invalid parameter was passed.								
MMSYSERR_INVALIDHANDLE	The text-to-speech handle was invalid.								
<b>Comments</b>	The speaking rate change is not effective until the next phrase boundary. All the queued audio encountered before the phrase boundary is unaffected.								
<b>See Also</b>	"TextToSpeechGetRate()" on page 21								

## Function: TextToSpeechSetSpeaker()

<b>Description</b>	The <b>TextToSpeechSetSpeaker()</b> function sets the voice of the speaker that the text-to-speech system is to use.																				
<b>Syntax</b>	MMRESULT <b>TextToSpeechSetSpeaker</b> (LPCTSTR phTTS, SPEAKER_T Speaker)																				
<b>Parameters</b>	<p><b>LPTTS_HANDLE_T phTTS</b> - Specifies an opened text-to-speech handle.</p> <p><b>SPEAKER_T Speaker</b> - Selects a speaker from the following list. These values are defined in include file ttsapi.h.</p> <table> <tr> <th>Speaker</th><th>Description</th></tr> <tr> <td>PAUL</td><td>Default (male) voice</td></tr> <tr> <td>HARRY</td><td>Full male voice</td></tr> <tr> <td>FRANK</td><td>Aged male voice</td></tr> <tr> <td>DENNIS</td><td>Male voice</td></tr> <tr> <td>BETTY</td><td>Full female voice</td></tr> <tr> <td>URSULA</td><td>Aged female voice</td></tr> <tr> <td>WENDY</td><td>Whispering female voice</td></tr> <tr> <td>RITA</td><td>Female voice</td></tr> <tr> <td>KIT</td><td>Child's voice</td></tr> </table>	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
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																				
<b>Return Value</b>	<p>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:</p> <table> <tr> <th>Constant</th><th>Description</th></tr> <tr> <td>MMSYSERR_NOERROR</td><td>Normal successful completion (zero).</td></tr> <tr> <td>MMSYSERR_INVALIDPARAM</td><td>An invalid parameter was passed.</td></tr> <tr> <td>MMSYSERR_INVALIDHANDLE</td><td>The text-to-speech handle was invalid.</td></tr> </table>	Constant	Description	MMSYSERR_NOERROR	Normal successful completion (zero).	MMSYSERR_INVALIDPARAM	An invalid parameter was passed.	MMSYSERR_INVALIDHANDLE	The text-to-speech handle was invalid.												
Constant	Description																				
MMSYSERR_NOERROR	Normal successful completion (zero).																				
MMSYSERR_INVALIDPARAM	An invalid parameter was passed.																				
MMSYSERR_INVALIDHANDLE	The text-to-speech handle was invalid.																				
<b>Comments</b>	The change in speaking voice is not effective until the next phrase boundary. All queued audio encountered before the phrase boundary is unaffected.																				
<b>See Also</b>	"TextToSpeechGetSpeaker()" on page 22																				

## Function: TextToSpeechShutdown()

<b>Description</b>	The <b>TextToSpeechShutdown()</b> function shuts down the text-to-speech system and frees all its system resources.	
<b>Syntax</b>	MMRESULT <b>TextToSpeechShutdown</b> (LPTTS_HANDLE_T phTTS)	
<b>Parameters</b>	<b>LPTTS_HANDLE_T phTTS</b> - Specifies an opened text-to-speech handle.	
<b>Return Value</b>	This function 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:	
	<b>Constant</b>	<b>Description</b>
	MMSYSERR_NOERROR	Normal successful completion (zero).
<b>Comments</b>	MMSYSERR_INVALIDHANDLE	The text-to-speech handle was invalid.
	<b>TextToSpeechShutdown()</b> 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.	
	<b>See Also</b> "TextToSpeechStartup() [Windows]" on page 45 "TextToSpeechStartup() [Linux]" on page 48 "TextToSpeechStartupEx()" on page 51	

## Function: TextToSpeechSpeak()

<b>Description</b>	The <b>TextToSpeechSpeak()</b> function queues a null-terminated string to the text-to-speech system.								
<b>Syntax</b>	MMRESULT <b>TextToSpeechSpeak</b> (LPTTS_HANDLE_T phTTS, LPSTR pszTextString, DWORD dwFlags)								
<b>Parameters</b>	<p><b>LPTTS_HANDLE_T phTTS</b> - Specifies an opened text-to-speech handle.</p> <p><b>LPSTR pszTextString</b> - Specifies a pointer to a null terminated string of characters to be queued.</p> <p><b>DWORD dwFlags</b> - 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>ttapi.h</code>:</p> <table> <tr> <th>Constant</th><th>Description</th></tr> <tr> <td>TTS_NORMAL</td><td>Insert characters in the text-to-speech queue.</td></tr> <tr> <td>TTS_FORCE</td><td>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.</td></tr> </table>	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.		
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.								
<b>Return Value</b>	<p>This function returns a value of type MMRESULT. The return value is zero if the call is successful.</p> <p>The return value is one of the following constants:</p> <table> <tr> <th>Constant</th><th>Description</th></tr> <tr> <td>MMSYSERR_NOERROR</td><td>Normal successful completion (zero).</td></tr> <tr> <td>MMSYSERR_NOMEM</td><td>Unable to allocate memory.</td></tr> <tr> <td>MMSYSERR_INVALIDHANDLE</td><td>The text-to-speech handle was invalid.</td></tr> </table>	Constant	Description	MMSYSERR_NOERROR	Normal successful completion (zero).	MMSYSERR_NOMEM	Unable to allocate memory.	MMSYSERR_INVALIDHANDLE	The text-to-speech handle was invalid.
Constant	Description								
MMSYSERR_NOERROR	Normal successful completion (zero).								
MMSYSERR_NOMEM	Unable to allocate memory.								
MMSYSERR_INVALIDHANDLE	The text-to-speech handle was invalid.								
<b>Comments</b>	<p>While the text-to-speech system is in the startup state, speech samples are routed to the audio device or ignored, depending on whether the startup function flag <code>DO_NOT_USE_AUDIO_DEVICE</code> is clear or set in the <code>dwDeviceOptions</code> parameter of the startup function.</p> <p>If the text-to-speech system is in a special mode (wave-file, log-file, or speech-to-memory modes), the speech samples are handled as the mode dictates.</p> <p>The speaker, speaking rate, and volume also can be changed in the text string by inserting voicecontrol commands, as shown in the following example:</p> <p><b>[ :name paul ]</b> I am Paul. <b>[ :nb ]</b> I am Betty. <b>[ :volume set 50 ]</b> The volume has been set to 50% of the maximum level. <b>[ :rate 120 ]</b> I am speaking at 120 words per minute.</p>								

**See Also**

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

"TextToSpeechOpenInMemory()" on page 25

"TextToSpeechOpenLogFile()" on page 27

"TextToSpeechOpenWaveOutFile()" on page 29

"TextToSpeechStartup() [Windows]" on page 45

"TextToSpeechStartup() [Linux]" on page 48

"TextToSpeechStartupEx()" on page 51

## Function: TextToSpeechStartLang()

<b>Description</b>	The <b>TextToSpeechStartLang()</b> function checks whether the specified language is installed and, if so, loads the language into the DECtalk ML engine.
<b>Syntax</b>	unsigned int <b>TextToSpeechStartLang</b> (char *lang)
<b>Parameters</b>	<b>char *lang</b> - Specifies the language to load; passed as a null-terminated string containing the two character language ID.
<b>Return Value</b>	<p>This function returns a value of type unsigned int:</p> <ul style="list-style-type: none"> <li>• A handle to the loaded language if the call is successful</li> <li>• An error value, with the TTS_LANG_ERROR bit set, if the call failed</li> </ul> <p>If the TTS_LANG_ERROR bit is set, the return can equal one of two values:</p> <ul style="list-style-type: none"> <li>• TTS_NOT_SUPPORTED – The application is not running DECtalk ML</li> <li>• TTS_NOT_AVAILABLE – The requested language is not installed</li> </ul> <p><b>NOTE:</b> The <b>TextToSpeechStartLang()</b> and <b>TextToSpeechSelectLang()</b> functions do not return MMRESULT status values in the manner of the <b>TextToSpeechSetLanguage()</b> function they replace. See the example of <b>TextToSpeechStartLang()</b> error checking provided below.</p>
<b>Comments</b>	<b>TextToSpeechStartLang()</b> must be called before a language can be selected and opened in a multi-language application.
<b>Example</b>	<pre> BOOL start_us (void) {     unsigned int handle ;     handle = TextToSpeechStartLang ("us") ;     if (handle &amp; TTS_LANG_ERROR) {         if (handle == TTS_NOT_SUPPORTED)             printf ("DECtalk ML was 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 ; } </pre>
<b>See Also</b>	<p>"TextToSpeechCloseLang()" on page 14</p> <p>"TextToSpeechEnumLangs()" on page 17</p> <p>"TextToSpeechSelectLang()" on page 37</p>



## Function: TextToSpeechStartup() [Windows]

<b>Description</b>	The <b>TextToSpeechStartup()</b> function for Windows initializes the text-to-speech system, defines the window procedure, checks for valid licenses, and loads the main and user pronunciation dictionaries. A single process can run multiple instances of DECTalk.
<b>Syntax</b>	MMRESULT <b>TextToSpeechStartup</b> (HWND hWnd, LPTTS_HANDLE_T *phTTS, UINT uiDeviceNumber, DWORD dwDeviceOptions)
<b>Parameters</b>	<p><b>HWND hWnd</b> - 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 functions are being used) or when <b>TextToSpeechSpeak()</b> encounters an index mark. A value of NULL is passed if no window handle is desired.</p> <p><b>LPTTS_HANDLE_T *phTTS</b> - Specifies a pointer to a pointer to a text-to-speech handle.</p> <p><b>UINT uiDeviceNumber</b> - Specifies the device number of the wave output device. A value of WAVE_MAPPER can be used to select the first available device.</p> <p><b>DWORD dwDeviceOptions</b> - Specifies how the wave output device is managed. It can be a combination of the following constants defined in include file ttsapi.h:</p>

Constant	Description
OWN_AUDIO_DEVICE	<p>The wave output device is open. No other process can allocate the wave output device until TextToSpeechShutDown() is called.</p> <p>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.</p>
REPORT_OPEN_ERROR	<p>If an attempt is made to open the wave output device while another process owns it, an error message is sent to the calling application.</p>
DO_NOT_USE_AUDIO_DEVICE	<p>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.</p>

**Return Value** This function 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** If a window procedure is defined, DECTalk Software will alert the calling application when 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: United States English.
- Speaking rate: 200 words per minute.
- Speaker: Paul

**NOTE:** (Windows) 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) (*DECTalk Software Programmer's Guide*)  
"TextToSpeechLoadUserDictionary()" on page 24  
"TextToSpeechOpenInMemory()" on page 25  
"TextToSpeechOpenLogFile()" on page 27  
"TextToSpeechOpenWaveOutFile()" on page 29  
"TextToSpeechShutdown()" on page 41  
"TextToSpeechSpeak()" on page 42  
"TextToSpeechStartupEx()" on page 51  
"TextToSpeechUnloadUserDictionary()" on page 57

## Function: TextToSpeechStartup() [Linux]

<b>Description</b>	The <b>TextToSpeechStartup()</b> function for Linux initializes the text-to-speech system, defines the callback routine, checks for valid licenses, and loads the main and user pronunciation dictionaries. A single process can run multiple instances of DECtalk.
<b>Syntax</b>	MMRESULT <b>TextToSpeechStartup</b> (LPTTS_HANDLE_T *phTTS, UINT uiDeviceNumber, DWORD dwDeviceOptions, VOID (*DtCallbackRoutine)(), LONG dwCallbackParameter)
<b>Parameters</b>	<p><b>LPTTS_HANDLE_T *phTTS</b> - Specifies a pointer to a pointer to a text-to-speech handle.</p> <p><b>UINT uiDeviceNumber</b> - Specifies a device number of the wave output device. A value of WAVE_MAPPER can be used to select the first available device.</p> <p><b>DWORD dwDeviceOptions</b> - Specifies how the wave output device is managed. It can be a combination of the device-option constants OWN_AUDIO_DEVICE, REPORT_OPEN_ERROR, and DO_NOT_USE_AUDIO_DEVICE, which are defined in the include file ttsapi.h and described below.</p> <p><b>VOID *(DtCallbackRoutine)()</b> - Specifies a callback routine, which 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 <b>TextToSpeechSpeak()</b> function encounters an index mark. Refer to Callback Routines and Window Procedures in the <i>DECtalk Software Programmer's Guide</i> for information about the argument list for the callback routine.</p> <p>A value of NULL is passed if no callback routine is desired.</p> <p><b>LONG dwCallbackParameter</b> - Specifies a pointer to a user-specified parameter. This is used to pass parameters into the callback routine.</p> <p>A value of NULL should be passed if no user-specified parameters are desired.</p>

Device-Option Constant (ttsapi.h)	Description
OWN_AUDIO_DEVICE	<p>The wave output device is opened. No other process can allocate the wave output device until <b>TextToSpeechShutdown()</b> is called.</p> <p>If OWN_AUDIO_DEVICE is NOT specified, the wave output device is opened after audio is queued by the <b>TextToSpeechSpeak()</b> function. The wave output device is released when the text-to-speech system has completed speaking.</p>
REPORT_OPEN_ERROR	<p>If an attempt is made to open the wave output device while another process owns it, a callback is made to the callback routine that was specified in the DtCallbackRoutine parameter.</p>
DO_NOT_USE_AUDIO_DEVICE	<p>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.</p>

**Return Value**

This function 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**

If a callback routine is defined, DECTalk Software will alert the calling application when 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: United States English.
  - Speaking rate: 200 words per minute.
  - Speaker: Paul

**See Also**

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

Dictionary Functions (Linux) (*DECTalk Software Programmer's Guide*)

"TextToSpeechLoadUserDictionary()" on page 24

"TextToSpeechOpenInMemory()" on page 25

"TextToSpeechOpenLogFile()" on page 27

"TextToSpeechOpenWaveOutFile()" on page 29

"TextToSpeechShutdown()" on page 41

"TextToSpeechSpeak()" on page 42

"TextToSpeechStartupEx()" on page 51

"TextToSpeechUnloadUserDictionary()" on page 57

## Function: TextToSpeechStartupEx()

<b>Description</b>	The <b>TextToSpeechStartupEx()</b> function initializes the text-to-speech system, defines the callback procedure, checks for valid licenses, and loads the main and user pronunciation dictionaries. A single process can run multiple instances of DECTalk.
<b>Syntax</b>	MMRESULT <b>TextToSpeechStartupEx</b> (LPTTS_HANDLE_T *phTTS, UINT uiDeviceNumber, DWORD dwDeviceOptions, VOID (*DtCallbackRoutine)(), LONG dwCallbackParameter)
<b>Parameters</b>	<p><b>LPTTS_HANDLE_T *phTTS</b> - Specifies a pointer to a pointer to a text-to-speech handle.</p> <p><b>UINT uiDeviceNumber</b> - Specifies the device number of the wave output device. A value of WAVE_MAPPER can be used to select the first available device.</p> <p><b>DWORD dwDeviceOptions</b> - Specifies how the wave output device is managed. It can be a combination of the device-option constants OWN_AUDIO_DEVICE, REPORT_OPEN_ERROR, and DO_NOT_USE_AUDIO_DEVICE, which are defined in the include file ttsapi.h and described below.</p> <p><b>VOID *(DtCallbackRoutine)()</b> - Specifies a callback routine, which is used by DECTalk Software to inform the application when the buffer is full (if DECTalk Software in-memory functions are being used) or when the TextToSpeechSpeak() function encounters an index mark. Refer to the <i>DECTalk Software Programmer's Guide</i>, Chapter 3, Introduction to the DECTalk Software API, Callback Routines and Window Procedures for information about the argument list for the callback routine.</p> <p>A value of NULL is passed if no callback routine is desired.</p> <p><b>LONG dwCallbackParameter</b> - Specifies a pointer to a user-specified parameter. This is used to pass parameters into the callback routine.</p> <p>A value of NULL should be passed if no user-specified parameters are desired.</p>

Device-Option Constant (ttsapi.h)	Description
OWN_AUDIO_DEVICE	<p>The wave output device is open. No other process can allocate the wave output device until TextToSpeechShutDown() is called.</p> <p>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.</p>
REPORT_OPEN_ERROR	<p>If an attempt is made to open the wave output device while another process owns it, a callback is made to the callback routine specified in the DtCallbackRoutine parameter.</p>
DO_NOT_USE_AUDIO_DEVICE	<p>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.</p>



**Return Value**

This function 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. (Linux only)
WAVERR_BADFORMAT	Wave output device does not support request format.

**Comments**

If a callback routine is defined, DECTalk Software will alert the calling application when 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: United States English
- Speaking rate: 200 words per minute
- Speaker: Paul

**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.

(Windows) 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 (*DECtalk Software Programmer's Guide*)

"TextToSpeechLoadUserDictionary()" on page 24

"TextToSpeechOpenInMemory()" on page 25

"TextToSpeechOpenLogFile()" on page 27

"TextToSpeechOpenWaveOutFile()" on page 29

"TextToSpeechShutdown()" on page 41

"TextToSpeechSpeak()" on page 42

"TextToSpeechStartupEx()" on page 51

"TextToSpeechUnloadUserDictionary()" on page 57

## Function: TextToSpeechSync()

<b>Description</b>	The <b>TextToSpeechSync()</b> function blocks until all previously queued text is processed.									
<b>Syntax</b>	MMRESULT <b>TextToSpeechSync</b> (LPTTS_HANDLE_T phTTS)									
<b>Parameters</b>	<b>LPTTS_HANDLE_T phTTS</b> - Specifies an opened text-to-speech handle.									
<b>Return Value</b>	This function 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									
	<table><tr><th>Constant</th><th>Description</th></tr><tr><td>MMSYSERR_NOERROR</td><td>Normal successful completion (zero).</td></tr><tr><td>MMSYSERR_ERROR</td><td>Unable to complete queued text.</td></tr><tr><td>MMSYSERR_INVALIDHANDLE</td><td>The text-to-speech handle was invalid.</td></tr></table>	Constant	Description	MMSYSERR_NOERROR	Normal successful completion (zero).	MMSYSERR_ERROR	Unable to complete queued text.	MMSYSERR_INVALIDHANDLE	The text-to-speech handle was invalid.	
Constant	Description									
MMSYSERR_NOERROR	Normal successful completion (zero).									
MMSYSERR_ERROR	Unable to complete queued text.									
MMSYSERR_INVALIDHANDLE	The text-to-speech handle was invalid.									
<b>Comments</b>	This function automatically resumes audio output if the text-to-speech system is in a paused state by a previously issued <b>TextToSpeechPause()</b> call.									
<b>See Also</b>	"TextToSpeechPause()" on page 31									

## Function: TextToSpeechTyping()

<b>Description</b>	<p>The <b>TextToSpeechTyping()</b> function speaks a single letter as quickly as possible, aborting any previously queued speech. This is somewhat slower if <b>TextToSpeechSpeak()</b> has been called since the last <b>TextToSpeechTyping()</b> or <b>TextToSpeechReset()</b> call.</p> <p>This function is primarily useful with the Access32 versions of DECtalk Software. The function exists in non-Access32 versions, but is not fast.</p>
<b>Syntax</b>	void <b>TextToSpeechTyping</b> (LPTTS_HANDLE_T phTTS, char cLetter)
<b>Parameters</b>	<p><b>LPTTS_HANDLE_T phTTS</b> - Specifies an opened text-to-speech handle.</p> <p><b>char cLetter</b> - Specifies the letter to speak.</p>
<b>Return Value</b>	None
<b>Comments</b>	This function should be called only when the application is synthesizing directly to an audio device (not to memory or to a file).

## Function: TextToSpeechUnloadUserDictionary()

<b>Description</b>	The <b>TextToSpeechUnloadUserDictionary()</b> function 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.							
<b>Syntax</b>	MMRESULT <b>TextToSpeechUnloadUserDictionary</b> (LPTTS_HANDLE_T phTTS)							
<b>Parameters</b>	<b>LPTTS_HANDLE_T phTTS</b> - Specifies an opened text-to-speech handle.							
<b>Return Value</b>	This function 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:							
	<table><tr><th>Constant</th><th>Description</th></tr><tr><td>MMSYSERR_NOERROR</td><td>Normal successful completion (zero).</td></tr><tr><td>MMSYSERR_INVALIDHANDLE</td><td>The text-to-speech handle was invalid.</td></tr></table>	Constant	Description	MMSYSERR_NOERROR	Normal successful completion (zero).	MMSYSERR_INVALIDHANDLE	The text-to-speech handle was invalid.	
Constant	Description							
MMSYSERR_NOERROR	Normal successful completion (zero).							
MMSYSERR_INVALIDHANDLE	The text-to-speech handle was invalid.							
<b>Comments</b>	A user dictionary is created using the User Dictionary Build tool.  <b>NOTE:</b> (Windows) 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.							
<b>See Also</b>	Dictionary Functions ( <i>DEctalk Software Programmer's Guide</i> ) "TextToSpeechLoadUserDictionary()" on page 24							

## Function: TextToSpeechVersion()

**Description** The **TextToSpeechVersion()** function requests version information from DECTalk Software that allows a calling application to test for DECTalk Software API (DAPI) compatibility. The function returns a numerically encoded version number and additionally may return a pointer to text information.

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

**Parameters** **LPSTR \*VersionStr** - Returns the address of a pointer to an array of characters containing text information, or NULL for no text information.

**Return Value** This function 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.

## Function: TextToSpeechVersionEx()

<b>Description</b>	The <b>TextToSpeechVersionEx()</b> function returns information about the currently running version of DECTalk Software.	
<b>Syntax</b>	ULONG <b>TextToSpeechVersionEx</b> (LPVERSION_INFO *ver)	
<b>Parameters</b>	<b>LPVERSION_INFO *ver</b> - Returns the address of a pointer to an array of characters with version information. The VERSION_INFO structure is as follows:	
	DWORD	StructSize;
	DWORD	StructVersion;
	WORD	DLLVersion;
	WORD	DTalkVersion;
	LPSTR	VerString;
	LPSTR	Language;
	DWORD	Features;
<b>Return Value</b>	This function 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.	
<b>Example</b>	<pre>BOOL IsDECTalkMLInstalled(void) {     LPVERSION_INFO verinfo;      TextToSpeechVersionEx(&amp;verinfo);     if (ver-&gt;Features &amp; TTS_FEATS_MULTILANG) return TRUE;     return FALSE; }</pre>	





# DECtalk Software 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 [:phoneme]” on page 74 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 “Design Voice [:dv]” on page 135.

**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 speakerdefinition 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
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 “Sync [sync]” on page 82 for more information.

Beginning with SAPI Version 5, you can use DECtalk Software in-line commands in SAPI text buffers. However, the in-line commands are not supported and are ignored in pre-Version 5 SAPI text buffers.

## 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.

<b>SYNTAX:</b>	[:comma DD]
<b>ABBREVIATION:</b>	[:comm DD]
<b>ALTERNATE COMMAND:</b>	[:cp DD] and [:cp 0]
<b>OPTIONS:</b>	none
<b>PARAMETERS:</b>	Pause time in milliseconds
<b>DEFAULT:</b>	160 ms
<b>EXAMPLES:</b>	[: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 speakerdefinition 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 “Design Voice [:dv]” on page 135.

## 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. A non-tone character generates a silent interval between dialed digits. White space characters (tabs, spaces) should not be used as dial tone characters.

<b>SYNTAX:</b>	[:dial YY]
<b>ABBREVIATION:</b>	none
<b>ALTERNATE COMMAND:</b>	none
<b>OPTIONS:</b>	none
<b>PARAMETERS:</b>	String of dial characters (0-9, A, B, C, D, #, *)
<b>DEFAULT:</b>	none
<b>EXAMPLES:</b>	[: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, logfile, or text-to-speech memory) instead, this command fails. See Special Text-To-Speech Modes 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.

<b>SYNTAX:</b>	[:error XX]
----------------	-------------

<b>ABBREVIATION:</b>	[:erro XX]	
<b>ALTERNATE COMMAND:</b>	none	
<b>OPTIONS:</b>	<b>text</b>	Log all text to a file in the current directory called log.txt
	<b>ignore</b>	ignore all errors
	<b>speak</b>	Speak error string in the current command
<b>PARAMETERS:</b>	none	
<b>DEFAULT:</b>	Error string is spoken	
<b>EXAMPLES:</b>	[: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 “Index Marks for Speech Status” on page 92.
- For more information about returning index marks to a calling application, see Callback Routines and Window Procedures 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.

<b>SYNTAX:</b>	[:index mark DD]	
<b>ABBREVIATION:</b>	[:inde mark DD]	
<b>ALTERNATE COMMAND:</b>	none	
<b>OPTIONS:</b>	none	
<b>PARAMETERS:</b>	Numeric index mark value	

<b>DEFAULT:</b>	none
<b>EXAMPLES:</b>	[:index mark 01]

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

## 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 Special Text-To-Speech Modes in the *DECtalk Software Programmer's Guide* for more information.

<b>SYNTAX:</b>	[:log XX YY]	
<b>ABBREVIATION:</b>	none	
<b>ALTERNATE COMMAND:</b>	none	
<b>OPTIONS:</b>	<b>text</b>	Log all text to a file in the current directory called log.txt
	<b>syllables</b>	ignore all errors
	<b>Phonemes</b>	Speak error string in the current command
<b>PARAMETERS:</b>	<b>on</b>	Log all text to a file in the current directory called log.txt
	<b>off</b>	ignore all errors
	<b>set</b>	Speak error string in the current command
<b>DEFAULT:</b>	All of the log options are turned off	
<b>EXAMPLES:</b>	[: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 “Sync [:sync]” on page 82 for information on how to make this command synchronous.

<b>SYNTAX:</b>	[:mode XX YY]	
<b>ABBREVIATION:</b>	none	
<b>ALTERNATE COMMAND:</b>	none	
<b>OPTIONS:</b>	<b>math</b>	Change interpretation of selected symbols
	<b>europe</b>	Select European cardinal pronunciation
	<b>spell</b>	Spell all words
	<b>name</b>	Pronounce all uppercase verbs as proper nouns (see also “Pronounce [:pronounce]” on page 77)
	<b>latin</b>	Not supported
	<b>email</b>	Activates email parsing rules
<b>PARAMETERS:</b>	<b>on</b>	Turns on the specified mode option
	<b>off</b>	Turns off the specified mode option
	<b>set</b>	Turns on the specified mode option while turning off all other mode options
<b>DEFAULT:</b>	All of the mode options are turned off	
<b>EXAMPLES:</b>	[: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 2-2 are treated differently:

**Table 2-2: 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 May 2000 18:07:28 EST
From: john@node.com (John Doe)
Organization: Fonix Corporation
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 a  
Fonix Corporation web site all about the DECTalk products.  
Take a look at URL:

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

Let me know what you think by mailing me at  
 john@aol.node.com 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, 10 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, 10 May 2000 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, 10 May 2000 19:33:57 -0700
%Message-Id:<32094F06.4045@node.com>
%Date: Wed, 10 May 2000 19:20:54 -0700
%From: john Doe <John@node.com>
%Organization: Fonix Corporation
%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.

<b>SYNTAX:</b>	[:name XX]
<b>ABBREVIATION:</b>	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 "Design Voice [:dv]" on page 135.

The Speak applet displays language-specific speaker names, as listed below; however, the Name [:name] in-line command and the Dtsample applet currently support only the English set:

ENGLISH	SPANISH	GERMAN	FRENCH
Paul	Pablo	Paul	Oliver
Harry	Humberto	Hans	Michel
Frank	Francisco	FRank	François
Dennis	Domingo	Dieter	Joël
Betty	Berta	Beate	marjoliane
Ursula	Úrsula	Ursula	Angèle
Wendy	Wendy	Wendy	Nadia
Rita	Rita	Rita	Jacqueline
Kit	Juanito	Karsten	Sébastien

## 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.

<b>SYNTAX:</b>	[:period DD]
<b>ABBREVIATION:</b>	[:peri DD]
<b>ALTERNATE COMMAND:</b>	[:pp DD] and [:pp 0]
<b>OPTIONS:</b>	none
<b>PARAMETERS:</b>	Pause time in milliseconds
<b>DEFAULT:</b>	640 ms
<b>EXAMPLES:</b>	[: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 through Table 4-6.

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 “DECtalk Software Reference Tables” on page 99. For a complete list of stress and syntactic symbols that can be used with phoneme text, see Table 4-7 on page 115 and Table 4-8 on page 115.

**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 or higher.

<b>SYNTAX:</b>	[:phoneme XX1 XX2 YY] or [:phoneme arpabet speak on]	
<b>ABBREVIATION:</b>	[:phon XX1 XX2 YY]	
<b>ALTERNATE COMMAND:</b>	none	
<b>OPTIONS:</b>	<b>arpabet</b>	Set phonetic interpretation to arpabet alphabet. (Currently, this option is the only alphabet allowed.)
	<b>speak</b>	If phoneme interpretation is on, speak encountered phonemes. The speak option is ignored if phoneme interpretation is off.
	<b>silent</b>	If phoneme interpretation is on, do not speak encountered phonemes. The silent option is ignored if phoneme interpretation is off.
<b>PARAMETERS:</b>	<b>on</b>	Set phoneme interpretation on
	<b>off</b>	Set phoneme interpretation off
<b>DEFAULT:</b>	Phonetic interpretation is off	

**EXAMPLES:**

```
[:phoneme arpabet speak on] [axs 'owshiyeyt] asso-
ciate
[:phoneme speak on] [axs 'owshiyeyt] associate
[:phoneme on] [axs 'owshiyeyt] associate
[:phoneme speak off] [axs 'owshiyeyt] pronounced
as axassociate
[:phoneme off] [axs 'owshiyeyt] pronounced as axso-
ciate
[:phoneme silent off] [axs 'owshiyeyt] pronounced as
axsassociate
[:phoneme silent on] [axs 'owshiyeyt] associate not
spoken
```

**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



<b>ALTERNATE COMMAND:</b>	none
<b>OPTIONS:</b>	none
<b>PARAMETERS:</b>	frequency in hertz
<b>DEFAULT:</b>	35
<b>EXAMPLES:</b>	[: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 *Special Text-To-Speech Modes* in the *DECTalk Software Programmer's Guide* for more information.

<b>SYNTAX:</b>	[:play <file>]
<b>ABBREVIATION:</b>	none
<b>ALTERNATE COMMAND:</b>	none
<b>OPTIONS:</b>	none
<b>PARAMETERS:</b>	A directory path and file name
<b>DEFAULT:</b>	none
<b>EXAMPLES:</b>	[: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 (Table 4-12 through Table 4-18) for examples of primary and alternate pronuncia-

tions 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.

<b>SYNTAX:</b>	[:pronounce XX]	
<b>ABBREVIATION:</b>	[:pron XX]	
<b>ALTERNATE COMMAND:</b>	none	
<b>OPTIONS:</b>	<b>alternate</b>	Set phonetic interpretation to arpa-bet alphabet. (Currently, this option is the only alphabet allowed.)
	<b>primary</b>	If phoneme interpretation is on, speak encountered phonemes. The speak option is ignored if phoneme interpretation is off.
	<b>name</b>	If phoneme interpretation is on, do not speak encountered phonemes. The silent option is ignored if phoneme interpretation is off.
<b>PARAMETERS:</b>	none	
<b>DEFAULT:</b>	Uses the primary pronunciation	
<b>EXAMPLES:</b>	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 “Preprocessor Rules for Parsing” on page 155 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.

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

<b>SYNTAX:</b>	[:rate DDD]
<b>ABBREVIATION:</b>	none
<b>ALTERNATE COMMAND:</b>	none
<b>OPTIONS:</b>	none
<b>PARAMETERS:</b>	Rate in words per minute
<b>EXAMPLES:</b>	[: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.

<b>SYNTAX:</b>	[:say XX]
<b>ABBREVIATION:</b>	none
<b>ALTERNATE COMMAND:</b>	none

<b>OPTIONS:</b>	<b>clause</b>	Speak on end of clause.
	<b>word</b>	Speak on end of word.
	<b>letter</b>	Speak on end of letter.
	<b>filtered letter</b>	Speak on end of letter, ignoring control characters, such as “vertical tab” and “line feed”
	<b>line</b>	Speak on end of line.
<b>PARAMETERS:</b>	none	
<b>DEFAULT:</b>	[:say clause]	
<b>EXAMPLES:</b>	[: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 “Preprocessor Rules for Parsing” on page 155 for information on the preprocessor rules for parsing.

<b>SYNTAX:</b>	[:skip XX]	
<b>ABBREVIATION:</b>	none	
<b>ALTERNATE COMMAND:</b>	none	
<b>OPTIONS:</b>	<b>all</b>	Skip all preprocessing
	<b>cpg</b>	Skip codepage translation
	<b>none</b>	Do not skip anything
<b>PARAMETERS:</b>	none	
<b>DEFAULT:</b>	The default is set to [:skip none].	
<b>EXAMPLES:</b>	[:skip cpg]	
	[:skip all]	
	[:skip none]	

**NOTE:** This command allows only one option to be in effect at a time; in the example, [:skip cpq] 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 in-line 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.

<b>SYNTAX:</b>	[:sync]
<b>ABBREVIATION:</b>	none
<b>ALTERNATE COMMAND:</b>	none
<b>OPTIONS:</b>	none
<b>PARAMETERS:</b>	none
<b>DEFAULT:</b>	N/A
<b>EXAMPLES:</b>	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.

<b>SYNTAX:</b>	[:tone DD, DD]	
<b>ABBREVIATION:</b>	none	
<b>ALTERNATE COMMAND:</b>	none	
<b>OPTIONS:</b>	none	
<b>PARAMETERS</b>	<b>frequency:</b>	Sets the frequency to the desired level
	<b>duration:</b>	Tone duration in milliseconds
<b>DEFAULT:</b>	none	
<b>EXAMPLES:</b>	[: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.

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

<b>SYNTAX:</b>	[:volume XX DD1 DD2]	
<b>ABBREVIATION:</b>	[:volu XX DD1 DD2]	
<b>ALTERNATE COMMAND:</b>	none	
<b>OPTIONS:</b>	<b>lset</b>	Sets the left channel to the desired level
	<b>lup</b>	Increases the left channel by the desired amount
	<b>ldown</b>	Decreases the left channel by the desired amount
	<b>rset</b>	Sets the right channel to the desired level
	<b>rup</b>	Increases the right channel by the desired amount
	<b>rdown</b>	Decreases the right channel by the desired amount
	<b>sset</b>	Sets the left channel to the DD1 amount and the right channel to the DD2 amount



<b>PARAMETERS:</b>	Volume amount
<b>DEFAULT:</b>	none
<b>EXAMPLES:</b>	[ :volume set 80 60]



# 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 “Design Voice [:dv]” on page 135.

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 4-8 “Syntactic Symbols” on page 115 lists these symbols.

# Developing an Electronic Mail-Reading Application

DECtalk Software supplies an email parser for Windows and for UNIX (not for MS-DOS). See the email option of “Mode [:mode]” on page 69 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 4-8 “Syntactic Symbols” on page 115 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 “Pronounce [:pronounce]” on page 77 for more information.
  - Replace the database word with its phoneme text. For this option, you must have phoneme interpretation turned on. See “Phoneme Interpretation [:phoneme]” on page 74 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 through Table 4-6 for a complete list of phoneme symbols, including the silent underscore ( \_ ) symbol. See Chapter 4 “DECtalk Software Reference Tables” on page 99 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 “DECtalk Software In-Line Commands” on page 61) 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 ' ehd ] yesterday that . . .

**NOTE:** For words that have two pronunciations (homographs), see the Homograph tables (Table 4-12 through Table 4-18).

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.

- 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 4-7 “Stress Symbols” on page 115 for a complete list of stress symbols.

## 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 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.

## 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 4-7 “Stress Symbols” on page 115 and Table 4-8 “Syntactic Symbols” on page 115.

### 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 "Phoneme Interpretation [:phoneme]" on page 74 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.

[ :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.
```

**If the selected language supports pitch rise and fall symbols [ / \ ] and emphatic stress symbols [ ' ], use them 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.
- 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] Here is the list [apple, banana,
strawberry].
```

```
[ :phoneme arpabet speak on Ladies and Gentlemen
```

- Beginning with SAPI Version 5, you can use DECtalk Software in-line commands in SAPI text buffers. However, the in-line commands are not supported and are ignored in pre-Version 5 SAPI text buffers.

# DECtalk Software Reference Tables





DECtalk Software reference tables include:

- Phonemic Symbols Listed By Language
- Stress and Syntactic Symbols
- Phonemes Listed in Unicode Sequence
- Pitch and Duration of Tones
- Homographs
- Supported SAPI Functions
- Supported SAPI Version 5 Features

## Phonemic Symbols Listed By Language

The phonemic symbol can be used to replace words that are spoken incorrectly. See “Phoneme Interpretation [:phoneme]” on page 74 for information on how to use phonemic symbols.

DECtalk Software provides a unified phoneme set for all supported languages, allowing you to specify phonemes from different languages within the context of your current language.

This section lists the phonemic symbols DECTalk Software uses for each supported language, as follows:

- Table 4-1 Phonemic Symbols - U.S. English
- Table 4-2 Phonemic Symbols - U.K. English
- Table 4-3 Phonemic Symbols - Castilian Spanish
- Table 4-4 Phonemic Symbols - Latin American Spanish
- Table 4-5 Phonemic Symbols - German
- Table 4-6 Phonemic Symbols - French

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 4-7 lists the supported stress symbols.

Phonemes can also be given duration and pitch attributes to create special effects, such as singing. See Table 4-10 “Phoneme Syntax for Singing” on page 119 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,” [ \* 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 “white-horse.”

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 or higher.

## U.S. English

**Table 4-1: Phonemic Symbols - U.S. English**

ASCKY	DT index	DT internal	Example	Arpabet	Unicode	Unicode Character Name
_	0	SIL	(silence)	_	U+5F	Low line
i	1	US_IY	bEAn	iy	U+69	Latin small letter I
l	2	US_IH	pIt	ih	U+26A	Latin small letter Capital I
e	3	US_EY	bAY	ey	U+65	Latin small letter E
E	4	US_EH	pEt	eh	U+25B	Latin small letter open E
@	5	US_AE	pAt	ae	U+E6	Latin small letter AE
a	6	US_AA	pOt	aa	U+251	Latin small letter Alpha
A	7	US_AY	bUY	ay	U+61, U+26A	Latin small letter A + Latin small capital I
	W	8	US_AW	brOW	aw	U+61, U+28A
^	9	US_AH	pUtt	ah	U+28C	Latin small letter turned V
c	10	US_AO	bOUght	ao	U+254	Latin small letter O
o	11	US_OW	nO	ow	U+6F, U+28A	Latin small letter O + Latin small letter Upsilon
O	12	US_OY	bOY	oy	U+254, U+26A	Latin small letter open O + Latin small letter capital I
U	13	US_UH	pUt	uh	U+28A	Latin small letter Upsilon
u	14	US_UW	bOOn	uw	U+75	Latin small letter U
R	15	US_RR	anothER	rr	U+25A	Latin small letter Schwa with hook
Y	16	US_YU	cUte	yu	U+6A, U+75	Latin small letter J + Latin small letter U
x	17	US_AX	About	ax	U+259	Latin small letter Schwa
	18	US_IX	kissEs	ix	U+268	Latin small letter I with stroke
l	19	US_IR	pEEr	ir	U+69, U+2B4	Latin small letter I + modifier letter small turned R
R	20	US_ER	pAir	er		
a	21	US_AR	bARn	ar	U+251, U+2B4	Latin small letter Alpha + modifier letter small turned R
c	22	US_OR	bOrn	or	U+254, U+2B4	Latin small letter open O + modifier letter small turned R
U	23	US_UR	pOOr	ur	U+28A, U+2B4	Latin small letter Upsilon + modifier letter small turned R
w	24	US_W	Why	w	U+77	Latin small letter W
Y	25	US_Y	Yank	yx	U+6A	Latin small letter J

**Table 4-1: Phonemic Symbols - U.S. English**

ASCKY	DT index	DT internal	Example	Arpabet	Unicode	Unicode Character Name
r	26	US_R	Rat	r	U+52	Latin capital letter R
l	27	US_LL	Lad	l	U+6C	Latin small letter L
h	28	US_HX	Had	hx	U+68	Latin small letter H
R	29	US_RX	coRe	rx	U+279	Latin small letter turned R with hook
l	30	US_LX	untiL	lx	U+26B	Latin small letter l with middle tilde
m	31	US_M	Mad	m	U+6D	Latin small letter M
n	32	US_N	Nat	n	U+6E	Latin small letter N
G	33	US_NX	baNG	nx	U+14B	Latin small letter Eng
L	34	US_EL	dangLe	el	U+6C, U+329	Latin small letter L combining vertical line below
D	35	US_DZ	wiDth	dz	U+64, U+32F	Latin small letter D + combining inverted breve below
N	36	US_EN	burdeN	en	U+6E, U+329	Latin small letter N + combining vertical line below
f	37	US_F	Fat	f	U+66	Latin small letter F
v	38	US_V	Vat	v	U+76	Latin small letter V
T	39	US_TH	THin	th	U+3B8	Greek small letter Theta
D	40	US_DH	THen	dh	U+F0	Latin small letter Eth
s	41	US_S	Sap	s	U+73	Latin small letter S
z	42	US_Z	Zap	z	U+7A	Latin small letter Z
S	43	US_SH	SHeep	sh	U+283	Latin small letter Esh
Z	44	US_ZH	meaSure	zh	U+292	Latin small letter Ezh
p	45	US_P	Pat	p	U+70	Latin small letter P
b	46	US_B	Bad	b	U+62	Latin small letter B
t	47	US_T	Tack	t	U+74	Latin small letter T
d	48	US_D	Dad	d	U+64	Latin small letter D
k	49	US_K	Cad	k	U+6B	Latin small letter K
g	50	US_G	Game	g	U+67	Latin small letter G
&	51	US_DX	riDer	dx	Internal use only	
Q	52	US_TX	baTTen	tx	U+74, U+294	Latin small letter T + Latin letter glottal stop
q	53	US_Q	we eat	q	U+294	Latin letter glottal stop
C	54	US_CH	CHeap	ch	U+2A7	Latin small letter Tesh digraph
J	55	US_JH	Jeep	jh	U+2A4	Latin small letter Dezh digraph

**Table 4-1: Phonemic Symbols - U.S. English**

ASCKY	DT index	DT internal	Example	Arpabet	Unicode	Unicode Character Name
F	56	US_DF	wriTer	df	Internal use only	
	57	US_TZ		tz		Hebrew complement
	58	US_CZ		cz		Hebrew complement

## U.K. English

**Table 4-2: Phonemic Symbols - U.K. English**

ASCKY	DT index	DT internal	Example	Arpabet
_	0	SIL	(silence)	_
i	1	UK_IY	bEAn	iy
l	2	UK_IH	pIt	ih
e	3	UK_EY	bAY	ey
E	4	UK_EH	pEt	eh
@	5	UK_AE	pAt	ae
a	6	UK_AA	pOt	aa
A	7	UK_AY	bUY	ay
W	8	UK_AW	brOW	aw
^	9	UK_AH	pUtt	ah
c	10	UK_AO	bOUght	ao
o	11	UK_OW	nO	ow
O	12	UK_OY	bOY	oy
U	13	UK_UH	pUt	uh
u	14	UK_UW	bOOOn	uw
R	15	UK_RR	anothER	rr
Y	16	UK_YU	cUte	yu
x	17	UK_AX	About	ax
	18	UK_IX	kissEs	ix
l	19	UK_IR	pEEr	ir
R	20	UK_ER	pAlr	er
a	21	UK_AR	bARn	ar
c	22	UK_OR	bOrn	or
U	23	UK_UR	pOOOr	ur
w	24	UK_W	Why	w
Y	25	UK_Y	Yank	yx
r	26	UK_R	Rat	r
l	27	UK_LL	Lad	l
h	28	UK_HX	Had	hx
	29	UK_OH		oh
l	30	UK_LX	untiL	lx
m	31	UK_M	Mad	m

**Table 4-2: Phonemic Symbols - U.K. English**

ASCKY	DT index	DT internal	Example	Arpabet
n	32	UK_N	Nat	n
G	33	UK_NX	baNG	nx
L	34	UK_EL	dangLe	el
D	35	UK_DZ	wiDth	dz
N	36	UK_EN	burdeN	en
f	37	UK_F	Fat	f
v	38	UK_V	Vat	v
T	39	UK_TH	THin	th
D	40	UK_DH	THeN	dh
s	41	UK_S	Sap	s
z	42	UK_Z	Zap	z
S	43	UK_SH	SHeep	sh
Z	44	UK_ZH	meaSure	zh
p	45	UK_P	Pat	p
b	46	UK_B	Bad	b
t	47	UK_T	Tack	t
d	48	UK_D	Dad	d
k	49	UK_K	Cad	k
g	50	UK_G	Game	g
&	51	UK_DX	riDer	dx
Q	52	UK_TX	baTTen	tx
q	53	UK_Q	we eat	q
C	54	UK_CH	CHeap	ch
J	55	UK_JH	Jeep	jh
F	56	UK_DF	wriTer	df

## Castilian Spanish

**Table 4-3: Phonemic Symbols - Castilian Spanish**

ASCKY	DT index	DT internal	Example	Arpabet
_	0	SIL	(silence)	_
	1	SP_A	Palabra	a
	2	SP_E	Leo	e
	3	SP_I	Hilo	i
	4	SP_O	Hola	o
	5	SP_U	Lunes	u
	6	SP_WX	(Rounded dip- thong semiv.)	wx
	7	SP_YX (Unround dip- thong semiv.)	yx	
	8	SP_RR	Rama	rr
	9	SP_L	Luna	l
	10	SP_LL	Calle	ll
	11	SP_M	Mama'	m
	12	SP_N	Nana	n
	13	SP_NH	Munoz	nh
	14	SP_F	Feo	f
	15	SP_S	Casa	s
	16	SP_J	Caja	j
	17	SP_TH	Caza	th
	18	SP_BH	Haba	bh
	19	SP_DH	Hada	dh
	20	SP_GH	Haga	gh
	21	SP_YH	Yate (affricate)	yh
	22	SP_P	Papa'	p
	23	SP_B	Barco	b
	24	SP_T	Tela	t
	25	SP_D	Dama	d
	26	SP_K	Casa	k
	27	SP_G	Gasa	g
	28	SP_CH	Charco	ch
	29	SP_Y	Haya (fricative)	y



**Table 4-3: Phonemic Symbols - Castilian Spanish**

ASCKY	DT index	DT internal	Example	Arpabet
	30	SP_R	Sara	r
	31	SP_Q	~n (offglide)	q
	32	SP_Z	Desde	z
	33	SP_W	Hueso	w
	34	SP_NX	Mango	nx
	35	SP_V	Afgano	v
	36	SP_IX	~n (offglide)	ix
	37	SP_MX	Infierno (nf)	mx
	38	SP_PH	Observar	ph

## Latin American Spanish

**Table 4-4: Phonemic Symbols - Latin American Spanish**

ASCKY	DT index	DT internal	Example	Arpabet
–	0	SIL	(silence)	–
	1	LA_A	Palabra	a
	2	LA_E	Leo	e
	3	LA_I	Hilo	i
	4	LA_O	Hola	o
	5	LA_U	Lunes	u
	6	LA_WX	(Rounded dip- thong semiv.)	wx
	7	LA_YX	(Unround dip- thong semiv.)	yx
	8	LA_RR	Rama	rr
	9	LA_L	Luna	l
	10	LA_LL	Calle	ll
	11	LA_M	Mama'	m
	12	LA_N	Nana	n
	13	LA_NH	Munoz	nh
	14	LA_F	Feo	f
	15	LA_S	Casa	s
	16	LA_J	Caja	j
	17	LA_TH	Caza	th
	18	LA_BH	Haba	bh
	19	LA_DH	Hada	dh
	20	LA_GH	Haga	gh
	21	LA_YH	Yate (affricate)	yh
	22	LA_P	Papa'	p
	23	LA_B	Barco	b
	24	LA_T	Tela	t
	25	LA_D	Dama	d
	26	LA_K	Casa	k
	27	LA_G	Gasa	g
	28	LA_CH	Charco	ch
	29	LA_Y	Haya (fricative)	y
	30	LA_R	Sara	r

**Table 4-4: Phonemic Symbols - Latin American Spanish**

ASCKY	DT index	DT internal	Example	Arpabet
	31	LA_Q	~n (offglide)	q
	32	LA_Z	Desde	z
	33	LA_W	Hueso	w
	34	LA_NX	Mango	nx
	35	LA_V	Afgano	v
	36	LA_IX	~n (offglide)	ix
	37	LA_MX	Infierno (nf)	mx
	38	LA_PH	Observar	ph

## German

**Table 4-5: Phonemic Symbols - German**

ASCKY	DT index	DT internal	Example	Arpabet
–	0	SIL	(silence)	–
	1	GR_A	mAnn	a
	2	GR_E	Englisch	e
	3	GR_AE	hAEtte	ae
	4	GR_EX	gabE	ex
	5	GR_I	mIt	i
	6	GR_O	pOst	o
	7	GR_OE	kOEnnen	oe
	8	GR_U	mUnd	u
	9	GR_UE	IUEcke	ue
	10	GR_AH	sAgen	ah
	11	GR_EH	gEben	eh
	12	GR_AZ	wAEhlen	az
	13	GR_IH	IIEb	ih
	14	GR_OH	mOnd	oh
	15	GR_OZ	mOEgen	oz
	16	GR_UH	hUt	uh
	17	GR_UZ	hUEten	uz
	18	GR_EI	kIEld	ei
	19	GR_AU	hAUs	au
	20	GR_EU	hEUte	eu
	21	GR_AN	pENSion	an
	22	GR_IM	tIMbre	im
	23	GR_UM	parfUM	um
	24	GR_ON	fONdue	on
	25	GR_J	Ja	j
	26	GR_L	Luft	l
	27	GR_RR	Rund	rr
	28	GR_R	waR	r
	29	GR_H	Hut	h
	30	GR_M	Mut	m
	31	GR_N	NeiN	n

**Table 4-5: Phonemic Symbols - German**

ASCKY	DT index	DT internal	Example	Arpabet
	32	GR_NG	riNG	ng
	33	GR_EL	nabEL	el
	34	GR_EM	grossEM	em
	35	GR_EN	badEN	en
	36	GR_F	Fall	f
	37	GR_V	Was	v
	38	GR_S	meSSen	s
	39	GR_Z	doSe	z
	40	GR_SH	SCHule	sh
	41	GR_ZH	Genie	zh
	42	GR_CH	niCHt	ch
	43	GR_KH	noCH	kh
	44	GR_P	Park	p
	45	GR_B	Ball	b
	46	GR_T	Turm	t
	47	GR_D	Dort	d
	48	GR_K	Kalt	k
	49	GR_G	Gast	g
	50	GR_Q	Be_amtet	q
	51	GR_PF	PFerd	pf
	52	GR_TS	Zahl	ts
	53	GR_DJ	Gin	dj
	54	GR_TJ	maTSCH	tj
	55	GR_KS	Extra	ks

## French

**Table 4-6: Phonemic Symbols - French**

ASCKY	DT index	DT internal	Example	Arpabet
_	0	SIL	(silence)	_
	1	FR_A		a
	2	FR_A3		a3
	3	FR_E2		e2
	4	FR_AU		au
	5	FR_E		e
	6	FR_E1		e1
	7	FR_EU		eu
	8	FR_I		i
	9	FR_O		o
	10	FR_O6		o6
	11	FR_OU		ou
	12	FR_U		u
	13	FR_AN		an
	14	FR_IN		in
	15	FR_ON		on
	16	FR_UN		un
	17	FR_AP		ap
	18	FR_L		l
	19	FR_R		r
	20	FR_W		w
	21	FR_WU		wu
	22	FR_Y		y
	23	FR_CH		ch
	24	FR_F		f
	25	FR_J		j
	26	FR_RX		rx
	27	FR_S		s
	28	FR_V		v
	29	FR_Z		z
	30	FR_B		b
	31	FR_D		d

**Table 4-6: Phonemic Symbols - French**

ASCKY	DT index	DT internal	Example	Arpabet
	32	FR_G		g
	33	FR_K		k
	34	FR_P		p
	35	FR_T		t
	36	FR_GN		gn
	37	FR_M		m
	38	FR_N		n
	39	FR_NG		ng
	40	FR_SG		sg

## Stress and Syntactic Symbols

Table 4-7 and Table 4-8 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 “Phoneme Interpretation [:phoneme]” on page 74 for more information.

**Table 4-7: 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 4-8: Syntactic Symbols**

Symbol	Name	Indicates	Unicode
-	Hyphen	syllable boundary	U+2D
*	Asterisk	morpheme boundary	U+2A

**Table 4-8: Syntactic Symbols**

Symbol	Name	Indicates	Unicode
#	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 4-9: English 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	?				



**Table 4-9: English Phonemes in Unicode Sequence**

Unicode	Unicode Character Name	ASCKY	DT index	DT internal	Example	Arpabet
U+52	Latin capital letter R	R	26	US_R	Rat	r
U+5C	Reverse solidus	\				
U+5F	Low line	_	0	US_SIL	(silence)	_
U+61, U+26A	Latin small letter A + Latin small capital I	A	7	US_AY	bUY	ay
U+61, U+28A	Latin small letter A + Latin small capital I	W	8	US_AW	brOW	aw
U+62	Latin small letter B	b	46	US_B	Bad	b
U+64, U+32F	Latin small letter D + combining inverted breve below	D	35	US_DZ	WiDth	dz
U+64	Latin small letter D	d	48	US_D	Dad	d
U+65	Latin small letter E	e	3	US_EY	bAY	ey
U+66	Latin small letter F	f	37	US_F	Fat	f
U+67	Latin small letter G	g	50	US_G	Game	g
U+68	Latin small letter H	h	28	US_HX	Had	hx
U+69, U+2B4	Latin small letter I + modifier letter small turned R	I	19	US_IR	pEEr	ir
U+69	Latin small letter I	i	1	US_IY	bEAn	iy
U+6A, U+75	Latin small letter J + Latin small letter U	Y	16	US_YU	cUte	yu
U+6A	Latin small letter J	Y	25	US_Y	Yank	yx
U+6B	Latin small letter K	k	49	US_K	Cad	k
U+6C, U+329	Latin small letter L + combining vertical line below	L	34	US_EL	dangLe	el
U+6C	Latin small letter L	l	27	US_LL	Lad	l
U+6D	Latin small letter M	m	31	US_M	Mad	m
U+6E, U+329	Latin small letter N + combining vertical line below	N	36	US_EN	burdeN	en
U+6E	Latin small letter N	n	32	US_N	Nat	n
U+6F, U+28A	Latin small letter O + Latin small letter upsilon	o	11	US_OW	nO	ow
U+70	Latin small letter P	p	45	US_P	Pat	p
U+73	Latin small letter S	s	41	US_S	Sap	s
U+74	Latin small letter T	t	47	US_T	Tack	t
U+74, U+294	Latin small letter T + Latin letter glottal stop	Q	52	US_TX	baTTen	tx

**Table 4-9: English Phonemes in Unicode Sequence**

Unicode	Unicode Character Name	ASCKY	DT index	DT internal	Example	Arpabet
U+75	Latin small letter U	u	14	US_UW	bOO <u>n</u>	uw
U+76	Latin small letter V	v	38	US_V	Vat	v
U+77	Latin small letter W	w	24	US_W	Why	w
U+7A	Latin small letter Z	z	42	US_Z	Zap	z
U+E6	Latin small letter AE	@	5	US_AE	pAt	ae
U+F0	Latin small letter Eth	D	40	US_DH	T <u>H</u> en	dh
U+14B	Latin small letter Eng	G	33	US_NX	ba <u>NG</u>	nx
U+251, U+2B4	Latin small letter Alpha + modifier letter small turned R	a	21	US_AR	bAR <u>n</u>	ar
U+251	Latin small letter Alpha	a	6	US_AA	p <u>O</u> t	aa
U+254, U+26A	Latin small letter open O + Latin small letter capital I	O	12	US_OY	bO <u>Y</u>	oy
U+254, U+2B4	Latin small letter open O + modifier letter small turned R	c	22	US_OR	bO <u>r</u> n	or
U+254	Latin small letter O	c	10	US_AO	bO <u>U</u> ght	ao
U+259	Latin small letter Schwa	x	17	US_AX	Ab <u>o</u> ut	ax
U+25A	Latin small letter Schwa with hook	R	15	US_RR	an <u>o</u> th <u>ER</u>	rr
U+25B	Latin small letter open E	E	4	US_EH	p <u>E</u> t	eh
U+268	Latin small letter I with stroke		18	US_IX	kiss <u>E</u> s	ix
U+26A	Latin small letter Capital I	I	2	US_IH	pl <u>I</u> t	ih
U+26B	Latin small letter I with middle tilde	I	30	US_LX	un <u>t</u> iL	lx
U+279	Latin small letter turned R with hook	R	29	US_RX	co <u>R</u> e	rx
U+283	Latin small letter Esh	S	43	US_SH	SH <u>ee</u> p	sh
U+28A, U+2B4	Latin small letter Upsilon + modifier letter small turned R	U	23	US_UR	pO <u>O</u> r	ur
U+28A	Latin small letter Upsilon	U	13	US_UH	p <u>U</u> t	uh
U+28C	Latin small letter turned V	^	9	US_AH	p <u>U</u> tt	ah
U+292	Latin small letter Ezh	Z	44	US_ZH	mea <u>S</u> ure	zh
U+294	Latin letter glottal stop	q	53	US_Q	we e <u>a</u> t	q
U+2A4	Latin small letter Dezh digraph	J	55	US_JH	Je <u>e</u> p	jh
U+2A7	Latin small letter Tesh digraph	C	54	US_CH	CH <u>ee</u> p	ch
U+2C8	Modifier letter vertical line	‘				
U+28CC	Modifier letter low vertical line	`				

**Table 4-9: English Phonemes in Unicode Sequence**

Unicode	Unicode Character Name	ASCKY	DT index	DT internal	Example	Arpabet
U+3B8	Greek small letter Theta	T	39	US_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 4-11 provides the pitch numbers, associated notes, and frequencies you need to code a phonemic sequence to produce musical sounds.

Figure 4-1 on page 120 is the code for the song, “Happy Birthday.” The command syntax for coding musical sequences is found in Table 4-10 on page 119. You can use the phonemic table for your language (see Table 4-1 through Table 4-6) to decode the phoneme symbols.

**Table 4-10: Phoneme Syntax for Singing**

<b>SYNTAX:</b>	[phoneme <duration, pitch number>]	
<b>OPTIONS:</b>	none	
<b>PARAMETERS:</b>	<b>duration</b>	Tone duration in milliseconds.
	<b>pitch number</b>	Pitch number from
<b>DEFAULT:</b>	none	
<b>EXAMPLES:</b>	See Figure 4-1 on page 120	

**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 4-11: 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

**Table 4-11: Tone table**

Pitch Number	Note	Pitch	Vocal Ranges			
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 (permit); if, however, the word is used as a verb, the accent is on the second syllable (permit). 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 before a word to obtain an alternative pronunciation for the 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.

This section lists the homograph phonetics in alphabetical groups, as follows:

- Table 4-12 Homograph Phonetics - (A)
- Table 4-13 Homograph Phonetics - (B-C)
- Table 4-14 Homograph Phonetics - (D-G)
- Table 4-15 Homograph Phonetics - (I-L)
- Table 4-16 Homograph Phonetics - (M-P)
- Table 4-17 Homograph Phonetics - (R)
- Table 4-18 Homograph Phonetics - (S-W)

**Table 4-12: 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 ' ihk t	' ae d ihk t
advocate	' aed v axk eyt	' aed v ax k axt
affix	' aef ihk s	axf ' ihk s
ally	' ael ay	axl ' ay
alternate	' aol t rrn ax t	' ao l t rrn ey t
animate	' aen ihm eyt	' aen ih m ax t
annex	' aen ehk s	axn ' ehk s
appropriate	axp r ' owp r iyaxt	axp r ' owp r iy eyt
arithmetic	axr ' ihythm ax t ixk	aer ixthm ' eht ixk
articulate	aar t ' ihk yxel eyt	aar t ' ih k yxel axt
associate	axs ' owshiyeyt	axs ' owshiyaxt
attribute	axt r ' ihbyut	' aet r ixbyut
august	' aog axs t	aog ' ahs t

**Table 4-13: Homograph Phonetics - (B-C)**

<b>Spelling</b>	<b>Primary</b>	<b>Alternate</b>
bass	b ' eys	b ' aes
baton	b axt ' 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 ixx 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 ' i h k t	k ' aan vi h k t
coordinate	k ow ' aor d en eyt	kow ' aor d ixn axt

**Table 4-14: Homograph Phonetics - (D-G)**

<b>Spelling</b>	<b>Primary</b>	<b>Alternate</b>
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



**Table 4-15: Homograph Phonetics - (I-L)**

<b>Spelling</b>	<b>Primary</b>	<b>Alternate</b>
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

**Table 4-16: Homograph Phonetics - (M-P)**

<b>Spelling</b>	<b>Primary</b>	<b>Alternate</b>
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	prf ' 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

**Table 4-17: Homograph Phonetics - (R)**

<b>Spelling</b>	<b>Primary</b>	<b>Alternate</b>
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 ixf ' yuz	r ' ehf yus
reject	rixjh'ehkt	r'iyjhehkt
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

**Table 4-18: Homograph Phonetics - (S-W)**

<b>Spelling</b>	<b>Primary</b>	<b>Alternate</b>
segment	s ' ehg m ixn t	s ehg m ' ehn t
separate	s ' ehp axr eyt	s ' ehp 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 (Windows 95/98/ME/NT/2000/XP Only)

Table 4-19 shows the Microsoft Speech API (SAPI) functions that DECtalk Software supports for Windows systems. See the Microsoft documentation and the Microsoft web site for more information on the SAPI functions.

**Table 4-19: Supported Functions of the Microsoft Speech API**

<b>SAPI Interface</b>	<b>Supported Functions</b>	<b>Functions With Limited Support</b>	<b>Unsupported Functions</b>
ITTSAttributes	SpeedGet SpeedSet VolumeGet VolumeSet PitchGet PitchSet	RealtimeGet (always returns 1) RealTimeSet (always returns 1)	
ITTSBufNotifySink	TextDataStart TextDataDone BookMarks WordPosition		
ITTSCentral	AudioPause AudioReset AudioResume ModeGet PosnGet TextData \com=string\ \emp\ \mrk=number\ \pau=number\ \pit=number\ \rst\ \spd=number\ \vol=number\ \prn=string=string\ Register Unregister Inject ToFileTime		TextData \chr=string[[,string...]]\ \ctx=string\ \eng[[;GUID]]:command\ \pro=number\ \prt=string\ \vce=character-act=value [[,character-act=value...]]\ Phoneme
ITTSDialogs	All functions		
ITTSEnum	All functions		
ITTSFind	none		All functions
IllexPronounce	none		All functions

**Table 4-19: Supported Functions of the Microsoft Speech API**

<b>SAPI Interface</b>	<b>Supported Functions</b>	<b>Functions With Limited Support</b>	<b>Unsupported Functions</b>
ITTSNotifySink	AudioStart AudioStop	AttribChanged	
Iaudio	All functions		
IaudioDest	All functions		
IaudioDestNotify	All functions		
IaudioSourceNotifySink	none		All functions

**SAPI Notes**

- 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.
- Beginning with SAPI Version 5, you can use DECtalk Software in-line commands in SAPI text buffers. However, the in-line commands are not supported and are ignored in pre-Version 5 SAPI text buffers.

## Supported SAPI Version 5 Features (Windows 98/ME/NT/2000/XP Only)

Table 4-20 shows the Microsoft Speech API (SAPI) Version 5 features that DECTalk Software supports for Windows systems. See the Microsoft documentation and the Microsoft web site for more information on the SAPI Version 5 features.

**Table 4-20: Supported Features of the Microsoft Speech API, Version 5**

<b>SAPI Version 5 Feature</b>	<b>Supported Functions</b>
ISpTTS Engine	Speak Skip GetOutputFormat SetRate SetVolume
Eventing	SAPI V5 required events
TTS XML Markup	Bookmark Silence Spell Pronounce Rate Volume Pitch
Real Time Rate/Volume	Real time rate change (takes effect at the beginning of the next clause) Real time volume change Audio State Tests Purge Before Speak Speak Pause Stop Speak Destroy
Lexicon Tests	User lexicon test Application lexicon test
Multiple Instance Tests	Multiple instance test Multiple instance with shared voice test
Emph	All functions
Phoneme & Viseme Events	All functions
PartOfSp	All functions

### **SAPI V5 Notes**

- DECTalk Software for Windows currently supports SAPI Version 5 functions for Windows 98, Windows ME, Windows NT, and Windows 2000; not for Windows 95 or Windows CE.
- SAPI Version 5 support currently is for U.S. English only.
- The Context feature currently is not supported. DECTalk accepts the context tags but does not use them.
- Beginning with SAPI Version 5, you can use DECTalk Software in-line commands in SAPI text buffers. For example, you can access a command such as `[ :tone 500 300 ]` using an XML tag such as `<dectalk tone 500 300>.`
- You can use DECTalk Software phonemes in SAPI text buffers. For example: `<dectalk phon arpa on>[hx' ehl ow][ :phon arpa off>.` With DECTalk's unified phoneme set, you can specify phonemes from different languages within the context of your current language.



# 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 Chapter 2 “DECtalk Software In-Line Commands” on page 61 and Chapter 3 “Using In-Line Commands” on page 87.

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.

DECtalk Software supports many speaker-definition options that can be modified. However, please be aware that approximating all the variations that can characterize a speaker -- sex, age, head size and shape, larynx size and behavior, pitch range, pitch and timing habits, dialect, and emotional state – can be very time-consuming

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**

<b>SYNTAX:</b>		[:dv XX YY]
<b>OPTIONS:</b>		
<b>ap</b>	Average pitch, in Hz	
<b>as</b>	Assertiveness, in %	
<b>b4</b>	Fourth formant bandwidth, in Hz	
<b>b5</b>	Fifth formant bandwidth, in Hz	
<b>bf</b>	Baseline fall, in Hz	
<b>br</b>	Breathiness, in decibels (dB)	
<b>f4</b>	Fourth formant resonance frequency, in Hz	
<b>f5</b>	Fifth formant resonance frequency, in Hz	
<b>g1</b>	Gain of cascade formant resonator 1, in dB	
<b>g2</b>	Gain of cascade formant resonator 2, in dB	
<b>g3</b>	Gain of cascade formant resonator 3, in dB	
<b>g4</b>	Gain of cascade formant resonator 4, in dB	
<b>g5</b>	Loudness of the voice, in dB	
<b>gf</b>	Gain of frication source, in dB	
<b>gh</b>	Gain of aspiration source, in dB	
<b>gn</b>	Gain of nasalization, in dB	
<b>gv</b>	Gain of voicing source, in dB	
<b>hr</b>	Hat rise, in Hz	
<b>hs</b>	Head size, in %	
<b>la</b>	Laryngealization, in %	
<b>lx</b>	Lax breathiness, in %	
<b>nf</b>	Number of fixed samples of open glottis	

<b>pr</b>	Pitch range, in %
<b>qu</b>	Quickness, in %
<b>ri</b>	Richness, in %
<b>sm</b>	Smoothness, in %
<b>sr</b>	Stress rise, in Hz
<b>sx</b>	Sex 1 (male) or 0 (female)
<b>save</b>	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 5-2: 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

**Table 5-2: Speaker Definitions for All DECtalk Software Voices**

Param	Paul	Harry	Frank	Dennis	Betty	Ursula	Wendy	Rita	Kit
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 5-3.

**Table 5-3: 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?
```

**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 5-4.

**Table 5-4: 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 5-5.

**Table 5-5: Fundamental Frequency Contour Options**

<b>bf</b>	Baseline fall, in Hz
<b>hr</b>	Hat rise, in Hz
<b>sr</b>	Stress rise, in Hz
<b>as</b>	Assertiveness, in %
<b>qu</b>	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 “DECtalk Software Reference Tables” on page 99 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 “DECTalk Software Reference Tables” on page 99 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 5-6.

**Table 5-6: 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()** function is called 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.



# 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 in-line command or modify it with the Punctuation in-line command. Three sets of rules apply to the parsing process:

- Email parsing rules
- Punctuation parsing rules
- General parsing rules

## Email Parsing Rules

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:

## Punctuation Parsing Rules

When the preprocessor encounters punctuation, it interprets each punctuation mark (by default) as a guide to speaking the text normally, unless you use in-line 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 Parsing 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 fortysecond 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-colons, or colons. 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 receives speech output samples that are written as 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 **TextToSpeech-CloseLogFile()** 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 that 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, speech samples are written into memory buffers rather than sent to an audio device. The **TextToSpeechAddBuffer()** function supplies the text-to-speech system with the memory buffers that it needs. The **TextToSpeechOpenInMemory()** function causes the text-to-speech system to enter 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()** function or the **TextToSpeechStartupEx()** function.

**startup state**

Startup state indicates that the **TextToSpeechStartup()** function or the **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” on page 164.

**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**

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

**voice-control command**

A DECtalk Software in-line 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-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.

**wave form output**

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

**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 in which 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.



# Index

## **Symbols**

.dic file 164

.tab file 168

## **A**

allophone 163

API 9

applet 163

Application Programming Interface 9

arpabet 163

Assertiveness 147

Average Pitch 148

## **B**

Baseline Fall 145

Breathiness 142

## **C**

Cascade Vocal Tract 151

clause boundary 163

clause mode 163

clause terminator 163

Comma Pause 63, 65, 163

Comma pause 94

## **D**

Design Voice 64, 65, 135

Dial Tones 64, 66

Documentation conventions 5

dynamic engine 164

## **E**

Email Parsing 90, 157

emphatic stress 164

Error 64, 66

## **F**

falling intonation 164

flush 164

functions 9

## **G**

Gender 138

General Parsing 159

## **H**

Hat Rise 145

Head Size 138

heuristic 164

homograph 91, 121, 164

## **I**

Index Mark 64, 67, 92, 164

in-line commands 63

- Comma Pause 63, 65

- Design Voice 64, 65

- Dial Tones 64, 66

- Error 64, 66

- Index Mark 64, 67

- Log 64, 68

- Mode 64, 69

- Name 64, 72

- Period Pause 64, 74

- Phoneme Interpretation 64, 74

- Pitch 64, 76

- Play Wave Files 64, 77

- Pronounce 64, 77

- Punctuation 64, 78

- Rate Selection 64, 79

- Say 64, 80

- Skip 64, 81

- Sync 64, 82

- Tone 64, 83

- Volume 64, 83

Intonation 89, 144, 165

## **L**

Laryngealization 144

Lax Breathiness 142

letter mode 165

Log 64, 68

log file 165

log-file mode 165



Loudness 149

## **M**

Mode 64, 69

morpheme 165

## **N**

Name 64, 72

Nopen Fixed 143

## **O**

Optimizing 91

Overloads 149

## **P**

Period Pause 64, 74, 165

Period pause 94

phoneme 166

phoneme arpabet command 166

Phoneme Interpretation 64, 74

phoneme string 166

Phonemes

- Castilian Spanish 108

- French 114

- German 112

- Latin American Spanish 110

- U.K. English 106

- U.S. English 103

- Unicode Sequence 116

phonemic mode 166

Phonemic Symbols 101

phonemic transcription 166

phonemicize 166

phrase boundary 166

Pitch 64, 76, 119, 144

pitch control symbols 166

Pitch Range 148

Play Wave Files 64, 77

primary stress 167

Pronounce 64, 77

proper name 167

Punctuation 64, 78

Punctuation Parsing 157

## **Q**

Quickness 147

## **R**

Rate Selection 64, 79

Relative Gains 149

Rhythm 89

Richness 143

## **S**

SAPI Functions 129

SAPI Version 5 131

Save 152

Say 64, 80

secondary stress 167

silence phonemes 167

Singing 119

Skip 64, 81

Smoothness 142

Sound Source Gains 150

Speaking Rate 93

speech-to-memory mode 167

startup function 167

startup state 167

static engine 168

Stress 89

Stress Rise 146

Stress Symbols 115

Sync 64, 82

syntactic function words 168

Syntactic Symbols 115

## **T**

TextToSpeechAddBuffer 11

TextToSpeechCloseInMemory 13

TextToSpeechCloseLang 14

TextToSpeechCloseLogFil 15  
TextToSpeechCloseWaveOutFile 16  
TextToSpeechEnumLangs 17  
TextToSpeechGetCaps 18  
TextToSpeechGetFeatures 19  
TextToSpeechGetLanguage 20  
TextToSpeechGetRate 21  
TextToSpeechGetSpeaker 22  
TextToSpeechGetStatus 23  
TextToSpeechLoadUserDictionary 24  
TextToSpeechOpenInMemory 25  
TextToSpeechOpenLogFile 27  
TextToSpeechOpenWaveOutFile 29  
TextToSpeechPause 31  
TextToSpeechReset 33  
TextToSpeechResume 35  
TextToSpeechReturnBuffer 36  
TextToSpeechSelectLang 37  
TextToSpeechSetLanguage 38  
TextToSpeechSetRate 39  
TextToSpeechSetSpeaker 40  
TextToSpeechShutdown 41  
TextToSpeechSpeak 42  
TextToSpeechStartLang 44  
TextToSpeechStartup 45, 48  
TextToSpeechStartupEx 51  
TextToSpeechSync 55  
TextToSpeechTyping 56  
TextToSpeechUnloadUserDictionary 57  
TextToSpeechVersion 58  
TextToSpeechVersionEx 59  
Text-Tuning 95  
Tone 64, 83, 119

## **U**

Unicode Sequence 116  
user dictionary 168

user dictionary builder 168

## **V**

Voice Definitions 137

Voice Quality 141

voice-control command 168

Volume 64, 83

## **W**

WAVE file 168

wave form output 169

wave-file mode 169

word boundary 169

word mode 169



Fonix Corporation  
9350 South 150 East Ste 700  
Sandy, UT 84070-2715  
801-553-6600  
[www.fonix.com](http://www.fonix.com)