DECtalk® Software
Programmer's Guide

June 2003

This guide introduces application programmers to DECtalk® Software and the DECtalk Software API. It also explains the basics of DECtalk applets and DECtalk multi-language programming.

Revision / Update Information:

This document supersedes the DECtalk Software Programmer's 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

June 2003

iii

Contents

Figures

Tables

Preface

Purpose and Audience

This guide, *DECtalk Software Programmer's Guide*, is for the application programmer who wants to design and build text-to-speech applications with DECtalk® Software. Use this guide in conjunction with the *DECtalk Software Reference Guide* and the *DECtalk Software Installation 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 or development of a DECtalk Software application.

The guide is organized as follows:

Chapter 1 Introduction to DECtalk Software
Chapter 2 Using the Applets to Learn DECtalk Software Basics
Chapter 3 Introduction to the DECtalk Software API
Chapter 4 Basics of Multi-Language Programming
Glossary Definitions of Terms Used in DECtalk Documentation

What's New in DECtalk Software V5.00?

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 ÍCommand - The right arrow key indicates an abbreviated instruction for choosing a command from a menu. For example, File ÍExit 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. `Courier type` also indicates file names.
`User Input` **Boldface** type in interactive examples indicates information you enter from the keyboard. For example:
A:>**SETUP**
XX YY and
XXn YYn
In DECtalk Software in-line command syntax, XX and YY indicate options and parameters. When more than one choice of options or parameters is allowed, the symbol XXn or YYn with n replaced by a numeral indicates each option or parameter in the symbolic representations, such as [:phoneme XX1 XX2 YY].
Note that the number of characters in the symbolic representation does **not** represent the number of characters allowed in the actual option or parameter name.
DD and DDn In DECtalk Software in-line command syntax, DD indicates a decimal (base 10) value. When more than one decimal values are allowed, the symbol DDn with n replaced by a numeral represents each allowed value, such as [:volume XX DD1DD2].
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.

Chapter 1 —
Introduction to DECtalk Software

DECtalk Software provides programming resources to support applications that require text-to-speech output. This chapter provides a general overview of DECtalk Software. Topics include:

- Features and functions
- Components
- Programming aids
- Text-to-Speech server (Windows 95/98/ME/NT/2000/XP only)
- Using the components
- How it works

Features and Functions

DECtalk Software enables applications to extend the capabilities of your computer by turning text files into spoken words.

High-Quality Speech and Word Pronunciation Accuracy

DECtalk Software provides the latest version of DECtalk speech synthesis technology. With only a standard sound card for audio output, the programming resources provided by DECtalk Software allow applications to accurately read ASCII text from a variety of sources, such as electronic mail and word processors. Nine different voices are provided, and users can control voice pitch, rate of speech, and word or phrase emphasis. DECtalk Software has a large built-in dictionary that enables accurate pronunciation of individual words and enhances their rhythmic naturalness.

Letter Mode, Word Mode, and Clause Mode

DECtalk Software can speak single characters immediately, without waiting for an entire clause to be buffered. DECtalk Software also provides normal clause buffering for natural speech. DECtalk Software can speak letters, words, phrases, clauses, paragraphs, and whole documents. DECtalk Software allows the application to terminate speech immediately instead of waiting for the buffered text to complete processing.

Short Command Strings

Many of the DECtalk Software in-line command strings can be abbreviated for greater ease of use in applications.

Pronunciation Heuristics

DECtalk Software includes pronunciation heuristics that recognize and parse unpronounceable sequences, such as sequences of uppercase initials (FBI, AAA, and so forth) and sequences with no vowels (CBS and NBC, for example).

Components

DECtalk Software components include:

- Sample graphical applets Dtsample (Windows only) and Speak
- Sample command-line applet Say
- Programming aids, including the DECtalk Software API (DAPI), the

Microsoft Speech API (SAPI) for Windows, in-line voice control commands, and dictionary facilities

- Text-to-speech (TTS) server for Windows
- Source code for selected sample programs

Dtsample Applet (Windows Only)

The Dtsample applet and components are provided to give you some ideas on how to get started with your application. Both the graphic user interface and source components used to develop that user interface are included as part of the Dtsample program. See Figure 1-1 for a visual overview of the Dtsample applet dialog.

See *Using the Applets to Learn DECtalk Software Basics* for additional description of the Dtsample program. Note that supported languages not installed with DECtalk Software are grayed out on the Languages menu. Languages shown in dark lettering are installed and ready to be used.

1-4

Figure 1-1 Visual Overview of the Dtsample Applet

❶
❷
❸
❺
❹

Legend

å Dialog

 Menu bar: File, Edit, Speak, Voice, Rate, Languages, and Help menus

ê Start, pause, and stop push buttons

. Speech speed-control slider

. Edit window for text input

1-5

Speak Applet

The Speak applet is included as a practical example of how the DECtalk Software API can be used to produce an innovative Text-To-Speech application that maximizes user interaction through a highly graphical interface. Figure 1-2 provides a visual overview of the Speak applet.

Figure 1-2 Visual Overview of the Speak Applet

Paul Betty Harry Frank Dennis Kit Ursula Rita Wendy

å

ê

.

.
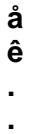
Legend

å Menu bar: File, Edit, and Help menus **.** Speech speed-control slider

 Voice-activation buttons **.** Start, pause, and stop push buttons

ê Edit window for text input

Notes

- The Speak applet is not provided and not supported for Windows CE systems.
- English, Spanish, German, or French speaker names are displayed, depending on the current default language. See the Name [:name] command description for lists of names.
- The faces display as cartoon characters when using VGA 16-color mode.

1-6

Say Command-Line Applet

The DECtalk Say command-line applet lets you run DECtalk Software from the operating system command prompt window. Say provides many of the standard DECtalk input and output options available in the Speak and Dtsample applets and in the DECtalk Software API. The general command syntax is as follows:

```
say [options] [text]
```

For a detailed description of the available command-line options, see *Using the Say Command-Line Applet*.

Notes

- The Say applet is not provided and not supported for Windows CE systems.
- On UNIX systems the `say` command must be entered in lower case letters.

Programming Aids

The DECtalk Software programming aids include application programming interfaces (APIs), in-line voice control commands, and dictionary facilities. Each component is introduced in this section and explained in detail in Chapter 3.

Application Programming Interfaces (APIs)

DECtalk Software supports two text-to-speech APIs:

- The DECtalk Software API (DAPI)
- The Microsoft Speech API (SAPI) [Windows 95/98/ME/NT/2000/XP only]

DAPI is a custom extension to the Multimedia PC (MPC) API specified by the Windows operating system. The DAPI function set gives you a flexible method of using and controlling DECtalk Software functionality from within your application. These functions perform a wide range of tasks associated with Text-To-Speech systems. See Chapter 1 in the *DECtalk Software Reference Guide* for the complete list of DAPI functions.

In addition to the DAPI, DECtalk Software supports the Microsoft Speech API (SAPI) for Windows 95/98/ME/NT/2000 systems and SAPI Version 5.0 for Windows 98/ME/NT/2000 systems. This allows DECtalk Software to work as an OLE server in any OLE application environment. Information about SAPI is available in the Microsoft Developer Network (MSDN) CD-ROM under the *Microsoft Software Development Kit (SDK)*. See Chapter 4 in the *DECtalk Software Reference Guide* for a list of the SAPI functions supported by DECtalk Software. For more information about SAPI, refer to Microsoft documentation and the Microsoft web site.

In-Line Voice Control Commands

DECtalk Software includes in-line commands that control voice characteristics. You can use these commands to perform simple voice-control operations, such as changing the speaking rate or speaking voice while DECtalk Software is speaking.

In-line commands can be inserted into the text entered into the edit window available in most applications supplied with DECtalk Software. In-line commands also can be included in data buffers passed to the DECtalk Software

Dynamic Link Libraries (DLLs) by way of various **TextToSpeech...()** API function calls. For more information, see the *DECtalk Software Reference Guide*.

DECtalk Software also has voice-control commands to modify the characteristics of each voice, control intonation and stress within written text, or

create special effects, such as singing.

In-line commands have special syntax rules and components that you need to use when you insert them into files. A few simple commands and command components are illustrated in Figure 1-3. Refer to the *DECtalk Software Reference Guide* and online help for more information on in-line commands.

Figure 1-3 In-Line Command Components

```
[:rate 150]Hello. How are you?
Now I'm [:rate 150][:nb][r ' iyl iy] thrilled!
```

å

ê

```
[:rate 150][:nb]Hello. How are you?
```

Legend

å In-line commands are inserted into ASCII text files, begin with a colon, and are always inserted between brackets. (The command here tells DECtalk Software to speak this line at 150 words per minute.) Most commands have parameters. (For example, : rate 150 = rate of 150 words per minute.)

   Two or more commands can be inserted after each other by enclosing each command within a set of square brackets. (For example, the rate and voice selections are shown here.)

ê Phonetic spellings of words can be included also. Phonetic spellings are enclosed within a set of square brackets. (For example, [r ' iyl iy] for really) Note: if you want to use phonetic spellings, you must use the **[:phoneme arpabet speak on]** command to turn on recognition of phonetic spellings.

Dictionary Facilities

DECtalk Software has two pronunciation dictionaries — a large internal (builtin) dictionary and a user-defined dictionary. With the large built-in dictionary, you can easily use many proper names and normally unpronounceable sequences, such as uppercase initials, in applications. With the user dictionary build tool, you can load application-specific words or cultural or language-

1-9

specific terms into the user dictionary. A sample user-dictionary file is installed with the software.

The location of pronunciation dictionaries is determined by the following:

• On Windows systems, if your text-to-speech engine is a dynamic engine, which uses DLLs, or a static engine, which uses only LIBs, the default locations of the main and user dictionaries are defined in registry entries, as presented in *Chapter 3*.

• On Linux systems, the location of the dictionaries is the DICTIONARIES directory, as defined in `locations.sh` in your installation directory.

• On Tru64 UNIX systems, the locations of the dictionaries are given in `dtuser.lib`.

Text-to-Speech Server (Windows 95/98/ME/NT/2000/XP Only)

The Text-to-Speech (TTS) server for Windows makes it possible for any Windows application that can call Dynamic Data Exchange (DDE) and that contains ASCII text to speak that text through DECtalk Software.

Under Windows, the TTS server is launched automatically whenever you start Microsoft Mail or manually whenever you click on the Text-to-Speech server icon in the DECtalk Software program group.

1-10

Using the Components

DECtalk Software applications and application-building components are
intended for two specific audiences: the application programmer and the general
user of text-to-speech applications developed with the DECtalk Software API.

Application Programmer
Application using DECtalk
Command Set
User Dictionary
Builder
DECtalk API
Other operating system specific programming services

As a DECtalk Software application programmer, you can use the DECtalk
Software programming aids and API components to create applications that use
DECtalk Software. These applications can incorporate text files and use
DECtalk Software in-line commands as permanent parts of the application.

General User
Abbreviated Command Set
Customized User Dictionary
Application
using
DECtalk
ASCII Text
File

The general user converts ASCII text files to speech by using an application that
incorporates DECtalk Software. By using an abbreviated set of DECtalk
Software in-line commands and the user-defined dictionary, the general user
1-11
can fine-tune the basic pronunciation and voice characteristics defined in the
application.
1-12

How It Works
DECtalk Software converts ASCII text into speech output through a speech
synthesizer. Two ways to feed text into the speech synthesizer are:
• Programming DECtalk Software API function calls in your own application
program
• Using the DECtalk Dtsample applet user interface (Windows only)
The flow of the text-to-speech process is shown in Figure 1-4.

Figure 1-4 Flow of the DECtalk Software Text-To-Speech Conversion Process
Text Input
❶
Command
and Text
Parser
❷
Letter to
Sound
❸
Phoneme
to Voice
Module
❹
Vocal
Tract
Module

❺
Legend

å Text is selected for processing by DECtalk Software.

A sentence parser breaks the input stream into separate words and locates some clause boundaries (indicated by commas and other punctuation marks as well as by special words loaded in the DECtalk Software internal dictionary). The sentence parser also recognizes and deals with phonemic symbols and commands that you might have added to the input text.

A word parser breaks words into their component parts, dividing words into their final pronounceable forms. Strings of text that do not form pronounceable words are spelled out letter by letter. A number formatter is used if the text contains numerals. The number formatter applies the rules for many common number formats and converts the numbers into words.

ê A dictionary lookup routine searches the pronunciation dictionaries. DECtalk Software has a built-in dictionary of many commonly used words. DECtalk Software also has a user dictionary for programmers and general users that can be filled with words specific to an application. This dictionary and how to load it are described in Chapter 3.

A letter-to-sound module uses a set of pronunciation rules to assign phonemic form and lexical stress patterns to words not found in the dictionary. See Chapter 3 for more information on modifying the phonemic form of words, and the DECtalk Software Reference Guide for enhancing special voice qualities, such as emphasis and singing.

1-13
Legend

. A phrase structure module recombines all phonemic output from the dictionary search and other modules. Duration of phonemes and pitch commands is computed for the clause, and appropriate sound variants are selected for those phonemes that can be pronounced in more than one way.

The phoneme-to-voice module processes clauses passed from the phrase structure module and converts them to control signals for the speech synthesizer. This module modifies the clauses by changing the phonemes/allophones into parameters that determine the natural resonant frequencies of the vocal tract (formants), and sound source amplitudes. The control parameters are sent to the speech synthesizer for output.

. The DECtalk speech synthesizer computes a speech waveform with acoustic characteristics that are determined by the synthesizer control commands.

2-1

Chapter 2 —
Using the Applets to Learn DECtalk Software Basics

This chapter introduces the basic operations involved in DECtalk Software voice and program control. If you are a general user, this chapter gives you the information you need to use the Dtsample, Speak, Say, and User Dictionary Build Tool applets. If you are an application programmer and not familiar with DECtalk Software, this chapter introduces you to the basic voice and program control functions used to control DECtalk Software from within an application program. Topics include:

- Speaking a text file
- Inserting in-line commands
- Changing the speaking voice
- Changing the speaking rate
- Using the User Dictionary Build Tool
- Using the Speak applet
- Using the Text-to-Speech server from Windows applications
- Using the Say command-line applet

Speaking a Text File

The fastest way to start learning about DECtalk Software is by using a DECtalk sample application, such as the Dtsample applet (Windows only) or the Speak applet, to convert an ASCII text file to speech. After you start the applet and speak a text file, you can then learn more about controlling voice characteristics through in-line commands and the user dictionary, which are covered later in this chapter.

In this section, Dtsample is used to illustrate speaking a text file. For equivalent examples that use Speak, see *Using the Speak Applet*.

Note
The drag and drop technique described in Table 2-1 applies equally to Dtsample and Speak.

Figure 2-1 Speaking a Text File Using the Dtsample Applet

(8)
(9)
(10)

Table 2-1 Speaking a Text File

Task/Location Action Result

Using the Dtsample applet File menu

DECtalk Software program
group (1)

Step 1. Double click the Dtsample applet icon. (2)

The Dtsample applet dialog is displayed. (3)

Dtsample applet dialog (3) Step 2. Pull down the Language
menu and select the language of your choice.

Step 3. Pull down the File menu and select Open. (4)

The Open a File dialog box is displayed. (5)

Open File dialog box (5) Step 4. Select the file you want
DECtalk Software to speak.
(6)

The file is displayed in the Dtsample applet dialog.(7)
Start button (8) Step 5. Click the start button. DECtalk Software starts speaking
the file.
Pause and stop buttons (9) (10) Step 6. Click the Pause or Stop
button as needed.
DECtalk Software pauses or stops the speaking session.

Using drag and drop
Windows Explorer with text file name visible (not illustrated)
Step 1. Click the file name and drag to one of the following:
• Dialog box for the Dtsample applet or the Speak applet
• Dtsample or Speak icon
The file is spoken.
If you drag the file name to a DECtalk icon, the application it represents is opened.
2-5

Inserting In-Line Voice Control Commands

Before you can use DECtalk Software in-line commands to modify speech
output, you need to know the proper syntax with which to include those
commands into a text file. Figure 2-2 illustrates the rules of DECtalk Software
in-line command syntax. For complete details about DECtalk Software in-line
commands, see the *DECtalk Software Reference Guide*.

Figure 2-2 Rules for DECtalk Software In-Line Command Syntax

❶ `[:rate 150][:name Betty]Hello. How are you?`

❷ `[:rate 150][:nb]Hello. How are you?`

❸ `[:dv ap 160 pr 50 save][:nv]Hi.`

❹ `[:dv ap 160 pr 50 save][:nv]Hi.`

❺ `[:dv ap 160 pr 50 save][:nv]Hi.`

❻ `[:nb][:np]Hello.`

❼ `Now I'm [:dv ap 90 pr 130][r ' iyl iy]`
`thrilled.`

Legend
❶ Enclose every command within brackets.
❷ Some commands provide an alternate form to simplify input. Here the `:name` command
and its argument `Betty` can be replaced by the alternate command **:nb**.
❸ Begin every command with a colon.
❹ Separate each command name and its option or parameter from the command name text
by a valid word boundary marker. The valid word boundary markers are a space and a
tab. A space is used here.
2-6
Legend
❺ Include several options and parameters within the same brackets if the command allows
more than one option and parameter. In this example, the option and parameter grouping
modifies the **[:dv]** command.
❻ If you give two conflicting commands, DECtalk Software uses the last command in the
sequence. In this example, DECtalk Software uses Paul's voice.
❼ If you enable phoneme interpretation by using the **[:phoneme arpabet speak on]**
command, you can include phonetic spelling for text-to-speech synthesis. The phonetic
spelling replaces the actual spelling and is enclosed within brackets. In this example, the
phonetic spelling of the word really (r ' iyl iy) is included.

Additional rules for in-line command syntax include the following:

• If the value in a **[:dv]** command is too low, DECtalk Software uses the
minimum valid value. If the value is too high, it uses the maximum valid
value.

- After you enter a command, that command applies to the remaining text until it is overridden by another command. For example, the command **[:nk]** invokes Kit's voice on all entered text until you enter another voiceselection command.
- Invalid commands are ignored. By setting the **[:error speak]** command, you can receive an *audible* warning that an invalid command has been entered.
- Do not put arpabet parameters within the brackets for another command.
- DECtalk Software interprets text between brackets as phonemes only after the **[:phoneme arpabet speak on]** command is sent to the application. If **[:phoneme arpabet speak on]** has not been sent, DECtalk Software interprets the brackets and characters between them literally. The **[:phoneme arpabet speak off]** command must be sent with literal characters if you want to insert brackets in normal text.
- If the command **[:phoneme arpabet speak on]** is set and you forget the right bracket ( ]), DECtalk Software attempts to interpret all text following the ASCII text as phonemes, skipping over illegal letter combinations. The resulting text sounds garbled. Enter a right bracket to fix this problem.

2-7

Changing the Speaking Voice

You can change a speaking voice by inserting DECtalk Software commands into the edit window of a DECtalk application, such as the Dtsample applet (Windows only) or the Speak applet. Within the Dtsample applet, you can also select text and then select a different name from the Voice menu in the Dtsample applet dialog. Voice changes made with commands are permanent for the session and remain intact as long as the command remains in the file. Voice changes made from the menu remain in effect only as long as the current DECtalk Software session is running.

Each voice selection is inserted into the command **[:n*X*]**, where *X* is a value representing a DECtalk speaking voice. Table 2-2 lists the names and their corresponding values. Figure 2-3 and Table 2-3 show steps you can make to change voices.

You can change voices with a command as shown in the following example.

```
[:nb] Hello. I'm Betty.
```

You can also change voices in the middle of a sentence.

```
[:np] This is a demo [:nb] of a sudden change in voice.
```

If a voice-change request occurs in the middle of a sentence, DECtalk Software automatically pauses. The pause is the equivalent of inserting a comma, or about half a second.

```
[:np] This is a demo, [:nb] of a sudden change in voice.
```

Table 2-2 DECtalk Voices and Their Associated Values

Name	Value	Name	Value
Paul	P	Betty	B
Harry	H	Ursula	U
Frank	F	Wendy	W
Dennis	D	Rita	R
Kit	K		

2-8

Figure 2-3 Changing the Speaking Voice

(1)

(2)

(3)
(4)
Table 2-3 Changing the Speaking Voice

Task/Location Action Result
Using the Dtsample applet Voice menu
Dtsample applet
dialog (1)
Step 1. Select the desired
speaking voice from the
Voice menu. (2)
Step 2. Press the start button. (3) The entire file or selected section is spoken.

Using in-line commands
Dtsample applet
dialog (4)
Step 1. Insert in-line commands in
the text at the points
where you want a new
voice to take effect.(4)
DECtalk Software changes the speaking voice
at the point where you insert commands.
2-9

Changing the Speaking Rate

DECtalk Software lets you edit text that is displayed in the edit window of a
DECtalk application, such as the Dtsample applet (Windows only) or the Speak
applet, and then play part or all of the edited text. Editing can include selecting,
cutting, pasting, and appending other text files. Figure 2-4 and Table 2-4 show
cutting and pasting, playing selected text, and changing the speaking rate.
You can also increase and decrease the rate at which DECtalk Software speaks
a file or a section of a file.
Figure 2-4 Changing the Speaking Rate
(1)
(3)
(4)
(2)
(6)
(5)
2-10
Table 2-4 Changing the Speaking Rate

Task/Location Action Result
Editing the text
Dtsample applet dialog (1)
Step 1. Select the range of text you want to edit. (2)
Step 2. Use cut, copy, paste, and delete from the Edit menu (3)
to manipulate the selected text.
You can also insert text and DECtalk Software commands
by placing the cursor anywhere in the text and typing.
Step 3. Click on the Start button. (4) DECtalk Software speaks the edited file in the
new, edited format.

Playing selected text
Dtsample applet
dialog (1)
Step 1. Select the range of text you want to play. (2)
Step 2. Click on the right mouse
(MB2 or 3) button.
DECtalk speaks the selected text.
Changing the rate
Dtsample applet dialog as DECtalk Software is speaking a file
(1)

Step 1. Use the mouse to position the pointer on the rate slider bar.
Press the left mouse (MB1) and drag the slider to the left and
wait for the speaking voice change to occur. Then drag the
slider to the right. (5)
The speed at which DECtalk Software speaks the file changes.
It increases if you drag to the right
and decreases if you drag to the left. The rate in words per minute is displayed to the right of the
slider bar. Changes in the speaking rate occur only on a clause boundary after all the previously
queued audio has played. Dtsample applet dialog. (1)
Step 2. You can also insert in-line rate commands in the text. (6)
When the text is played, DECtalk changes the rate according to the commands.
2-11

Using the User Dictionary Build Tool

The User Dictionary Build Tool creates a loadable dictionary (.dic) file from a
list of words and their corresponding pronunciations.
The dictionary can then be loaded with a call to the

TextToSpeechLoadUserDictionary() API function or from the File menu in
the Speak applet.

Note

You can use the User Dictionary Build Tool's Translate menu to create phonemic
translations of words or phrases.
2-12

Menus and Commands

Figure 2-5 and the accompanying legend illustrate and explain the User
Dictionary Build Tool's menus and commands.
Figure 2-5 User Dictionary Build Tool
2-13

Legend
(1) Edit Window
In the edit window, enter word-pronunciation pairs that are not predefined or not pronounced
as desired in the DECtalk Software user dictionary. A word-pronunciation pair is a word,
followed by its phonemic spelling enclosed in square brackets.
(2) File menu
Open... A pop-up dialog box that opens up a dictionary definition file (*.tab).
Close Closes the dictionary definition file. If the entries in the edit window have been
modified, a dialog box asks if you wish to save changes.
Compile Dictionary Compiles the current file to a .dic file with the same name.
Save... A pop-up dialog box that saves the file and calls the compiler to create a dictionary file
(*.dic).
Save as... A pop-up dialog box that lets the user rename the file. This also calls the compiler
to create a dictionary file (*.dic).
Exit Exits the program. If the entries in the edit window have been modified, a dialog box
asks if you wish to save changes.
(3) Edit menu
Undo Undo the previous command.
Cut Cut the selected region.
Copy Copy the selected region.
Paste Paste the selected region.
Delete Delete the selected region.
Select All Select all of the word-pronunciation pairs from the edit window.
Find... A pop-up dialog box that prompts you to search for selected text.
(4) Translate menu
Translates selected text into phonemic spelling.
(5) Language menu
Lists the DECtalk languages. Uninstalled languages are grayed out.
(6) Help menu

Selects the online help.
(7) Pronounce Word button
When clicked, the selected text in the edit window is spoken. If a word-pronunciation pair is
selected, the phonemic pronunciation is used. If only the word is selected, the currently
stored pronunciation is used.
(8) Previous button
When clicked, the previous word-pronunciation pair in the list is spoken.
(9) Next button
When clicked, the next word-pronunciation pair in the list is spoken.
2-14

Building a User Dictionary

Building a user dictionary is a two-step process. First, you create a .tab source
file with the User Dictionary Build Tool to define the pronunciation of special
words you want to place in the dictionary, as shown in Figure 2-6 and Table
2-5.

Figure 2-6 Creating or Modifying a User Dictionary
Table 2-5 Creating or Modifying a User Dictionary

Task/Location Action Result
DECtalk
Software (1)
program group
Step 1. Double click on the User
Dictionary icon.
The DECtalk Software Dictionary Builder dialog
is displayed. (2)
2-15
File menu (3) Step 2. Select Open to open an
existing dictionary definition
(.tab) file.
You can also create a new
dictionary definition file by
entering words directly.
The Open a File dialog box is displayed (not
shown). The file you select is displayed in the
edit window.
Edit window Step 3. Enter words that are not predefined or pronounced as desired in the DECtalk
user dictionary. DECtalk needs to know both the word and the phonemic spelling
you want to associate with it.
If you know the phonemic spelling, enter it using the following format:
word [phonemic spelling]
If you do not know the phonemic spelling, enter the word as it sounds, rather
than as it is spelled.
Step 4. Select the word and click
Translate. (4)
For example, the Yugoslavian
name Jovicic sounds like:
Yoeveecheech
The User Dictionary Build Tool converts
what you entered into a phonemic spelling.
The User Dictionary Build Tool converts the
name to:
[yx' owv iychiych]
Step 5. To hear how DECtalk interprets
the phonemic spelling, select it,
and click Pronounce Word. (5).
DECtalk speaks the word.
Step 6. Besides pronunciation, you can also define word usage.
For example, to define the word soda to be equivalent to pop; define water to be
pronounced with a New England accent; and to take into account the dialectic
regional preferences so that the word sub is called a hero, you would use the
following pronunciation pairs:

Soda [p ' aap]
water [w ' aot rr]
sub [hx' iyr ow]
Refer to the Reference Tables and to Using In-Line Commands (available in the
Reference Guide and in online help) for more information on modifying and
enhancing pronunciation, including a complete list of phonemic, stress, and
syntactic symbols.

After you create the .tab source file, you compile the .tab file to produce a .dic
file as shown in Figure 2-7 and Table 2-6. The .dic file can be loaded into a
DECtalk Software session through the Speak applet or an API function call.

2-16

Figure 2-7 Saving and Compiling the Dictionary

2-17

Table 2-6 Saving and Compiling the Dictionary

Task/Location Action Result
Saving and compiling a new dictionary
File menu (1) Step 1. Display the File menu.
If you have not yet
given your file a name,
choose one of the
following actions:
Compile Dictionary (2)
Save (3)
Save As... (4)
The Save dialog box is displayed in all three cases. (5)
Save as dialog
box
Step 2. Navigate to the
location where you
want to save the
dictionary session and
enter a file name. (6)
A success message (7) is displayed, indicating that
both the text file (in .tab format) and a dictionary file (in
.dic format) have been saved.

Saving and compiling an existing dictionary
File menu (1) Step 1. Choose Open from the
File menu to open an
existing file.
The Open a File dialog box is displayed (not shown).
The dictionary file (.tab file type) you choose is
displayed in the edit window.
Step 2. Edit the file as
described in Table 2-5.
File menu (1) Step 3. To save the changes
you have made to the
.tab file without
compiling the
dictionary, pull down
the File menu (1) and
click Save (3).
The changes you have made are saved.
File menu (1) Step 4. To save your changes
and compile the
dictionary, pull down
the File menu (1) and
click Compile
Dictionary. (2)
A success message (7) is displayed, indicating that
both the text file (in .tab format) and a dictionary file (in
.dic format) have been saved.

2-18

Using the Speak Applet

The Speak applet demonstrates the DECtalk Software voice set, as shown in Figure 2-8 and Table 2-7. Each picture displayed in the Speak dialog represents one of the nine built-in voices. You can select a specific voice by clicking the picture. DECtalk Software then uses the selected voice to speak the contents of a text file displayed in the edit window or from a file you drag and drop into the edit window of the Speak dialog.

Notes

• The Speak applet is not provided and not supported for Windows CE systems.

• The Speak applet faces display as cartoon characters when using VGA 16-color mode.

2-19

Figure 2-8 Using the Speak Applet

2-20

Table 2-7 Using the Speak Applet

Task/Location	Action	Result
Editing a text file and playing selected segments		
DECtalk program group (1)	Step 1. Double-click the Speak icon.	The Speak applet dialog (2) displays.
File menu (3)	Step 2. Choose Open (4) to choose a text file, or enter text in the edit window.	The Open A File dialog box displays (not shown). The file you choose is displayed in the edit window
Speak applet dialog (2)	Step 3. Select the range of text you want to edit or play. (5)	
	Step 4. Use the Cut, Paste, and Insert keys or corresponding commands from the Edit menu to manipulate the selected text. For example, cut and paste a single sentence for DECtalk Software to speak.	
	Step 5. Click the face of the voice you want to speak. (6)	
Edit menu	Step 6. Use the right mouse button (MB2 or 3) to speak a selected region or press the Play, Pause, and Stop buttons (7) to play the whole file.	The edited file is spoken.
Changing the speaking rate		
Speak applet dialog while DECtalk Software is speaking a file	Step 1. Use the mouse to position the pointer on the rate slider button. (8) Press MB1 and drag the slider to the left and wait for the speaking rate to slow. Then move the slider to the right to increase the rate.	The rate in words per minute displays to the right of the slider bar. Rate changes occur only at a clause

boundary and only after all previously
queued audio has played.

Loading a User Dictionary

File menu (3) Step 1. Select Load User Dictionary. (9) The
current User Dictionary is loaded if
available. Note that the current User
Dictionary is defined in a platformspecific
manner; see Dictionary
Functions (Linux and UNIX) or
Dictionary Functions (Windows).
The Load Dictionary dialog box (not
shown here) is displayed.
Step 2. Select the User Dictionary you want to
load, and press Enter or click OK.
That dictionary is loaded and remains in
effect until you change the dictionary or
end the session.
2-21

Saving text as a .wav f ile

File menu (3) Step 1. Pull down the File menu and select
Convert to Wave File. (10) Select
the desired format. (11) Select a file
name for the audio file.
The Convert to Wave File dialog box is
displayed (not shown here). The text
file is saved to the selected file as a
standard .wav file.

Figure 2-9 shows how to highlight a selection of text for speaking.

Figure 2-9 Highlighting Spoken Text from the Speak Applet

(1)
(2)
(4)
(3)

Task/Location Action Result
Highlighting Spoken Text

Speak applet Edit
menu. **(1)**
1. Click Highlighting. **(2)** The Highlighting option is checked.
Speak applet
controls.
2. Click the Start button. **(3)** As DECtalk speaks the text, each word is
highlighted. **(4)**
2-22

Using the Text-to-Speech Server from Windows Applications

On Windows 95, 98, ME, NT, 2000, and XP systems, you can run DECtalk
Software by way of the Text-To-Speech (TTS) server from any application that
supports Dynamic Data Exchange (DDE). When run from such applications,
you can have DECtalk speak any selected text associated with the application.
For example, DECtalk can be "plugged into" Microsoft Word and then used as
a proofing tool to read entire files or selected sections from those files.
The following section is an example of how to set up and run DECtalk Software
from within a Microsoft Word document. The steps include:
1) Creating a Word macro and associating it with a template
2) Associating the macro with a button, menu selection, or hot key
3) Launching the DECtalk TTS Server and speaking the text
4) Speaking the text from within the document

Step 1 — Creating a Word Macro and Associating It with a Template

The first step in running DECtalk Software from a Microsoft Word document is creating the Word macro that links Word and DECtalk through DDE, as shown in Figure 2-10 and Table 2-8. You only have to create this macro once and associate it with a button, hot key, or menu command. If you make the macro part of a permanent .dot (template) file, DECtalk Software is always available when you edit a document associated with that template.

Note
This example uses Word Version 7.0. Using other versions of Word, the displays and the specific steps will differ somewhat, but the overall procedure will apply. Refer to the Help for your version of Word for the specific steps required to create Word macros.

2-24
Figure 2-10 Creating a Word Macro
Table 2-8 Creating a Word Macro

Task/Location Action Result
Word dialog Step 1. Select Macro from the Tools
Menu. (1)
The Macro dialog box is displayed. (2)
Macro dialog
box (2)
Step 2. Enter the name of the macro
you are going to create. (3)
Macro dialog
box (2)
Step 3. Click the Create button. (4) The Macro creation window is displayed.(5)
2-25

Task/Location Action Result
Macro creation
window (5)
Step 4. Insert the macro code. The **word.txt** file located in your DECtalk installation directory offers two versions of the appropriate macro code. For Word Version 7, insert the following code from **word.txt**:

```
Sub DECtalkTTSmacro()
Dim Response, rightNow, diffTime, chan
If Tasks.Exists("TTSsrv") = 0 Then
StatusBar = "Starting DECtalk Server"
System.Cursor = wdCursorWait
Response = Shell("TTSSRV.EXE", 0)
rightNow = Now()
diffTime = 0
While diffTime < 0.00003
diffTime = Now() - rightNow
Wend
StatusBar = " "
System.Cursor = wdCursorNormal
If Response = 0 Then
Response = MsgBox("Cannot find TTSSRV.EXE in the path",
vbYesNo, "Speak")
GoTo FINIS
End If
End If
chan = DDEInitiate(App:="TTSserver", Topic:="Speak")
If chan = 0 Then
Response = MsgBox("Unable to connect to TTSSRV.EXE", vbYesNo,
"Speak")
GoTo FINIS
Else
DDEPoke channel:=chan, Item:="Speak", Data:=Selection.Text
DDETerminate channel:=chan
End If
FINIS:
End Sub
```
File menu Step 5. On the File menu, click Close

(6). Click Yes (7) when you
are asked if you want to keep
changes to the macro.
The macro is created.
2-26

Step 2 — Associating the Word Macro with a Toolbar Button

After you create a macro to run DECtalk from within a Microsoft Word
document (step 1), you can associate the macro with a button on a toolbar as
shown in Figure 2-11 and Table 2-9. You can also associate the macro with a
menu selection from one of the pull-down menus or with a power-key
combination from the keyboard. This section illustrates how to link the macro
with a button on a toolbar.

Figure 2-11 Associating the Word Macro with a Toolbar Button
2-27
Table 2-9 Associating the Word Macro with a Toolbar Button

Task/Location Action Result
Word dialog
Tools menu
Step 1. Select Customize. The Customize dialog box is displayed. (1)
Customize dialog
box
Step 2. Select the Toolbars tab and
click on Macros (2).
Step 3. Locate the macro you
created and click the
name. (3)
Step 4. Place the cursor (arrow) on
the macro name, press and
hold MB1 and drag the
cursor to the toolbar in
which you want the
DECtalk macro
placed.（4）
Word toolbar you
select
Step 5. An outlined box appears
where Word will place the
new button.（4）
The Custom Button box is displayed.（5）
Custom button
dialog box（5）
Step 6. Click the button you want
to associate with the
DECtalk macro. (6)
The button is outlined.
Step 7. Click Assign. (7) The new button now appears in the gray outlined
box in the toolbar.
2-28

Step 3 — Launching and Configuring the DECtalk TTS Server

After you associate the DECtalk macro with a button (Step 2), you must have
the Text-to-Speech (TTS) server running before DECtalk can speak selected
sections of text from an application, as shown in Figure 2-12 and Table 2-10.
This section outlines how to start the TTS server and configure the TTS server
for speaker selection, User Dictionary selection, and speaking rate.
The TTS server is launched by the Word macro and is displayed as a small
window. You can control the speaking voice and speaking rate from the
Window menu.

Applications can access the TTS server using the service name "TTSserver." The service topic names are "Speak," "LoadDictionary," and "UnloadDictionary." Each of these topics requires a string that is either text to be spoken (for "Speak") or a dictionary file string.

Figure 2-12 Launching and Configuring the DECtalk Server

2-29

Table 2-10 Launching and Configuring the DECtalk TTS Server

Task/Location Action Result
Microsoft Word
editing session
Step 1. Click the TTS Server icon (1)
you created when you
associated your Word macro
with a toolbar button. (See Table
2-9.)
The TTS Server dialog is displayed. （2）
TTS Server
dialog (2)
Step 2. Pull down the Window menu （3）
and select Configure... （4）
The Change Settings dialog box is
displayed. （5）
Change Settings
dialog box （5）
Step 3. Click the desired speaking voice
and speaking rate. (6)
Step 4. Click on the Test button (7) to
preview your selections.
Step 5. Adjust the settings until you
produce the desired result.
Step 6. Click OK. (8) The selected settings remain in effect as
the default settings until they are changed.
(Note: In-line commands in selected text
played through the TTS server will override
the defaults.)
2-30

Step 4 — Speaking Microsoft Word Text With the TTS Server

Once the TTS server is launched by the Word macro and you configure it for the desired speaking voice, speaking rate, and with a User Dictionary file (step 3), you can have DECtalk speak selected sections of text from an opened file, as shown in Figure 2-13 and Table 2-11.

Figure 2-13 Speaking Text in a Word File With the TTS Server

2-31

Table 2-11 Speaking Microsoft Word Text With the TTS Server

Task/Location Action Result
Microsoft Word
dialog with file
opened (1)
and the TTS
Server macro
button visible
（2）
Step 1. Select a range of text for
DECtalk to speak. （3）
Step 2. Click the TTS Server button
（2）
DECtalk speaks the selected text and displays
the TTS Server dialog. (4)
TTS Server

dialog `( 4 )`
Step 3. Click the Pause or Stop
button if you want to pause or
stop DECtalk as it is
speaking the text.
Note: When you pause the
server, all queued text
remains queued up even
though you deselect it in the
Word document. When you
stop the server, all queued
text is flushed from the
DECtalk system.
If the server is not accessible, an error is
returned. `( 5 )`

Using the Say Command-Line Applet

The DECtalk Say command-line applet lets you run DECtalk Software from the
operating system command prompt window. Say provides many of the standard
DECtalk input and output options available in the Speak and Dtsample applets
and the DECtalk API. The command syntax is as follows:

`say [options] [text]`

Notes

- The Say applet is not provided and not supported for Windows CE systems.
- On UNIX systems, the `say` command must be entered in lower case letters.

Help Options

-h or -? Write this file to the console. This option cancels any others on the
command line.

Output Options

-w outFile Convert the text to the specified .wav file instead of speaking to
the sound device.
-l[t] outFile Turn on text logging, which logs all input text to a file. This text
includes any pre- and post- commands as well as commands sent
to DECtalk by the Say program itself.
Because this is the default logging mode, the 't' immediately
following the '-l' is optional.
-ls outFile Turn on syllable logging, which logs each syllable to a file.
-lp outFile Turn on phoneme logging, which converts the input text to
phonemes. This is useful if you want to get DECtalk Software to
sing. You convert the text to phonemes and then insert the tone
commands into the phoneme file.
If no output options are specified, Say sends its output to the
installed sound device, usually a sound card. Only one output
option can be specified; if you specify more than one, the last one
on the command line is used.

Input Options

-pre preText Text to be passed to DECtalk Software before the normal input.
The prefix text is forced out before the input text is read.

-post postText Text to be passed to DECtalk Software after the normal input.
This is useful for passing terminating commands to DECtalk
Software that normally are not part of the input. If the postfix text
has spaces, it must be enclosed in quotes. An example is
"**[:phoneme off]**" or "The End".
The normal input is forced out before the postfix text is read.
text Text appearing on the command line is spoken. The text to be
spoken can come either from standard input or from the command

line.

Anything on the command line that is not an option is interpreted as text, as is anything following it on the command line. In other words, text to be spoken must appear on the command line after all options.

If the first word in the text has a dash (-) or slash (/) as its first character, you must precede it with another dash or slash. For example, to tell DECtalk to say the number -123, you would type the command:

```
say --123
```

This is necessary to avoid having Say interpret the number as a command line option. If you embed DECtalk commands into your text, you must enclose them in quotes if they contain spaces. This is necessary because Say treats each space-delimited command-line argument as a separate word, while DECtalk commands must be processed as single words by the Say program.

If no text is specified, Say takes its input from the standard input. For example, you could have Say speak a directory listing in Betty's voice by typing the following:

```
dir | say -pre "[:nb]"
```

Alternatively, you could just type the following command

```
say
```

Then enter text at the console. In this case, Say speaks each line after you press Enter, and exits after you press Ctrl/x. If you want Say to take its input from a file, use file redirection as in the following example, which reads the file `foo.txt` in Harry's voice:

```
say -pre "[:nh]" < foo.txt
```

Dictionary Options

-d userDict Loads the specified User Dictionary before speaking. This dictionary is loaded in place of any default User Dictionary determined by DECtalk.

3-1

Chapter 3 — Introduction to the DECtalk Software API

This chapter introduces the DECtalk Software API (DAPI) and explains how to use the API to write application programs. This API is designed to be extensible for future text-to-speech growth while still being easy to use. The current DECtalk Software implementation supports multiple instances of DECtalk Software per processor.

The DECtalk Software API functions give you a flexible method for manipulating the various parameters of DECtalk Software functionality from within your application. These functions perform a wide range of tasks associated with the text-to-speech system and are listed by function in Table 3-1.

Note (Windows Only)
In addition to the DAPI, DECtalk Software supports the Microsoft Speech API (SAPI) for Windows 95/98/ME/NT/2000/XP systems and SAPI Version 5.0 for Windows 98/ME/NT/2000/XP systems. This allows DECtalk Software to work as an OLE server in any OLE application environment. Information about SAPI is available in the Microsoft Developer Network (MSDN) CD-ROM under the Microsoft Software Development Kit (SDK). See Chapter 4 in the DECtalk Software Reference Guide for a list of SAPI functions supported by DECtalk Software. For more information or documentation on SAPI, contact Microsoft.

3-2

Table 3-1 Text-To-Speech Functions by Category

Functions Purpose
Core API
TextToSpeechStartup() Initializes and starts up text-to-speech system.
• In Windows, the user defines the
window handle to which post
messages can be sent.
• In Linux and Tru64 UNIX, the user
defines the callback routine.
The calling application can then receive index
marks, memory buffers, or status information
through the above operating system dependent
mechanism.
TextToSpeechStartupEx() Initializes and starts the text-to-speech system.
The user defines the callback routine that can
be called when index marks, memory buffers,
or status information needs to be sent to the
calling application.
TextToSpeechSpeak() Speaks text from a buffer.
TextToSpeechShutdown() Shuts down the text-to-speech system.

Audio Output Control
TextToSpeechPause() Pauses output.
TextToSpeechResume() Resumes output.
TextToSpeechReset() Purges the TTS syatem and stops output.

Blocking Synchronization
TextToSpeechSync() Synchronizes to the text stream.

Control and Status
TextToSpeechSetSpeaker() Selects one of nine speaking voices.
TextToSpeechGetSpeaker() Returns the last speaking voice to have

spoken.
TextToSpeechSetRate() Sets the speaking rate of the text-to-speech
system.
TextToSpeechGetRate() Returns the speaking rate of the text-to-speech
system.
TextToSpeechSetLanguage() Sets the language to be used. (not supported)
TextToSpeechGetLanguage() Returns the language in use. (not supported)
TextToSpeechGetStatus() Gets the status of the text-to-speech system.
3-3

Functions Purpose
TextToSpeechGetCaps() Retrieves the capabilities of the text-to-speech
system.

Special Text-To-Speech Mode
TextToSpeechOpenWaveOutFile() Opens a file for output. **TextToSpeechSpeak()**
writes audio data in wave format to this file.
TextToSpeechCloseWaveOutFile() Closes the specified wave file and returns the
text-to-speech system to its startup state.
TextToSpeechOpenLogFile() Opens a log file. **TextToSpeechSpeak()** writes
text, phonemes, or syllables to this file.
TextToSpeechCloseLogFile() Closes the specified log file and returns the
text-to-speech system to its startup state.
TextToSpeechOpenInMemory() Produces buffered speech samples in wave
format whenever **TextToSpeechSpeak()**
function is called. The calling application is
notified when memory buffer is filled.
TextToSpeechCloseInMemory() Returns the text-to-speech system to its
startup state.
TextToSpeechAddBuffer() Adds a shared-memory buffer allocated by the
calling application to the memory buffer list.
TextToSpeechReturnBuffer() Returns the current shared-memory buffer.

Loading and Unloading a User Dictionary
TextToSpeechLoadUserDictionary() Loads a user dictionary.
TextToSpeechUnloadUserDictionary() Unloads a user dictionary.
3-4

The Core API Functions
The core DECtalk Software API functions are:
- **TextToSpeechStartup()** and **TextToSpeechStartupEx(),** known as the
startup functions**,** allocate system resources.
- **TextToSpeechStartup()**, which is operating-system dependent, has
different calling parameters. On Linux and UNIX, a callback routine may
be defined; on Windows, a window handle can be passed to the
TextToSpeechStartup() call.
- **TextToSpeechStartupEx(),** which works the same way on all platforms,
allows a callback routine to be passed.
- **TextToSpeechSpeak()** queues text to the system.
- **TextToSpeechShutdown()** returns all system resources allocated by the
startup functions.
The simplest application might use only these functions.

TextToSpeechSpeak
The **TextToSpeechSpeak()** function is used to pass a null terminated string of
characters to the text-to-speech system. The system queues all characters up to
the null character. If the TTS_FORCE argument is not used in the call to this

function, then the queued characters are seamlessly concatenated with previously queued characters. The TTS_FORCE argument is used to force a string of characters to be spoken even though the string might not complete a clause. For example:

```
TextToSpeechSpeak( phTTS, "This will be spoken. ", TTS_NORMAL);
```

This text is spoken immediately by the system because it is terminated by a period and a space. These last two characters are one way to create a clause boundary.

```
TextToSpeechSpeak( phTTS, "This will be spok", TTS_NORMAL );
```

This produces output only after the following line of code executes to complete the phrase.

```
TextToSpeechSpeak( phTTS, "en. ", TTS_NORMAL );
```

3-5

Finally, a nonphrase string can be forced to be spoken by using the TTS_FORCE argument.

```
TextToSpeechSpeak( phTTS, "This will be spok", TTS_FORCE );
```

Note that the word *spoken* is not pronounced correctly in this case even if the final characters, *en* are queued immediately afterward.

The TTS_FORCE argument causes the previous line to be spoken before taking any subsequently queued characters into account.

Important Text-Queuing Information

It is important that all sentences are separated with a space, new line, or line feed character. To make sure of this, it is recommended that a space character is routinely included after the final punctuation in a sentence. An example of what happens without this is shown below:

```
TextToSpeechSpeak( phTTS, "They are tired.", TTS_NORMAL );
TextToSpeechSpeak( phTTS, "I am Cold.", TTS_NORMAL );
```

Because there is no space, the text-to-speech system processes the following string:

```
"They are tired.I am Cold."
```

The string, *tired.I*, will be pronounced incorrectly because the system treats it as one item instead of two words.

Clause-Based Synthesis

DECtalk Software processes all text on a clause basis. A clause is defined as a group of words terminated by a period, question mark, exclamation point, question mark, semicolon, or a colon and followed by white space. If you send DECtalk Software a string of text with NO terminator, it goes into an infinite loop waiting for a clause terminator. The **Sync** in-line command can be used to complete a clause without using a terminator.

3-6

Callback Routines and Window Procedures

After **TextToSpeechSpeak()** is called, the text-to-speech system can notify the calling application in one of two ways. On Tru64 UNIX, the information can be sent back through the callback routine defined in the startup function. On Windows, the information is passed back through window messages or callback routines, depending on which startup function was called. See the *DECtalk Software Reference Guide* for more information on startup functions.

Callback routines should not call **TextToSpeech…()** functions. If a callback routine does call **TextToSpeech…()** functions, a crash may occur in the application calling DECtalk Software.

Both the window procedures and the callback routines have a message type and the WPARAM and the LPARM parameters. There are three message types

defined: one for error and status messages, one for index marks, and one to return memory buffers when using the **TextToSpeechOpenInMemory()** function. The WPARAM and the LPARAM parameters contain specific information, based on the message type.

Phoneme Notifications

Phoneme notifications are sent using the standard notification method (callback or message). The message identifier is the DECtalk_Visual_Message registered window message. The data portions of the message contain a structure that indicates the current phoneme, the next phoneme (not yet implemented), and the duration of the current phoneme.

The callback (and message) parameters are as follows:

- `uiMsg` DECtalk_Visual_Message
- `lParam1` A DWORD that maps the dwData part of the PHONEME_TAG structure
- `lParam2` The time, in system milliseconds (timeGetTime()),

when the phoneme started

3-7

The union PHONEME_TAG (from `ttsapi.h`) defines the data format for the phoneme part of the message. The following is included for reference, but the definitions always should be taken from the `ttsapi.h` file.

```
typedef struct {
UCHAR cThisPhoneme; // current phoneme
UCHAR cNextPhoneme; // next phoneme, if known
WORD wDuration; // duration in milliseconds
} PHONEME_MARK;
typedef union {\
PHONEME_MARK pmData;
DWORD dwData
} PHONEME_TAG
```

The cThisPhoneme and cNextPhoneme fields are ASCII printable singlecharacter phoneme identifiers. The phoneme identifiers are specific to each language. See the *DECtalk Software Reference Guide* for the phoneme identifiers for cThisPhoneme and cNextPhoneme.

An example callback routine to show phoneme notification is as follows:

```
VOID TTSCallbackRoutine (LONG lParam1,
LONG lParam2,
DWORD dwInstanceParam,
UINT uiMsg)
{
PHONEME_TAG ptPhoneme; // place to put the phoneme data..
ptPhoneme.dwData = lParam2;
fprintf(fpLogfil,"{%ld} ",timeGetTime());
if (uiMsg == uiID_Index_Msg)
{
fprintf(fpLogfil,
" [Index] p1=%081x p2=%081x i=%081x",
lParam1, lParam2, dwInstanceParam);
// watch for index marks..
if (lParam2 == 1)
uiSystemState = TEXT_STARTED;
if (lParam2 == 2)
uiSystemState = TEXT_DONE;
}
else if (uiMsg == uiID_Error_Msg)
{
fprintf(fpLogfil,
" [Error] p1=%081x p2=%081x i=%081x",
lParam1, lParam2, dwInstanceParam);
}
else if (uiMsg == uiID_Buffer_Msg)
{
```

```
fprintf(fpLogfil,
" [Buffer] p1=%081x p2=%081x i=%081x",
lParam1, lParam2, dwInstanceParam);
}
```
3-8
```
else if (uiMsg == uiID_Visual_Msg)
char szThisPhoneme[10]="";
char szNextPhoneme[10]="";
fprintf(fpLogfil,
" [Visual] p1=%081x p2=%081x i=%081x",
lParam1, lParam2, dwInstanceParam);
// decode it..
if (ptPhoneme.pmData.cThisPhoneme =='\0')
{
// null
strcpy (szThisPhoneme,"<null>");
}
else
{
szThisPhoneme[0]=ptPhoneme.pmData.cThisPhoneme;
szThisPhoneme[1]='\0';
}
if (ptPhoneme.pmData.cNextPhoneme =='\0')
{
// null
strcpy (szNextPhoneme,"<null>");
}
else
{
szNextPhoneme[0]=ptPhoneme.pmData.cThisPhoneme;
szNextPhoneme[1]='\0';
}
fprintf(fpLogfil,
" time: %ld this:%s next:%s expected at %ld",
lParam1,
szThisPhoneme,
szNextPhoneme,
timeGetTime()+ptPhoneme.pmData.wDuration);
}
else
{
fprintf(fpLogfil,
" [??] msg=%081x, p1=%081x, p2=%081x, i=%081x",
(DWORD)uiMsg, lParam1, lParam2,
dwInstanceParam);
}
fprintf(fpLogfil, "\n");
}
```
3-9

Error Messages

The message type for error and status messages is defined as follows:

• For Callback Routines:
```
uiID_Error_Message = TTS_MSG_STATUS;
```
• For Window Messages:
```
uiID_Error_Message =
RegisterWindowMessage("DECtalkErrorMessage");
```
One of the error codes listed below, defined in the ttsapi.h file, is contained in the WPARAM parameter. The LPARAM parameter contains a value of type MMRESULT. The values can be found in the ttsapi.h file.

Error Code Values
```
#define ERROR_IN_AUDIO_WRITE
#define ERROR_OPENING_WAVE_OUTPUT_FILE
#define ERROR_GETTING_DEVICE_CAPABILITIES
#define ERROR_READING_DICTIONARY
#define ERROR_WRITING_FILE
#define ERROR_ALLOCATING_INDEX_MARK_MEMORY
```

```
#define ERROR_OPENING_WAVE_FILE
#define ERROR_BAD_WAVE_FILE_FORMAT
#define ERROR_UNSUPPORTED_WAVE_FILE_FORMAT
#define ERROR_UNSUPPORTED_WAVE_AUDIO_FORMAT
#define ERROR_READING_WAVE_FILE
#define TTS_AUDIO_START
#define TTS_AUDIO_STOP
```

Index Mark Messages

The message type for index marks is defined as follows:

• For Callback Routines:

```
uiID_Index_Message = TTS_MSG_INDEX_MARK;
```

• For Window Messages:

```
uiID_Index_Message =
RegisterWindowMessage("DECtalkIndexMessage");
```

The LPARAM parameter contains the index mark value. Note that the index mark information can also be returned in the buffer message. This happens when the text-to-speech system is in the speech-to-memory mode as a result of the **TextToSpeechOpenInMemory()** call. See *Special Text-To-Speech Modes* for more information on this topic..

Buffer Messages

The message type for buffered speech samples is defined as follows:

• For Callback Routines:

```
uiID_Buffer_Message = TTS_MSG_BUFFER;
```

• For Window Messages:

```
uiID_Buffer_Message =
RegisterWindowMessage("DECtalkBufferMessage");
```

A pointer to the returned memory buffer is contained in the LPARAM parameter. Additional information about index marks and phonemes may also be returned here. See *Return of Memory Buffer* for additional details on what information gets returned.

Callback Routine Example

```
VOID main()
{
LPTTS_HANDLE_TAG phTTS;
TextToSpeechStartupEx(&phTTS, WAVE_MAPPER,REPORT_OPEN_ERROR,
Callback, 0);
}
VOID Callback(LONG lParam1, LONG lParam2, DWORD dwCallbackParameter,
UINT uiMsg)
{
if (uiMsg == TTS_MSG_STATUS)
{
// lParam1 contains error code
// lParam2 contains value of MMRESULT
}
3-11
else if (uiMsg == TTS_MSG_INDEX_MARK)
{
// lParam2 contains index mark
}
else if (uiMsg == TTS_MSG_BUFFER)
{
// lParam2 contains Pointer to buffer
}
}
```

Window Procedure Example

```
static UINT uiID_Error_Message = 0;
static UNIT uiID_Buffer_Message = 0;
static UNIT uiID_Index_Message = 0;
// Window Procedure
VOID MyWndProc(LONG lParam1, LONG lParam2, DWORD dwCallbackParameter,
```

```
UINT uiMsg)
{
if (uiMsg == uiID_Error_Message)
{
// lParam1 contains error code
// lParam2 contains value of MMRESULT
}
else if (uiMsg == uiID_Index_Message)
{
// lParam2 contains index mark
}
else if (uiMsg == uiID_Buffer_Message)
{
// lParam2 contains Pointer to buffer
}
}
VOID main()
{
LPTTS_HANDLE_TAG phTTS;
```
3-12
```
TextToSpeechStartupEx(&phTTS, WAVE_MAPPER, REPORT_OPEN_ERROR,
MyWndProc, 0);
// Registering messages to report DECtalk asynchronous events
uiID_Error_Message = RegisterWindowMessage("DECtalkErrorMessage");
// Callback message is registered indicating index marks
uiID_Index_Message = RegisterWindowMessage("DECtalkIndexMessage");
// Callback message is registered indicating DECtalk has filled up
an audio buffer
uiID_Buffer_Message = RegisterWindowMessage("DECtalkBufferMessage");
}
```
3-13

Audio Output Control Functions

The audio output control functions are used to:

- Pause the audio output
- Resume output after pausing
- Reset the text-to-speech system

An application can control speech output using **TextToSpeechPause(),**
TextToSpeechResume(), and **TextToSpeechReset().** A reset discards all
queued text and stops and discards all queued audio. If the application calls
TextToSpeechOpenInMemory() to store speech samples in memory, a reset
causes all buffers to be returned to the application.

3-14

Blocking Synchronization Function

The **TextToSpeechSync()** function blocks execution of the application until all
text previously queued by the **TextToSpeechSpeak()** function is spoken. After
the blocking synchronization function is called, there is no way to abort until all
text is processed. This could take hours if there is a great deal of text queued.
Non-blocking synchronization can be provided using the **Index Mark** in-line
command.

3-15

Control and Status Functions

The control and status functions described in Table 3-2 provide additional

information for the text-to-speech system.

Table 3-2 API Control and Status Functions

Function Description

TextToSpeechSetSpeaker() Sets the speaker's voice, which becomes active at the next clause boundary.

TextToSpeechGetSpeaker() Returns the value of the last speaker to have spoken.

TextToSpeechSetRate() Sets the speaking rate, which becomes active at the next clause boundary.

TextToSpeechGetRate() Gets the speaking rate.

TextToSpeechSetLanguage() Sets the text-to-speech system language. Refer to the `ttsapi.h` file for a list of valid languages, e.g. TTS_AMERICAN_ENGLISH. (not supported)

TextToSpeechGetLanguage() Returns the current text-to-speech system language. (not supported)

TextToSpeechGetStatus() Returns various text-to-speech system parameters, such as the number of characters in the text pipe, the ID of the wave output device, and a Boolean value that indicates whether the text-to-speech system is currently speaking or silent.

TextToSpeechGetCaps() Returns the capabilities of the text-to-speech system, which includes the version number of the system, the number of speakers, the maximum and minimum speaking rate, and the supported languages.

3-16

Special Text-To-Speech Modes

After the startup function is called by an application, the application then can call **TextToSpeechSpeak()** to speak text. This converts text-to-speech and sends the speech sample to an audio device, depending on the setting of the **dwFlags** field in the startup routine.

The DECtalk Software API also provides alternatives for the speech samples by allowing the user to select one of the special text-to-speech modes. The special text-to-speech mode functions allow the speech samples to be written to a Wave file; converted to text, phonemes, or syllables and stored in a log file; or saved in memory buffers to be passed back to the calling application. Each modeswitch function has a corresponding function to return the text-to-speech system to the startup state. These functions are listed in Table 3-3.

Table 3-3 Special Text-To-Speech Modes

Open Function Mode Close Function

TextToSpeechOpenWaveOutFile() wave-file TextToSpeechCloseWaveOutFile()

TextToSpeechOpenLogFile() log-file TextToSpeechCloseLogFile()

TextToSpeechOpenInMemory() speech-to-memory TextToSpeechCloseInMemory()

The text-to-speech system must be in the startup state before calling any of the open functions listed in Table 3-3. The corresponding close functions return the system to the startup state.

Wave-File Mode

After calling the startup function, an application can call **TextToSpeechOpenWaveOutFile().** This call blocks until all previously queued text is processed. After the call returns, all text subsequently queued by the **TextToSpeechSpeak()** function is converted to speech samples and written into a wave file. The **TextToSpeechCloseWaveOutFile()** call blocks until the speech from all previously queued text is written to the file.

Log-File Mode

After calling the startup function, an application can call the **TextToSpeechOpenLogFile()** function. This call blocks until all previously queued text is processed. After the call returns, all text subsequently queued by the startup function is written to a log file as text, phonemes, or syllables. The
3-17
phonemes and syllables are written using the arpabet phoneme alphabet. The **TextToSpeechCloseLogFile()** function terminates phoneme logging and blocks until the speech from all previously queued text is processed.

Speech-To-Memory Mode
After calling the startup function, an application can call the **TextToSpeechOpenInMemory()** function. This call blocks until all previously queued text is processed. After the call returns, all text subsequently queued by the **TextToSpeechSpeak()** function is converted to speech and stored in the memory buffers supplied by the **TextToSpeechAddBuffer()** call. The **TextToSpeechCloseInMemory()** function blocks until the speech from all previously queued text is processed.
When a memory buffer is completed, the buffer is returned to the calling application. See *Callback Routines and Window Procedures* for more information about passing data back to the calling application.

Initialization of Memory Buffers
A memory buffer is a TTS_BUFFER_T structure. This structure and the elements of its **lpData**, **lpPhonemeArray**, and **lpIndexArray** members must be *allocated*. (Note that these last two pointers can be set to NULL optionally if they are not used by the application.)
• The **lpData** element points to a byte array. The **dwMaximumBufferLength** element must be set to the length of this array.
• If the **lpPhonemeArray** element is set to NULL, no phonemes are returned. Otherwise, the **lpPhonemeArray** element must point to an applicationallocated array of structures of type TTS_PHONEME_T. The length of this array must be copied into the **dwMaximumNumberOfPhonemeChanges** element.
• If the **lpIndexArray** element is set to NULL, no index marks are returned. Otherwise, the **lpIndexArray** element must point to an applicationallocated array of structures of type TTS_INDEX_T. The length of this array must be copied into the **dwMaximumNumberOfIndexChanges** element.
3-18
TTS_BUFFER_T Structure (ttsapi.h)
This structure is allocated by the calling application and passed to the text-tospeech system through the **TextToSpeechAddBuffer()** function.
```
typedef struct TTS_BUFFER_TAG
{
LPSTR lpData;
LPTTS_PHONEME_T lpPhonemeArray;
LPTTS_INDEX_T lpIndexArray;
DWORD dwMaximumBufferLength;
DWORD dwMaximumNumberOfPhonemeChanges;
DWORD dwMaximumNumberOfIndexMarks;
DWORD dwBufferLength;
DWORD dwNumberOfPhonemeChanges;
DWORD dwNumberOfIndexMarks;
DWORD dwReserved;
} TTS_BUFFER_T;
typedef TTS_BUFFER_T * LPTTS_BUFFER_T;
```
TTS_PHONEME_T Structure (ttsapi.h)

This structure is used to store phoneme, stress, and syntactic codes of the speech sample. Refer to *Phonemic Symbols* and to *Stress and Syntactic Symbols* in the *DECtalk Software Reference Guide* for a detailed list of these symbols.

```
typedef struct TTS_PHONEME_TAG
{
// Phoneme, Stress or Syntactic symbols
DWORD dwPhoneme;
// Indicates which sample in the memory buffer
// corresponds to the phoneme symbol
DWORD dwPhonemeSampleNumber;
// Duration of phoneme symbol in milliseconds
DWORD dwPhonemeDuration
DWORD dwReserved;
} TTS_PHONEME_T;
typedef TTS_PHONEME_T * LPTTS_PHONEME_T;
```

TTS_INDEX_T Structure (ttsapi.h)

This structure is used to store index marks defined by the **Index Mark** in-line command. Refer to the Index Mark in-line command in the *DECtalk Software Reference Guide* for information on the syntax and uses of index marks.

```
typedef struct TTS_INDEX_TAG
{
```

3-19

```
// Index Mark value
DWORD dwIndexValue;
// Indicates which sample in the memory buffer
// corresponds to the index mark value
DWORD dwIndexSampleNumber;
DWORD dwReserved;
} TTS_INDEX_T;
typedef TTS_INDEX_T * LPTTS_INDEX_T;
```

TTS_CAPS_T Structure (ttsapi.h)

This structure is used by the **TextToSpeechGetCaps()** function to store language and proper name support, the sample rate, the minimum and maximum speaking rates, the number of predefined speaking voices, the character-set supported, and the version number.

```
typedef struct TTS_CAPS_TAG
{
DWORD dwNumberOfLanguages;
LPLANGUAGE_PARAMS_T lpLanguageParamsArray;
DWORD dwSampleRate;
DWORD dwMinimumSpeakingRate;
DWORD dwMaximumSpeakingRate;
DWORD dwNumberOfPredefinedSpeakers;
DWORD dwCharacterSet;
DWORD Version;
} TTS_CAPS_T;
typedef TTS_CAPS_T * LPTTS_CAPS_T;
```

The **lpLanguageParamsArray** element is a pointer to an array of structures of type LANGUAGE_PARAMS_T. The **dwNumberOfLanguages** element contains the number of elements in this array. The **dwLanguage** element of each structure in this array equals one of the supported languages. The **dwLanguageAttributes** element of each structure can contain the following constant, defined in include file ttsapi.h:

PROPER_NAME_PRONUNCIATION

Return of Memory Buffers

When the memory buffer is completed, it is returned to the calling application. A memory buffer is considered to be completed when any one of the following occurs:

• The memory buffer, which is pointed to by the **lpData** field, is filled.

3-20

• The phoneme array is filled. Refer to the *DECtalk Software Reference Guide* for more information on the phoneme codes.

- The index mark array is filled. Refer to *DECtalk Software Reference Guide* for more information on the index marks.
- The TTS_FORCE argument is used in the call to **TextToSpeechSpeak().** The application must not modify any buffer passed to the text-to-speech system by the **TextToSpeechAddBuffer()** function until the buffer is returned from the text-to-speech system to the calling application. The application then owns the buffer. If no buffers are available, the system blocks. If the application is processing relatively long passages of text, the application should queue several buffers and then requeue each buffer after finishing with that buffer, so that the system is not idle.

A call to **TextToSpeechReset()** returns all buffers to the application. The **TextToSpeechReturnBuffer()** function forces the return of the current memory buffer, whether it is filled or not. This function might not be required by most applications. It is included so an application can obtain the last buffer without forcing that buffer to be sent with the TTS_FORCE argument in the **TextToSpeechSpeak()** function.

When the memory buffer, a TTS_BUFFER_T structure, is returned to the calling application, it contains the following return values:

Parameter Value

dwBufferLength Number of bytes of audio samples.

lpData Pointer to the audio sample data.

dwNumberOfPhonemeChanges Number of phoneme changes.

lpPhonemeArray Pointer to the phoneme information.

dwNumberOfIndexMarks Number of index marks.

lpIndexArray Pointer to the index mark information.

The index and phoneme arrays each contain a time stamp in the form of a sample number. This sample number is initialized at zero at startup and after each call to **TextToSpeechReset().** The phoneme array also contains the current phoneme duration in frames. Each frame is approximately 6.4 milliseconds.

3-21

Dictionary Functions (Linux)

DECtalk Software comes with a main dictionary, which is a compiled list of words and their associated phonemic interpretation. This main dictionary is loaded during the startup function. In addition to a main dictionary, users can create their own user dictionaries and access them using the **TextToSpeechLoadUserDictionary()** and **TextToSpeechUnloadUserDictionary**() functions.

Creating a User Dictionary

DECtalk Software includes two dictionary applets. Both applets help users to create their own user dictionaries, which can be loaded at startup time or while the text-to-speech system is active.

Applet Location Comments

userdict /usr/local/bin/userdict
(Linux)
/usr/ucb/userdict (UNIX)
Compiles a user dictionary from a file containing
both the word and its phonemic pronunciation.
windict /usr/local/X11/bin/windict
(Linux)
Provides a graphical user interface for:
• Pronouncing words
• Translating words into their phonemic
symbols
• Compiling a user dictionary from a list of
words and their phonemic symbols

Loading the Main Dictionary

The startup function loads the DECtalk Software main pronunciation dictionary:

• `/usr/local/lib/DECtalk/dtalk_ langcode.dic` (Linux)

Replace *langcode* with the designation for the appropriate language, such as `us` for United States English, `uk` for United Kingdom English, `sp` for Castilian Spanish, `la` for Latin American Spanish, `gr` for German, or `fr` for French; for example, `dtalk_us.dic`.

On Linux systems, the main dictionary pathnames are defined in the file `/etc/DECtalk.conf` at label `LANGCODE_dict:` (for example, at label `US_dict:` for US English).

3-22

If the dictionary file cannot be found or is loaded improperly, then the startup function returns an error.

Note

DECtalk Software also provides variants and supplements to the default main pronunciation dictionaries, as follows:

• `dtalk_fl_gr.dic` is a supplemental German foreign-language dictionary, which allows you to include foreign phrases in German speech

• `dtalk_grs.dic` is a smaller variant of the German main dictionary, which provides a subset of the content of `dtalk_gr.dic`

• `dtalk_uks.dic` is a smaller variant of the U.K. English main dictionary, which provides a subset of the content of `dtalk_gr.dic`

• `dtalk_usm.dic` is a medium-sized variant of the U.S. English main dictionary, which provides a medium-sized subset of the content of `dtalk_us.dic`

• `dtalk_uss.dic` is a smaller variant of the U.S. English main dictionary, which provides a small subset of the content of `dtalk_us.dic`

Loading the User Dictionary

The startup function attempts to load the DECtalk Software user pronunciation dictionary from the user's home directory:

- $HOME/udict_ langcode.dic (Linux)

On Linux systems, replace *langcode* with the designation for the appropriate language, such as `us` for United States English, `uk` for United Kingdom English, `sp` for Castilian Spanish, `la` for Latin American Spanish, `gr` for German, or `fr` for French; for example, `udict_us.dic`.

On Linux systems, the user dictionary names are defined in the file `/etc/DECtalk.conf` at label `LANGCODE_udict:` (for example, at label `US_udict:` for US English).

If the dictionary file is found but cannot be loaded, the startup function returns an error.

After the startup function has completed successfully, the user can load and unload user dictionaries by using the **TextToSpeechLoadUserDictionary()** and **TextToSpeechUnloadUserDictionary()** functions.

3-23

Dictionary Functions (Windows)

DECtalk Software comes with a main dictionary, which is a compiled list of words and their associated phonemic interpretation. This main dictionary is loaded during the startup function call. In addition to a main dictionary, users can create their own user dictionaries and access them using the **TextToSpeechLoadUserDictionary()** and **TextToSpeechUnloadUserDictionary()** functions.

Creating a User Dictionary

DECtalk Software includes the following files, which help users to create their own user dictionaries. A user dictionary can be loaded at startup time or while the text-to-speech system is active.

File Name Location Comments

windic.exe \Program Files\DECtalk Application that has a graphical user interface and can:
- Pronounce words
- Translate words into their phonemic symbols
- Compile a user dictionary from a list of words and their phonemic symbols

user.tab \Program Files\DECtalk Source file used by windic.exe to create the user dictionary.

Loading the Main Dictionary (Dynamic or Static Engine)

The **TextToSpeechStartup()** function attempts to find the entry for the DECtalk Software main pronunciation dictionary in the registry at:
```
REGISTRY KEY:
HKEY_LOCAL_MACHINE\SOFTWARE\DECtalk Software\DECtalk\ version\langcode
Value: MainDict
```
The dictionary is then set to:
```
[user selected dir]\system\dtalk_ langcode.dic
```
Replace `version` in the path designation with the current version number for the DECtalk Software, using the format x.xx; for example, 4.62.

3-24

Replace `langcode` with the designation for the appropriate language, such as `US` for United States English, `UK` for United Kingdom English, `SP` for Castilian Spanish, `LA` for Latin American Spanish, `GR` for German, or `FR` for French; for example, `dtalk_us.dic`.

If the MainDict entry, as specified in the registry entry, cannot be found, the **TextToSpeechStartup()** call returns a value of MMSYSERR_ERROR. If the registry entry is missing, then the **TextToSpeechStartup()** function defaults to looking in the application's default directory for `DTALK_ langcode.dic` to be loaded as the main pronunciation dictionary.

If the main pronunciation dictionary fails to be loaded, the **TextToSpeechStartup()** function returns a value of MMSYSERR_ERROR.

Note

DECtalk Software also provides variants and supplements to the default main pronunciation dictionaries, as follows:

• `dtalk_fl_gr.dic` is a supplemental German foreign-language dictionary, which allows you to include foreign phrases in German speech

• `dtalk_grs.dic` is a smaller variant of the German main dictionary, which provides a subset of the content of `dtalk_gr.dic`

• `dtalk_uks.dic` is a smaller variant of the U.K. English main dictionary, which provides a subset of the content of `dtalk_gr.dic`

• `dtalk_usm.dic` is a medium-sized variant of the U.S. English main dictionary, which provides a medium-sized subset of the content of `dtalk_us.dic`

• `dtalk_uss.dic` is a smaller variant of the U.S. English main dictionary, which provides a small subset of the content of `dtalk_us.dic`

Loading the Main Dictionary (Static Engine)

The **TextToSpeechStartup()** function attempts to find the dictionary in the home directory of the statically linked application. If that fails, the **TextToSpeechStartup()** function then looks in the defined PATH of the Windows operating system. The file name in either case is `DTALK_ langcode.dic.` If the PATH look up fails, the value from the registry entry for the static engine is used. The registry entry for the static engine is:

```
REGISTRY KEY:
HKEY_LOCAL_MACHINE\SOFTWARE\DECtalk Software\DECtalk\ langcode
3-25
Value: MainDict
```

If the main pronunciation dictionary fails to be loaded, the **TextToSpeechStartup()** function returns a value of MMSYSERR_ERROR.

Loading the User Dictionary

The **TextToSpeechStartup()** function attempts to find the entry for the DECtalk Software user pronunciation dictionary in the registry at:

```
REGISTRY KEY:
HKEY_CURRENT_USER\Software\DECtalk Software\DECtalk\ version\langcode
Value: UserDict
```

The dictionary is then set to:

```
[user selected dir]\user.dic
```

Replace `version` in the path designation with the current version number for the DECtalk Software, in the format x.xx; for example, 5.00.

Replace `langcode` with the designation for the appropriate language, such as `US` for United States English, `UK` for United Kingdom English, `SP` for Castilian Spanish, `LA` for Latin American Spanish, `GR` for German, or `FR` for French.

If the UserDict entry as specified in the registry entry cannot be found, the **TextToSpeechStartup()** function returns a value of MMSYSERR_ERROR. If the registry entry is missing, the **TextToSpeechStartup()** function defaults to looking in the application default directory for `user.dic` to be loaded as the user pronunciation dictionary.

If the user pronunciation dictionary fails to be loaded, then the

TextToSpeechStartup() function returns a value of MMSYSERR_ERROR.

Registry Entry Information

OEM customers can install the 32-bit code for DECtalk Software, together with other application software, on an end-user computer system, without using the DECtalk installation software. In such cases, you need to make registry entries on the end-user system to enable DECtalk Software to work correctly with your application. When the DECtalk engine starts, it checks for licenses, descriptions, numbers of instances, and so on in the registry entries.

An installation program that you create to install your application and the necessary components of DECtalk Software on an end-user system must set up registry entry information on that system.

In addition to installation considerations, you may also need registry entry information to resolve some kinds of errors. For example, DECtalk Software may report that the user dictionary is not found in the expected location. You can look up registry entry information in the following list to see how to interpret registry entries correctly and to see what registry entries have changed in the current version of DECtalk.

Note

Some earlier versions of DECtalk Software use **DigitalEquipmentCorporation** in the registry entries, as follows:

```
HKEY_LOCAL_MACHINE\SOFTWARE\DigitalEquipmentCorporation\DECta
lk\version\ langcode
```


Registry Entry Formats and Locations

```
[HKEY_LOCAL_MACHINE\SOFTWARE\DECtalk
Software\Classes\CLSID\{clsidno}]
"DECtalk_{langcode}"="DECtalk TTS Engine {langcode}"
[HKEY_LOCAL_MACHINE\SOFTWARE\DECtalk
Software\Classes\CLSID\{clsidno}\InprocServer32]
@="C:\WIN95\Speech\dtlkttse_{langcode}.dll"
[HKEY_LOCAL_MACHINE\SOFTWARE\DECtalk
Software\Classes\Software\DECtalk\{version}
"Version"="DECtalk MultLang version {version}
"Language"="MULTI LANGUAGE"
```
```
[HKEY_LOCAL_MACHINE\SOFTWARE\DECtalk
Software\Microsoft\Windows\CurrentVersion\App Paths\{appname}.exe]
"Path"="C:\Program Files\DECtalk\;C:\Program Files\DECtalk\Help"
@="C:\Program Files\DECtalk\{appname}.exe"
[HKEY_LOCAL_MACHINE\SOFTWARE\DECtalk
Software\Microsoft\Windows\CurrentVersion\Uninstall\DECtalkDeinstK
ey]
"UninstallString"="C:\WIN95\uninst.exe -f\"C:\Program
Files\DECtalk\DeIsL6.isu\""
"DisplayName"="DECtalk V{version}"
[HKEY_LOCAL_MACHINE\SOFTWARE\DECtalk
Software\Voice\TextToSpeech\Engine]
"DECtalk_{langcode}"="{clsidno}"
[HKEY_LOCAL_MACHINE\SOFTWARE\DECtalk Software\DECtalk\{version}]
"Company"="q"
"Installer"="q"
"LicUpdPwd"="{licpswd}"
"Lock_MGR"="2"
"Licenses"="{liccount}"
```

```
"MultiLang"="1"
[HKEY_LOCAL_MACHINE\SOFTWARE\DECtalk
Software\DECtalk\{version}\{langcode}]
"Version"="DECtalk {langcode} version {version}"
"Language"="{langname}"
"MainDict"="C:\Program
Files\DECtalk\{langcode}\DTALK_{langcode}.dic"
[HKEY_LOCAL_MACHINE\SOFTWARE\DECtalk Software\DECtalk\Langs]
"US"="ENGLISH,US"
"DefaultLang"="UK"
"SP"="CASTILIAN SPANISH"
"LA"="LATIN AMERICAN SPANISH"
"GR"="GERMAN"
"UK"="ENGLISH,UK"
"FR"="FRENCH"
[HKEY_CURRENT_USER\Software\DECtalk
Software\DECtalk\{version}\{langcode}]
"UserDict"="C:\Program
Files\DECtalk\{langcode}US\USER_{langcode}.dic"
```
3-28

Registry Entry Key

• Where langcode is "US", "UK", "SP", "LA", "GR", or "FR".
Quotation marks are not part of the value.
• Where langname is the language name for a specific
language:
langcode langname
US "ENGLISH,US"
SP "CASTILIAN SPANISH"
LA "LATIN AMERICAN SPANISH"
GR "GERMAN"
UK "ENGLISH,UK"
FR "FRENCH"
Quotation marks are not part of the value.
• Where version is "5.00" or current version number.
• Where appname is the name of an application using
DECtalk.
• Where licpswd is encrypted password for licenseu.exe
program.
• Where liccount is encrypted number of authorized engine
licenses.
• Where clsidno is the CLSID numbers for a specific
langcode:
langcode clsidno
"US" "{ED737300-8FCB-11ce-AB5D-00AA00590F2B}"
"SP" "{99EE9560-A4A6-11d1-BEB2-0060083E8376}"
"LA" "{99EE9550-A4A6-11d1-BEB2-0060083E8376}"
"GR" "{99EE9570-A4A6-11d1-BEB2-0060083E8376}"
"UK" "{99EE9540-A4A6-11d1-BEB2-0060083E8376}"
"FR" "{99EE9580-A4A6-11d1-BEB2-0060083E8376}"
You must include the {} around the clsidno specified.
Quotation marks are not part of value.
3-29

Sample Programs (UNIX)

Some applets are included with DECtalk Software. These sample applications

are included to demonstrate the use of DECtalk Software APIs. These sources can be used as templates for other applications that you might want to develop. Some of the applets demonstrate the DECtalk Software API extension to the multimedia API specified in the Multimedia Services for Tru64 UNIX software product. These applets are those that play back speech samples to an audio device and are built using the `libttsmme.so` shared library.

Other applets demonstrate the DECtalk Software API that does not include the multimedia extensions. These applets are those that do not require any audio devices. These applets can be built using the `libtts.so` shared library.

Sources to these programs can be found in:

`/usr/examples/dtk/dtsamples`

The samples and a brief description are listed below.

Applet Description

xmsay Demonstrates the use of DECtalk Software APIs in the Motif windows environment.

say Converts text into speech samples that are sent to an audio device.

dtmemory Converts text into speech samples stored in memory buffers.

aclock Announces the time at specified intervals.

3-30

Building Sample Programs

Sample programs can be created from the sources provided in `/usr/examples/dtk/dtsamples`. This section describes the procedure for building the sample programs. Before proceeding, make sure that the DECtalk Software development kit has been installed. See the *DECtalk Software Installation Guide* for more information on different components of DECtalk Software.

1. Create a local directory in which you want to build the sample programs.

2. Copy all the files in `/usr/examples/dtk/dtsamples` into the directory that you just created.

3. Generate a *Makefile* from the *Imakefile* by typing:

`usr/bin/X11/xmkmf`

4. Compile and link the sample application programs by typing the following command while still in the directory that you just created:

`make all`

5. After the make program completes successfully, the sample programs are ready to run.

In addition to the sample programs, you will also find some demo text files in your directory. These files demonstrate some of the DECtalk Software capabilities.

Header Files

DECtalk Software provides header files that contain all the public data-structure definitions to which the DECtalk Software API refers. They are `ttsapi.h`, `dtmmedefs.h`, and `L_ALL_PH.H`. When DECtalk Software is installed, these files are in `/usr/include/dtk`.

• **ttsapi.h** contains definitions of constants used in the DECtalk Software API functions, data structures that define the buffers that DECtalk Software returns, and the API function prototype definitions.

• **dtmmedefs.h** contains the basic data structure definitions used by DECtalk Software. It also contains definitions of error codes and audio formats. This file enables you to compile, link, and run certain DECtalk Software programs even if Multimedia Services for Tru64 UNIX is not

3-31

installed. Specifically, if you are writing an application program that does not use the audio drivers, but you want to use DECtalk Software to produce synthesized speech buffers using the **TextToSpeechOpenInMemory()** function, you can use `dtmmedefs.h` to circumvent the requirement for Multimedia Services for Tru64 UNIX.

• **L_ALL_PH.H** contains all phoneme codes for all supported languages. (Before DECtalk Version 4.62, a separate L_*LANGCODE*_PH.H file was provided per language.) Within L_ALL_PH.H, phonemes are listed with a language prefix, *LANGCODE_*, where *LANGCODE_* is one of the following:

US_ English (United States)
GR_ German
SP_ Castilian Spanish
LA_ Latin American Spanish
UK_ English (United Kingdom)
FR_ French

Shareable Libraries
DECtalk Software APIs are available to programmers in two shareable libraries.

• **libtts.so** contains device independent DECtalk Software routines.

• **libttsmme.so** contains the DECtalk Software library that requires Multimedia Services for Tru64 UNIX.

As in the case of the header files, if you want to use DECtalk Software to write an application that produces buffers of synthesized speech, then the program is linked with the `libtts` shared library. If, on the other hand, you want to use Multimedia Services for Tru64 UNIX to communicate with the audio subsystem, the application has to be linked with the `libttsmme` shared library.

3-32

Sample Programs (Windows)
Sample programs include:

dtsample This is an example of a basic Window editor with integrated text-to-speech.
Say This is an example of a command-line program using DECtalk Software.
dtmemory This is an example of synthesis into memory buffers.
ttstst This is a sample program to demonstrate the Microsoft speech API interfaces supported by the DECtalk Software speech engine.

The sample programs are furnished as examples of simple Windows applications you can build, using the DECtalk Software API. The source code for the sample programs is placed in the root directory, `\DECTALK\SAMPLE\`, during the installation. These programs include the API definition file `ttsapi.h` and are linked with `dectalk.lib`.

Using the DECtalk Software Development Kit (the Win32 SDK must be installed):

1. Bring up a Windows console window.
2. Set your default directory to the `\DECTALK\SAMPLE\DTSAMPLE` directory.
3. Enter **nmake**

This procedure produces the `dtsample.exe` executable file. This file can be executed at the Windows console window or the File Manager.

The dtsample program demonstrates most of the DECtalk Software API functions. Any application can call any of the API functions by including the ttsapi header file (`ttsapi.h`) and linking with the DECtalk Software library (`dectalk.lib`), as long as the associated dll file (`dectalk.dll`) is present.

The `dectalk.dll` file must be installed and the dictionary file `dectalk.dic` must be in the dictionary path in the registry or in the local directory.

Chapter 4 —
Basics of Multi-Language Programming

Multi-language programming using DECtalk Software requires loading a DECtalk language, selecting a loaded language for a program thread, starting a TTS instance, closing the program thread, and closing that language. Swapping a language requires shutting down and closing the thread and the language selector. You can load as many languages as you want during the execution of your program as well as select two or more languages concurrently. The following steps summarize the methodology to start, select, and close a single language.

Note
You must start and select a language before using DECtalk Software to speak. Multi-language programming is supported by the dynamic engine only. The static engine does not support multi-language programming.

The basic elements of multi-language programming are presented as follows:

- Starting a language
- Selecting a language
- Closing a language
- Multi-language programming example

4-2

Starting a Language

To start a DECtalk language, use the **TextToSpeechStartLang()** function to pass the two-letter name of the language you want to load. After the language is loaded, you receive a handle to that language. If the language is already loaded, you get the previously loaded handle to that language as the return value. The same handle is used for all instances of a language per run. On failure, you receive a handle with the TTS_LANG_ERROR bit set as defined in `ttsfeat.h`

Example:
```
TextToSpeechStartLang ("uk")
```

Selecting a Language

Before a call can be made to any of the standard DAPI functions, and after you have the handle to a language, you must select the language. This is accomplished through the **TextToSpeechSelectLang()** function. You must select the language before each call to any DAPI function that does not take in a valid LPTTS_HANDLE_T handle. The first parameter to **TextToSpeechSelectLang()** must be NULL. The second parameter is the handle retrieved from **TextToSpeechStartLang().** The return value from **TextToSpeechSelectLang()** is a Boolean value: TRUE signifies success and FALSE signifies failure.

Example:
```
TextToSpeechSelectLang (NULL,UKhandle)
```

Closing a Language

After you close your program threads, use the **TextToSpeechCloseLang()**

function to close the language. This function accepts the two-character language name of the language to release and then attempts to release the language and its associated files. If other threads are still using the language, the **TextToSpeechCloseLang()** function reduces the instance counter by one for that language. The file is free when all language hooks are freed.

Example:
```
TextToSpeechCloseLang ("uk")
```
4-3

Example

This example shows DECtalk ML allocating English (us), creating a thread, destroying the thread, and finally closing the language.
```
void sample(void) {
unsigned long int UShandle;
MMRESULT result;
LPTTS_HANDLE_T phTTS;
UShandle = TextToSpeechStartLang("us");
if (UShandle & TTS_LANG_ERROR) {
printf("Error loading US English\n");
exit(1);
}
if (TextToSpeechSelectLang(NULL, UShandle) == FALSE) {
printf("Error selecting language\n");
TextToSpeechCloseLang("us");
exit(1);
}
result = TextToSpeechStartup(hWnd, &phTTS, WAVE_MAPPER,
REPORT_OPEN_ERROR);
if (result != MMSYSERR_NOERROR) {
printf("Unable to start DECtalk Speech Engine\n");
TextToSpeechCloseLang("us");
exit(1);
}
TextToSpeechShutdown(phTTS);
TextToSpeechCloseLang("us");
printf("Example completed successfully!\n");
return;
}
```
Glossary - 1

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 in-line command.

Glossary - 2

.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 is accessed by applications as .lib files, using a dynamic link library (DLL). 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. The dynamic engine object code is not part of the application's executable code, except for runtime link references to the dynamic engine.

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 durations 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 **TextToSpeechCloseLogFile()** function returns the text-to-speech system to the startup state.

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

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

phoneme
The smallest unit of speech that distinguishes one word from another. Phonemes are divided into vowel and consonant phonemes. DECtalk Software interprets text 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 past tense verb *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 **TextToSpeechStartup()** or **TextToSpeechStartupEx()** has been successfully called and the text-to-speech system is *not* in one of the three special modes; wavefile, log-file, or speech-to-memory mode. While in 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 is accessed by applications as .lib files without using dynamic link libraries (DLLs). The static engine object code is part of the application's executable code. See also **dynamic engine**.

syntactic function words
A set of words that are either unstressed or have secondary stress. They include prepositions, conjunctions, determiners, auxiliary verbs, pronouns, 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.
Glossary - 6

user dictionary
The dictionary that you define 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 textto-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 or tab) 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

T

U