

# **VoxFonts<sup>TM</sup>**

**The “Software Only” Text-to-Speech Library**

**Revision A (October 1996)**

**Version 3.0**

VXF-000-012

# Contents

|  |           |
|--|-----------|
| ◆ <b>USER'S GUIDE</b> .....                                    | <b>1</b>  |
| INTRODUCTION.....  | 3         |
| <i>How it works:</i> .....                                     | 4         |
| <i>Windows and DOS!</i> .....                                  | 4         |
| <i>Hardware Requirements</i> .....                             | 5         |
| <i>Using a Math Coprocessor</i> .....                          | 5         |
| <i>Software Requirements</i> .....                             | 6         |
| <i>Notational Conventions</i> .....                            | 6         |
| <i>VoxFonts Technical Capabilities</i> .....                   | 7         |
| SAMPLE APPLICATION.....  | 9         |
| <i>Start-up Checklist</i> .....                                | 9         |
| <i>What We'll Do</i> .....                                     | 9         |
| <i>Running the DOS Sample Program</i> .....                    | 10        |
| <i>DOSTst Screen Description</i> .....                         | 11        |
| <i>Running the Windows Sample Program</i> .....                | 12        |
| <i>WinTst Screen Description</i> .....                         | 13        |
| ◆ <b>REFERENCE GUIDE</b> .....                                 | <b>15</b> |
| ABOUT TEXT-TO-SPEECH.....                                      | 17        |
| <i>Putting it all Together: How Text-to-Speech Works</i> ..... | 24        |
| <i>What is VoxFonts™?</i> .....                                | 25        |
| FUNCTION OVERVIEW.....   | 29        |
| <i>VoxFonts Standard Functions</i> .....                       | 30        |
| <i>VoxFonts DOS Replacement Functions</i> .....                | 32        |
| FUNCTION REFERENCE.....  | 33        |
| <i>Overview</i> .....  | 33        |
| <i>The FntTst Program</i> .....                                | 33        |
| <i>Visual Basic Interface</i> .....                            | 34        |
| <i>VoxFonts SDK Functions</i> .....                            | 34        |
| <i>VoxFonts Standard Functions</i> .....                       | 34        |
| <i>VoxFonts DOS Replacement Functions</i> .....                | 40        |
| ◆ <b>APPENDIX</b> .....  | <b>45</b> |

INSTALLATION & SETUP.....47  
    *VoxFonts Software Installation* .....47  
    *The VoxFonts SDK Directories* .....48  
TROUBLESHOOTING & RECOVERY .....49  
    *System Configuration*.....49  
    *Error and Warning Messages*.....50  
    *Error Messages* .....50  
    *Warning Messages* .....51  
    *Technical Support* .....51  
LICENSING.....55  
    *Licensing* .....55  
**◆ GLOSSARY .....59**  
**◆ INDEX .....64**

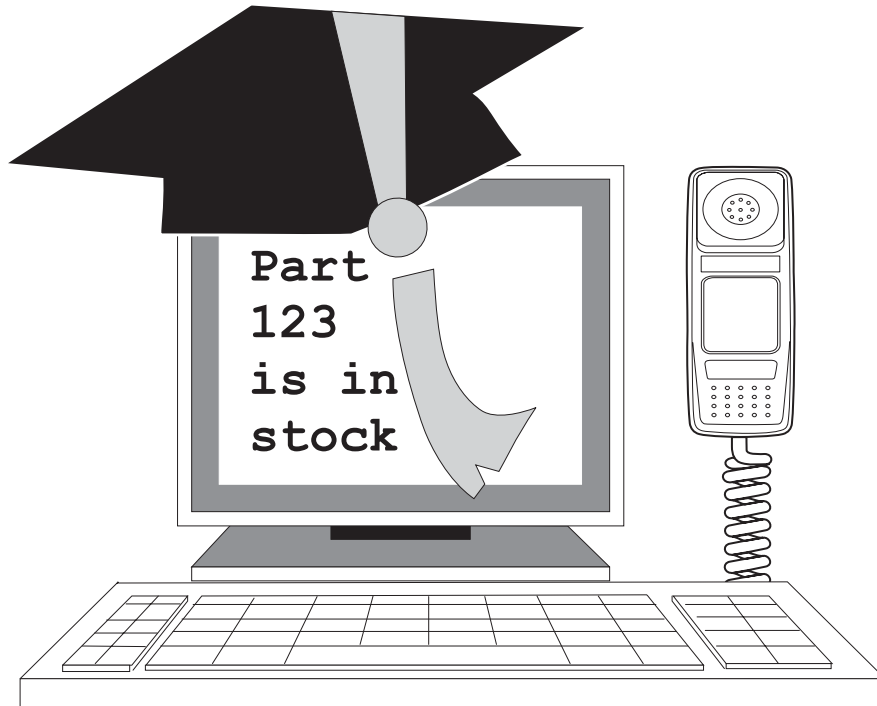
# ◆ User's Guide

Chapter 1, *Introduction*, provides a brief overview of *VoxFonts*—what it does and how it works. This chapter covers hardware and software requirements as well as *VoxFonts* technical capabilities.

Chapter 2, *Sample Session*, gives you a step-by-step, hands-on introduction to using *VoxFonts*.



# Introduction



**VoxFonts™** lets your PC actually read out loud!

Tired of recording a gazillion prompts? Wouldn't it be nice if your computer could just read typed text? Well, now it can with **VoxFonts™**!

With **VoxFonts**, your PC can actually *read out loud*!

**VoxFonts** converts typed text into spoken words. Banks use text-to-speech to deliver customer balances. The blind use it to access computer information. Imagine the possibilities! Let our **VoxFonts** text-to-speech software add an unlimited vocabulary to your Interactive Voice Response (IVR), internet, and multimedia applications. Now you can take advantage of text-to-speech too!

**VoxFonts** is a text-to-speech synthesis library that translates ASCII text into a digital audio file. Then play that file with your existing telephony, network, or multimedia hardware. **VoxFonts** provides speech synthesis comparable to hardware assisted systems — but at a fraction of the cost!

*VoxFonts* gives you Microsoft “C” (DOS & Windows) and Visual Basic callable functions for converting ASCII text into a digital file. And you can add translation rules or specify pronunciations for difficult or foreign words!

No more hours of recording to produce every possible message — your vocabulary’s now unlimited, just a function call away!

## How it works:

*VoxFonts* works by first reading a text string, then breaking it down into individual words. These words are then analyzed, and according to a sophisticated set of rules are subdivided into their phonetic “sounds” using the international phonetic alphabet.

*VoxFonts* then takes those phonetic sounds and synthesizes the corresponding audio from low level speech components. That synthesized speech is then stored in a digital audio file.

Once *VoxFonts* has stored the audio in a file, you are free to work with it in a variety of ways. Play the file using your multimedia or telephony hardware, transmit it over a network to be played elsewhere, or you can save it for future use.

## Windows *and* DOS!

VISI’s *VoxFonts Software Development Kit (SDK)* lets you work in both the MS Windows and MS DOS environments. The *VoxFonts SDK* contains sample programs that use the libraries and DLLs. These programs were created using Visual Basic and Microsoft C version 8.0, and demonstrate usage under both Windows and DOS. The sample program performs a simple text-to-speech conversion to both “.WAV” and “.VOX” formats.

## With *VoxFonts* You Can Do It

This innovative program gives you the capability to do with English text what you can do with pre-recorded messages. *VoxFonts* lets you automatically create your recordings. Automatically generate test messages to prototype a system design — fill in messages until the final version is done — combine recorded and generated message to read virtually any type of data.

## Skills

The *VoxFonts SDK* is a programming library designed for Microsoft “C” (DOS & Windows) and Visual Basic developers. We’ve made the product as easy as possible to use, but simple Visual Basic or “C” programming knowledge is required. Some knowledge about digital audio is helpful, but not required.

## Put *VoxFonts* to Work for You

All this is fascinating, but perhaps it’s time to get to work and earn some money. Maybe you’re working for a parts distributor. You want people to call in from a phone and find whether a certain part number is in stock. Because you carry 10,000 different parts, you don’t want to record the names of each. Simply have *VoxFonts* read the names for you as part of your interactive database application.

Or maybe you develop voice systems. Your customer will eventually have all the prompts recorded from the script that you’ve been handed - but in the meantime, what do you do? You don’t want to record all those message yourself (just to be thrown away later)! Have *VoxFonts* generate the audio files for you from the scripts provided, without having to call in expensive outside help.

## Hardware Requirements

To run *VoxFonts* you’ll need an IBM PC (or compatible), with a hard drive, and at least 512K of memory. A 486 or coprocessor and 640K is recommended.

*VoxFonts* DLLs require Microsoft Windows, 8 megabytes of memory, and at least 4 megabytes of space on your hard disk.

## Using a Math Coprocessor

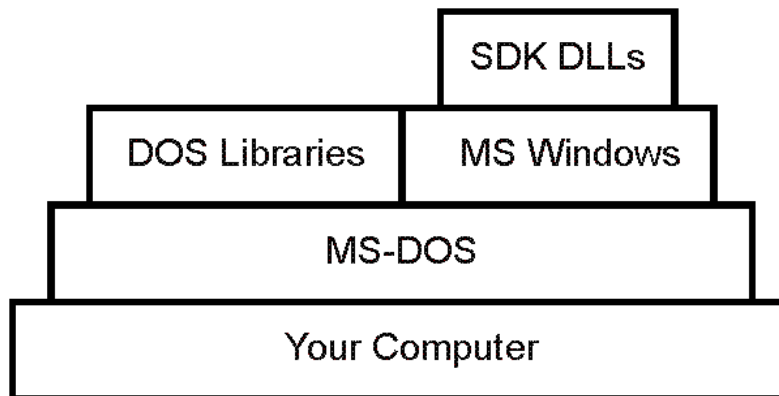
*VoxFonts* functions can benefit from the use of a math coprocessor. *VoxFonts* will automatically detect and use an 80287/80387 (or compatible) math coprocessor if one is installed. Similarly, *VoxFonts* will automatically detect the presence of the coprocessor built into chips such as the Intel 80486DX or Pentium®.

Following are the times required to produce a 10 minute (600 second) long 8 kHz 4-bit ADPCM speech file:

| Processor Type | Time in seconds | Speed ratio    |
|----------------|-----------------|----------------|
| 80386/33       | 50              | 12:1           |
| 80386/80387    | 40              | 15:1           |
| 80486/66       | 30              | 20:1           |
| 80586/133      | 17              | 36:1 (Windows) |
|                | 13              | 44:1 (DOS)     |

## Software Requirements

To run *VoxFonts* you'll need DOS 3.3 or higher. To run *VoxFonts* Dynamic Link Libraries (DLLs) you'll need Microsoft Windows 3.x, Windows 95, or Windows NT.



## Notational Conventions

Here are a few notes about terms and typographical conventions used in this manual.

The names of keys are spelled out and appear in capital letters. On your keyboard the keys may be marked with an abbreviation or be named a little differently.

The names of keys are shown within less-than “<” and greater-than “>” signs. For example, an enter key (or return key) is shown as <ENTER>.

- ◆ **Simultaneous Activation:** When two or more keys must be pressed at the same time, they will be shown joined without punctuation. These combinations are shown as “<KeyA><KeyB>”. For example, holding down the CONTROL key while tapping the BREAK key is shown as <CTRL><BREAK>.

**Sequential Activation:** If keys are to be pressed one at a time and released, then they will be separated by a comma. These key sequences are shown as “<KeyA>,<KeyB>”. For example, holding down the ALTERNATE key while tapping the D key, followed by tapping the H key is shown as <ALT><D>,<H>.

## VoxFonts Technical Capabilities

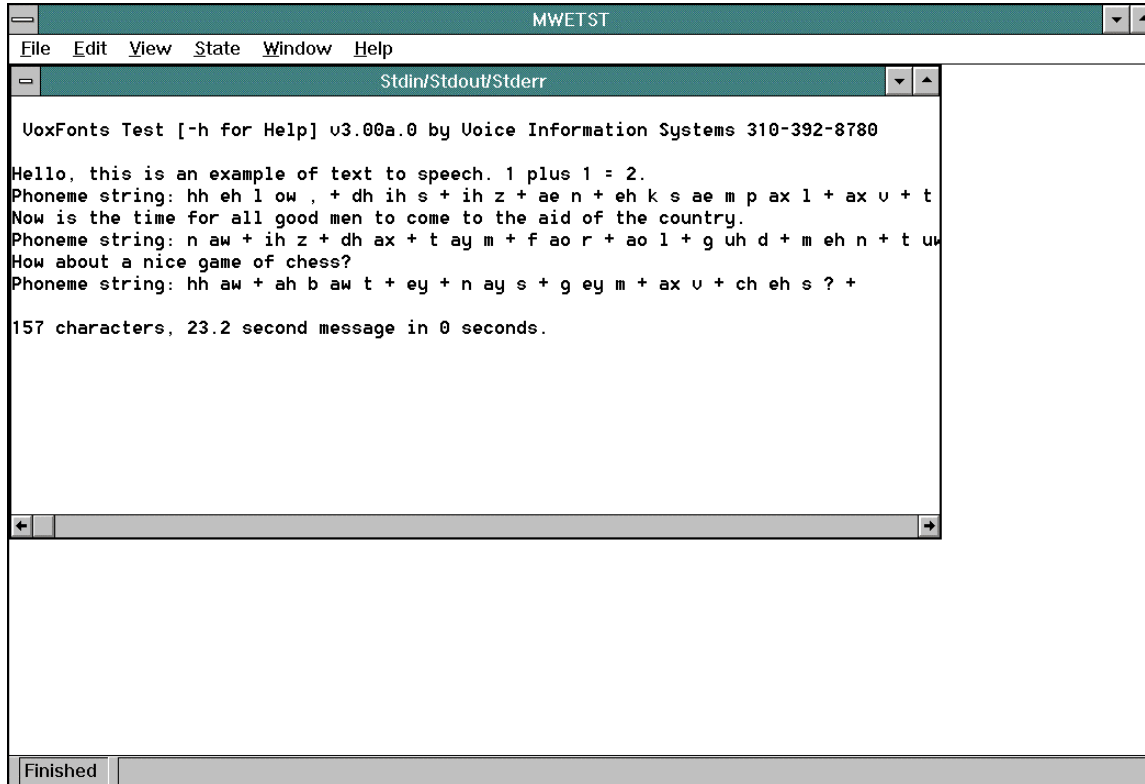
**VoxFonts** is the **only** stand-alone software text-to-speech library for the DOS and Microsoft Windows<sup>®</sup> environments that supports both telephony and multimedia audio formats. “Software only” means you that you don’t need any additional hardware to run the product 🕒 **VoxFonts** takes advantage of the signal processing power of the PC. The ability to use both the DOS and Microsoft Windows<sup>®</sup> environments and support for multimedia audio formats means outstanding flexibility in how you package your application.

### Main Features:

- ◆ **No Added Hardware:** Works with your existing multimedia, network, or voice response hardware 🕒 nothing to add!
- ◆ **Simple, Powerful:** Simple programming interface maintains maximum compatibility with existing standards. Audio file generation guarantees powerful integration into existing systems!
- ◆ **Unlimited File Length:** Our disk and speech caching technology lets you read any length message - yet uses as little as 128K of memory!
- ◆ **Fast:** Better than 30:1 conversion speed on a 586 class computer. This means you can generate 1 minute of speech in less than 2 seconds time!
- ◆ **Multi-line:** Supports multiple lines or servers with the same library!
- ◆ **Industry Standard Formats:** Supports both Dialogic compatible telephony and Multimedia Wave audio formats!
- ◆ **Flexible licensing options:** VISI offers a wide variety of licensing options, including unlimited distribution right out of the box. Private labeling and multi-threaded processing options are also available!

- ◆ ***Integration Control:*** Complete control of translation. User replaceable memory functions for advanced DOS resource control!
- ◆ ***DOS and Windows, C and VB:*** Integrate with your “C”, Visual Basic, and other Windows compatible programming languages!
- ◆ ***Pronounce any word:*** Built-in translation rules are ready to speak any word!
- ◆ ***Fully Programmable:*** Add translation rules or specify pronunciations for difficult or foreign words!

# Sample Application



```
VoxFonts Test [-h for Help] v3.00a.0 by Voice Information Systems 310-392-8780
Hello, this is an example of text to speech. 1 plus 1 = 2.
Phoneme string: hh eh l ow , + dh ih s + ih z + ae n + eh k s ae m p ax l + ax u + t
Now is the time for all good men to come to the aid of the country.
Phoneme string: n aw + ih z + dh ax + t ay m + f ao r + ao l + g uh d + m eh n + t u
How about a nice game of chess?
Phoneme string: hh aw + ah b aw t + ey + n ay s + g ey m + ax u + ch eh s ? +
157 characters, 23.2 second message in 0 seconds.
```

Finished

## *VoxFonts* Sample Session

## Start-up Checklist

This chapter contains a step-by-step, hands-on introduction to using *VoxFonts*. Since you will be working with the program, be sure that you have:

- ◆ Installed the *VoxFonts* software; Appendix A, *Setup and Installation*, tells you how to set up *VoxFonts* on your computer.

## What We'll Do

During installation we loaded some sample files onto your hard disk for you to use. We'll tell you how to read what is on your screen in both the DOS and Windows environments, and how to perform a simple text-to-speech conversion. We'll also

familiarize you with the command line options available to you in the *VoxFonts* sample application.

Some customers prefer to use *VoxFonts* from the DOS environment, some from the Windows environment, and of course, some use both. If you prefer work solely from the Windows environment, simply skip to the section named *Running the Windows Sample Program*.

## Running the DOS Sample Program

First, move into the *VoxFonts* directory. For example, if you installed *VoxFonts* in the C:\VoxFnt directory, enter:

```
C: <ENTER>CD \VoxFnt <ENTER>
```

Run the sample DOS program and create the file TSTMSG.VOX by entering:

```
DOSTst TstMsg.Vox -tFntTst.Txt <ENTER>
```

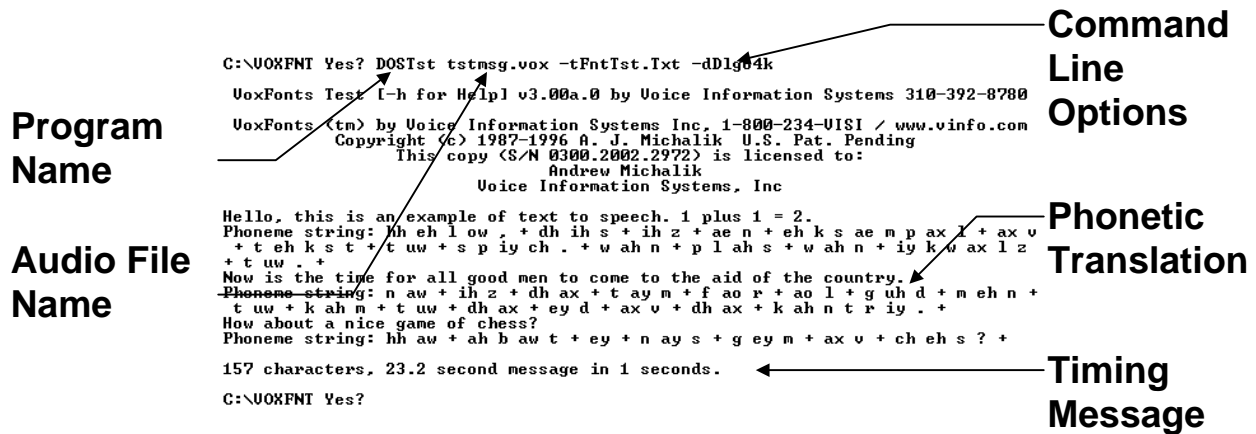
You should now see the file being converted line by line.

Now play the file TSTMSG.VOX through the your voice response systems, or with a playback utility such as VISI's *Scribe Transcription Playback Utility*, or with VISI's *VFEdit*® *Professional Prompt Editor*.

Press the <ENTER> key to stop the program.

The DOSTst application demonstration provides a simple example of how to use *VoxFonts* functions under DOS. Please refer to the actual source code for more detailed programming information.

## DOSTst Screen Description



The *Program Name* appears immediately following the DOS command prompt, and is the name of the sample executable file.

The *Audio File Name* appears immediately following the *Program Name*, and is the name of the audio file that you wish to create.

The *Command Line Options* control the operation of the program, and are used to tell the sample program how to perform its job. When run, *VoxFonts* checks the command line for these user-specified options.

The *Phonetic Translation* appears immediately following each line of ASCII text, and is the International Phonetic Alphabet (IPA) representation of the input text.

The *Timing Message* appears immediately following the last line of *Phonetic Translation*, and contains valuable information regarding the speed and performance of the conversion process.

---

## DOSTst

### FntTst [-h] [VoxFile] [-d -tTxtFile]

This sample program performs a basic text-to-speech conversion. Source code is included for the user to copy and review.

| <u>Parameter</u> | <u>Description</u>              |
|------------------|---------------------------------|
| VoxFile:         | Output file name specification. |
| -dXXXXXX         | Digitized data format           |

| <u>Value</u> | <u>Meaning</u>                      |
|--------------|-------------------------------------|
| DLG24K       | Dialogic 4 bit @ 6kHz.              |
| DLG32K       | Dialogic 4 bit @ 8kHz (default).    |
| DLG48K       | Dialogic 8 bit @ 6kHz.              |
| DLG64K       | Dialogic 8 bit @ 8kHz.              |
| WAV008       | Wave Multimedia 8 bit @ 11.025 kHz  |
| WAV016       | Wave Multimedia 16 bit @ 11.025 kHz |

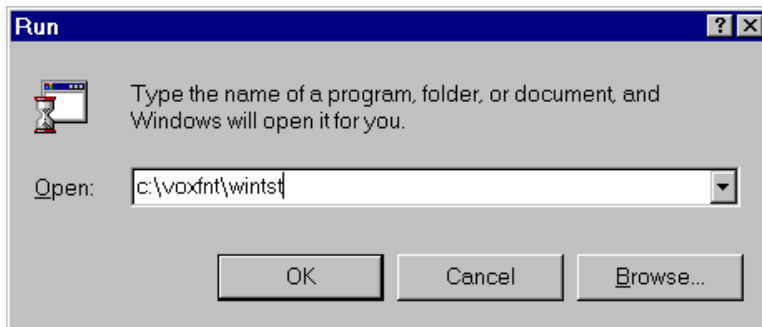
**-t**txtfile      Input text file.

### **Exiting**

Press the <ENTER> key to stop the program.

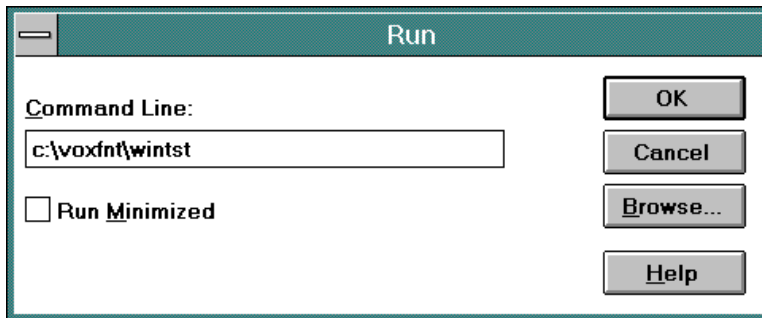
## **Running the Windows Sample Program**

If you are running *VoxFonts* under Microsoft Windows 95, use the Start Programs menu item to select the *VoxFonts* “WinTst” executable entry.



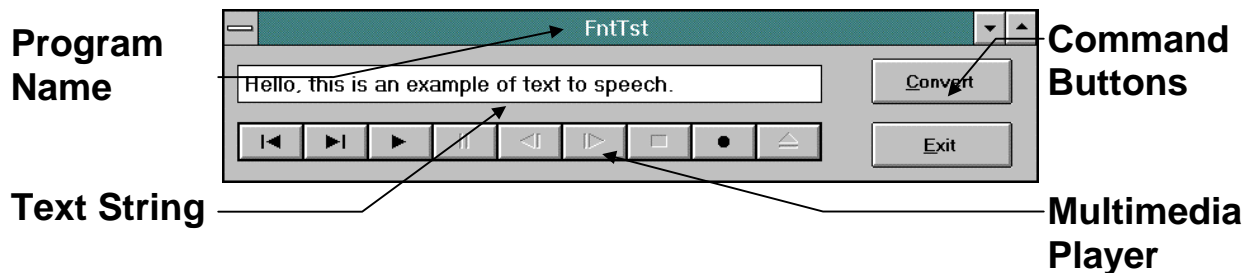
Windows 95 Start Run Dialog Box

If you are running *VoxFonts* under Microsoft Windows 3.x, use the Program Manager File Run menu item to select the *VoxFonts* “WinTst” executable.



Windows 3.x File Run Dialog Box

## WinTst Screen Description



The **Program Name** appears across the main window, and is the name of the sample executable file.

The **Text String** contains the line of English text that you wish to translate into a digital audio file.

The **Command Buttons** control the operation of the program, and are used to control operation of the program.

The **Multimedia Player** lets you play the converted speech file if you have a multimedia sound card installed in your system.

The program will prompt you for the text string you want read. You can use the sample string we have already provided, or type in your own string and press “Convert”. When completed, you will see the following dialog box:



Now play the file *FntTst.Wav* through the your multimedia sound card, or with a playback utility such as VISI's *Scribe Transcription Playback Utility*, or with VISI's *VFEdit*<sup>®</sup> *Professional Prompt Editor*.

Click on "Exit" to stop the program.

The *WinTst* application demonstration provides a simple example of how to use *VoxFonts* functions under Windows using Visual Basic. Please refer to the actual source code for more detailed information.

# ◆ Reference Guide

*VoxFonts* affords wide flexibility in handling text-to-speech processing with multiple levels of API calls. You can even add your own pronunciation rules for improved speech quality.

This guide covers specific information relating to the use of function calls. In addition, we have included a brief introduction to the principles behind text-to-speech.



# About Text-to-Speech

Text-to-speech works by first reading a text string, then breaking it down into individual words. These words are then analyzed, and according to a sophisticated set of rules, are subdivided into their phonetic “sounds” using the international phonetic alphabet.

A text-to-speech library then takes those phonetic sounds and synthesizes the corresponding audio from low level speech components. That synthesized speech is then stored in a digital audio file.

Once the audio has been stored in a file, you are free to work with it in a variety of ways. Play the file using your multimedia or telephony hardware, transmit it over a network to be played elsewhere, or you can save it for future use.

Some applications of computer voice output are:

- ◆ ways to transmit information from English language data bases to remote locations by telephone
- ◆ a way to communicate with users of products with complicated visual displays where additional text messages would be intolerable.
- ◆ reading machines for the blind.

In such applications the potential utility of computer controlled speech synthesizers is greatly enhanced if the speech is not restricted to a previously stored vocabulary.

## Phonemes

The smallest unit of speech is called a *phoneme*. Phonemes are actually smaller than syllables and even half syllables.

Consider the following example: The word “bee” is a monosyllabic (one syllable) word. However, the syllable “bee” is actually composed of two phonemes. The first phoneme is the sound of the letter **b** and has been named /b/. The second phoneme is the sound of the letters **ee** and is what we refer to as the long **e** sound. This sound is equivalent to the **ea** sound in “eat” and is named /e/.

Although a phoneme rarely appears as an entire word (e.g., /a/ as in **a** desk), a change in a single phoneme can change one word into another. For example:

| <u>Word</u> | <u>Phonemes</u> |
|-------------|-----------------|
|-------------|-----------------|

|     |         |
|-----|---------|
| bee | /b/ /e/ |
|-----|---------|

|     |         |
|-----|---------|
| pea | /p/ /e/ |
|-----|---------|

The phonetic symbols /p/ and /b/ are sufficiently different to signal a difference in meaning between the words “pea” and “bee”; however, the phonemes /p/ and /b/ mean nothing by themselves. Therefore, a phoneme does not typically have meaning but is used to distinguish meanings between words. This process of making speech sounds is known as *articulatory phonetics*.

There are approximately 42 phonemes in the English language.

## Allophones

To compare with the above example, another word which uses the phonemes /b/ and /e/ is “bleed.” In theory, the sounds /b/ and /e/ should sound exactly the same as the /b/ and /e/ sounds in “bee.” However, the sounds we actually make when speaking are affected by the sounds that precede and follow the particular sound. In this case, the /b/ as well as the /e/ is affected by the sound of the letter **l**. It should also be noted that the /b/ and the /e/ affect the sound of the letter **l**. The phonemes can be thought of as “blending together” at their “edges.” This blending does not create the addition of new phonemes. It actually creates a *variation* of a given phoneme, which is called an *allophone*. Let’s look at our example:

| <u>Word</u> | <u>Phonemes</u> |
|-------------|-----------------|
|-------------|-----------------|

|     |         |
|-----|---------|
| bee | /b/ /e/ |
|-----|---------|

|       |           |
|-------|-----------|
| bleed | /b/- /e/- |
|-------|-----------|

Since the sounds are slightly different from each other when pronouncing each word, the allophones may be represented as follows:

| <u>Word</u> | <u>Allophones</u> |
|-------------|-------------------|
|-------------|-------------------|

|     |       |
|-----|-------|
| bee | B1 E1 |
|-----|-------|

|       |       |
|-------|-------|
| bleed | B2 E2 |
|-------|-------|

For another example, consider the word “Dad.” The phonetic description of “Dad” would be: /d/ /æ/ /d/. The ideal /d/ sound would be considered a phoneme. However, the initial and final /d/ sounds are different: therefore, they can each be referred to as D1 and D2. D1 and D2 are allophones of the phoneme /d/. In essence, an allophone is the variation of a particular phoneme, depending upon its position in a word.

Phonemes and allophones can be combined to make whole words, large vocabularies can be stored in a small amount of memory. If the set of sub-word units is designed properly, *any* word in a given language can be constructed from this set.

## Coarticulation

We have just reviewed one characteristic of acoustic phonetics: the physics of the sound wave characteristics of speech signals. This was illustrated when comparing two words (“bee” and “bleed”) that use the same phonemes. Another characteristic of a speech waveform arises when trying to extract discrete speech sound from the continuously varying speech signal. During the pronunciation of a word, the articulators are constantly moving from one phoneme position to another. For this reason, each sound in the sequence influences every other sound around it. Because these speech sounds “overlap,” as previously described, a problem arises when extracting a speech sound that is “pure” (i.e., a sound that does not contain cues to other speech sounds). For example, if you try to extract the **b** sound from the word “brain” by taking larger portions of the acoustic signal from the beginning of the word, one would encounter a non speech-like noise and then the sound **br**. There is no point at which the **b** sound can be heard in isolation. Due to this *coarticulation effect*, information is lost when extracting these sounds and the speech quality is reduced somewhat. Taking this coarticulation effect into account results in more natural sounding speech.

## How to create words from basic sounds

Among the numerous approaches to providing such unrestricted text-to-speech translation, the simplest is to use a small set of letter-to-sound rules to guess at the pronunciation of any word. Each rule specifies a phonetic correspondence to one or more letters. In some cases the letter’s context is used to determine which rule should be applied. An example is the elementary school rule, “when two vowels go walking, the first one does the talking,” which indicates that when one vowel is followed by another, the first is transcribed into the long vowel phoneme whereas the second vowel is silent and receives no phonetic symbol. In other cases no context is necessary, as with the letter *j*, which usually receives the /dʒ/ phoneme. (The International Phonetic Alphabet, or IPA, will be used to denote English phonemes and indicate pronunciations.)

## Pronunciation Rules

A more sophisticated approach adds a large pronunciation dictionary supplemented by various sets of rules. Words are isolated from the text and looked up in the dictionary. If the lookup fails, various rules are used to break the word into constituent parts for which there are dictionary entries. Finally, if all else fails, letter-to-sound rules are used to guess at the pronunciation.

A text-to-speech library accepts text, applies the translation rules, and returns the translated results. The sample programs accept input from the keyboard or a text file. The complete translation from English to IPA may be requested by choosing the sequence of function calls.

The pronunciation rules are kept in character strings in a form easy for human beings to read and write. They are interpreted by the program to form the actual pronunciation. Each rule has the form

$$A [B] C = D$$

which means “The character string B, occurring with the left context A and right context C, gets the pronunciation D.”

D consists of IPA symbols ☹ or rather a capitalized Latin-letter representation of IPA to cater to computer character sets (see Table 1). B is a letter or text fragment to be translated. A and C are patterns; like B they may be strings of letters and other characters, but some special symbols denote classes of strings such “voiced consonants” and “vowel cluster”. Table 2 lists the symbols that have such special interpretations. Blanks are significant, because they identify the beginnings and ends of words.

Table 1

## Latin–Letter Representation of IPA

| Standard IPA | Representation | Example        | Standard IPA | Representation | Example         |
|--------------|----------------|----------------|--------------|----------------|-----------------|
| i            | IY             | be <u>et</u>   | g            | G              | go <u>at</u>    |
| ɪ            | IH             | bi <u>t</u>    | f            | F              | fa <u>ult</u>   |
| e            | EY             | ga <u>te</u>   | v            | V              | va <u>ult</u>   |
| ɛ            | EH             | ge <u>t</u>    | θ            | TH             | et <u>her</u>   |
| æ            | AE             | fa <u>t</u>    | ð            | DH             | ei <u>ther</u>  |
| a:           | AA             | fa <u>ther</u> | s            | S              | su <u>e</u>     |
| ɔ            | AO             | la <u>wn</u>   | z            | Z              | zo <u>o</u>     |
| o            | OW             | lo <u>ne</u>   | ʃ            | SH             | lea <u>sh</u>   |
| ʊ            | UH             | fu <u>ll</u>   | ʒ            | ZH             | lei <u>sure</u> |
| u            | UW             | fo <u>ol</u>   | h            | HH             | ho <u>w</u>     |
| ɜr           | ER             | mu <u>nder</u> | m            | M              | su <u>m</u>     |
| ə            | AX             | ab <u>out</u>  | n            | N              | su <u>n</u>     |
| ʌ            | AH             | bu <u>t</u>    | ŋ            | NX             | su <u>ng</u>    |
| aɪ           | AY             | hi <u>de</u>   | l            | L              | la <u>ugh</u>   |
| aʊ           | AW             | ho <u>w</u>    | w            | W              | we <u>ar</u>    |
| ɔɪ           | OY             | to <u>y</u>    | j            | Y              | yo <u>ung</u>   |
| p            | P              | pa <u>ck</u>   | r            | R              | ra <u>te</u>    |
| b            | B              | ba <u>ck</u>   | tʃ           | CH             | cha <u>r</u>    |
| t            | T              | ti <u>me</u>   | dʒ           | JH             | ja <u>r</u>     |
| d            | D              | di <u>me</u>   | ɹ            | WH             | we <u>re</u>    |
| k            | K              | co <u>at</u>   |              |                |                 |

Table 2

## Special Symbols Appearing in the English-to-IPA Translation Rules

| Symbol | Meaning  |
|--------|--|
| #      | One or more vowels*  |
| *      | One or more consonants†  |
| •      | One of B, D, V, G, J, L, M, N, R, W, and Z: a voiced consonant   |
| \$     | One consonant followed by an E or I  |
| %      | One of (ER, E, ES, ED, ING, ELY): a suffix   |
| &      | One of (S, C, G, Z, X, J, CH, SH): a sibilant  |
| @      | One of (T, S, R, D, L, Z, N, J, TH, CH, SH): a consonant influencing the sound of a following long <u>u</u> (cf. <u>rule</u> and <u>mule</u> ) |
| ^      | One consonant  |
| +      | One of (E, I, Y): a front vowel  |
| :      | Zero or more consonants  |

\* Vowels are A, E, I, O, U, Y.

† Consonants are B, C, D, F, G, H, J, K, L, M, N, P, Q, R, S, T, V, W, X, Z.

For example, a typical rule is

$$C[O]M = /AA/$$

which means that an O after an initial C and before an M gets the pronunciation /a/, the a-sound in father. Another rule is

$$:[E] = /IY]$$

Where the colon denotes any sequence of zero or more consonants, which means that final e, if the only vowel in a word, gets the long-e sound /i/ of be and she.

The translation algorithm scans input text from left to right and, for each character scanned, sequentially searches the rules pertinent to that character until it finds one whose left-hand side matches the text at the correct position. It outputs the right-hand side, passes over the characters bracketed in the rule, and resumes the scan with the next character of text. The input string is never altered.

To illustrate the operation of the algorithm, we will describe a worked example: the translation of *RATIO* using the English-to-IPA rules.

To the left of the first character, *R*, the program adds a blank to delimit the word, and the scan starts with the *R*, as we indicate with a pointer:  $\uparrow$ *RATIO*. The program searches the *R* rules—the rules with *R* as the first character between brackets. The first *R* rule, “[*RE*] ^ # = /*R IY*/” fails to match, since it requires that *R* be followed by *E*. The next, and last, rule, “[*R*] = /*R*/”, is the default; it matches any *R* not matched by earlier rules. Consequently, /*R*/ goes into the output string, and the scan moves past the *R* to *A*: *R* $\uparrow$ *ATIO*.

The search of the *A* rules turns up no match before “[*A*] ^ + # = /*EY*/”, which applies when *A* is followed by a single consonant, a front vowel (*E*, *I*, or *Y*), and another vowel. The program adds /*EY*/ to the output and moves the pointer past *A* to *T*: *RA* $\uparrow$ *TIO*.

The first *T* rule that matches is “[*TI*] O = /*SH*/”. Consequently, /*SH*/ goes into the output, and the pointer moves past *TI* to *O*: *RATI* $\uparrow$ *O*. The program does not search the *I* rules, since the *I* occurs inside the brackets with the *T*; the string *TI* as a whole gets the pronunciation /*SH*/ and no output phonemes correspond to *I* alone.

The first match among the *O* rules is “[*O*] = /*OW*/”; the program outputs /*OW*/ and moves the pointer past the *O* to the blank at the end of the word: *RATIO* $\uparrow$ . The output string is /*R*/ /*EY*/ /*SH*/ /*OW*/, which represents the IPA /*re* $\downarrow$ *o*/, the correct transcription [12]. If the translation continued, the next matching rule would be in the set that passes blanks, commas, periods, and other punctuation into the output string as /< >/, /< , >/, /< . >/, etc. The program would output /< >/ and move the pointer past the blank to the beginning of the next word, if any.

The IPA output string is the input to the speech synthesizer which generates the actual audio and stores this information in a file for later use.

## About Digital Audio

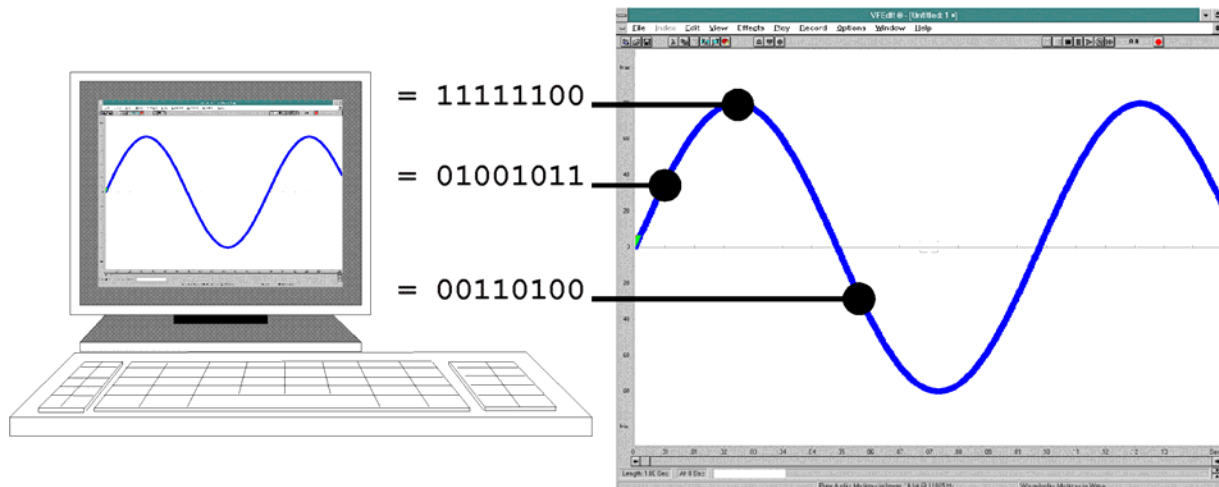
Computers are designed to be operated in the digital realm, where everything is a series of **on** or **off** voltages, formatted as bits. To store a sound in the computer we must convert the continuously changing analog audio signal into digital data. The circuitry that transforms data from analog to digital, and vice-versa is termed the *DAC* (digital to analog converter).

At regular intervals a *DAC* instantaneously freezes the audio signal voltage and holds it steady while another circuit selects the binary code that most closely

represents the sampled voltage. The DAC outputs a number in binary format that represents the input signal at any given instant in time.

Digital-to-analog conversion (for playback) is the exact opposite. In digital-to-analog conversion, the digital data is converted to a continuously changing series of voltage levels. The shape of this continuously changing stream of voltage levels approximates the shape of the original wave. This signal is then passed through a low-pass filter, which removes the digital “switching noise.”

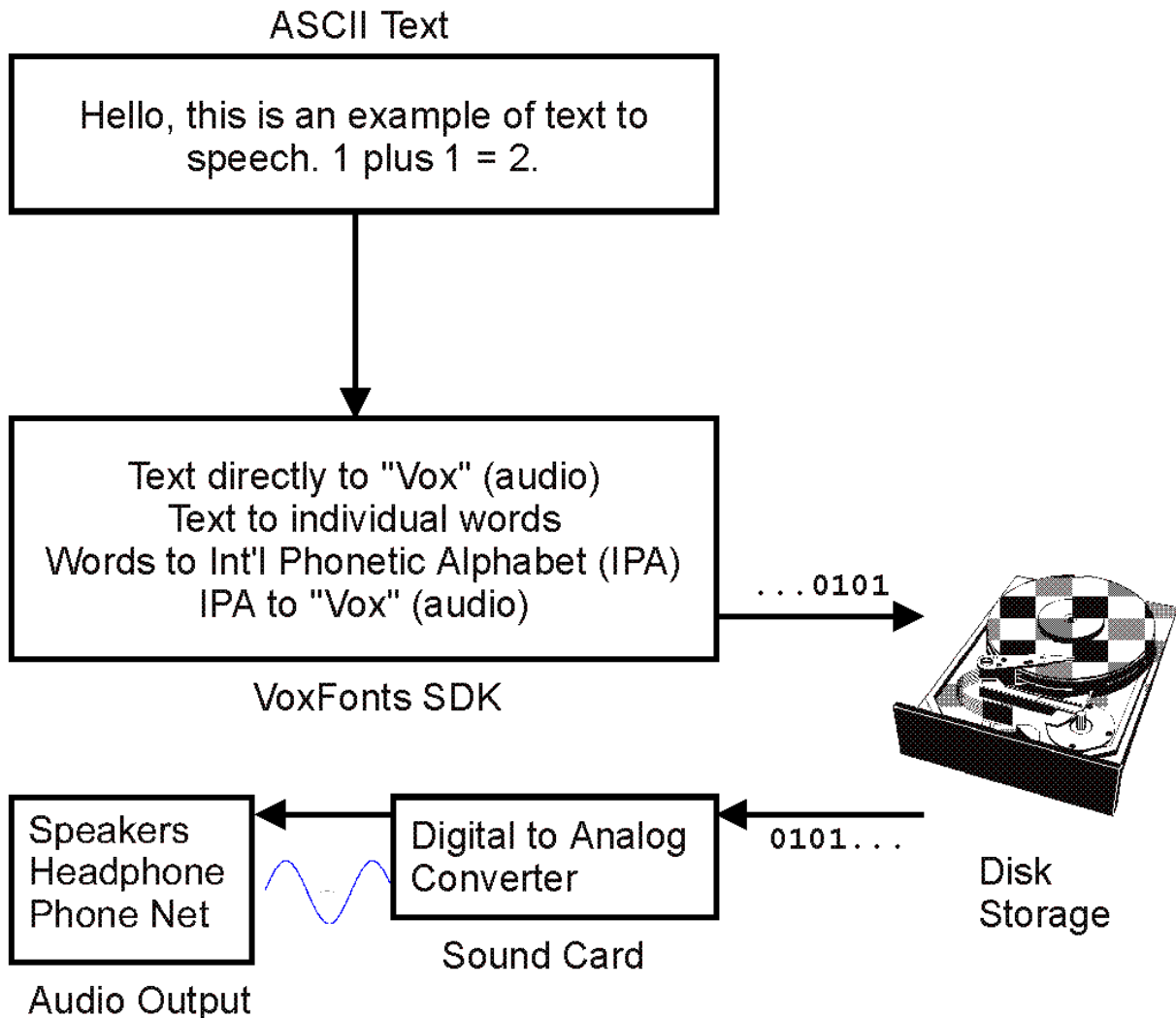
Once in digital form, the audio is essentially immune to degradation caused by system noise or defects in the storage or transmission medium (unlike older analog systems). The digitized audio signal is easily stored on a hard disk drive, where it can be kept indefinitely without loss of fidelity.



Digital to Analog Conversion

## Putting it all Together: How Text-to-Speech Works

Utilizing these basic sounds of the English language, an unlimited vocabulary can be created for your personal computer. That synthesized speech is then stored in a digital audio file. The following figure provides a block diagram of how the text-to-speech library interacts with the components you’ve seen so far.



How Text-to-Speech Works

## What is *VoxFonts*™?

*VoxFonts* is a text-to-speech synthesis library that translates ASCII text into a digital audio file. Then play that file with your existing telephony, network, or multimedia hardware. *VoxFonts* provides speech synthesis comparable to hardware assisted systems — but at a fraction of the cost!

*VoxFonts* works by first reading a text string, then breaking it down into individual words. These words are then analyzed, and according to a sophisticated

set of rules, are subdivided into their phonetic “sounds” using the International Phonetic Alphabet.

*VoxFonts* then takes those phonetic sounds and synthesizes the corresponding audio from low level speech components. That synthesized speech is then stored in a digital audio file.

Once *VoxFonts* has stored the audio in a file, you are free to work with it in a variety of ways. Play the file using your multimedia or telephony hardware, transmit it over a network to be played elsewhere, or you can save it for future use. And *VoxFonts* has been specifically designed to deal with the specialized hardware and audio formats used by telephony systems.

```

MWETST
File Edit View State Window Help
Stdin/Stdout/Stderr
VoxFonts Test [-h for Help] v3.00a.0 by Voice Information Systems 310-392-8780
Hello, this is an example of text to speech. 1 plus 1 = 2.
Phoneme string: hh eh l ow , + dh ih s + ih z + ae n + eh k s ae m p ax l + ax u + t
Now is the time for all good men to come to the aid of the country.
Phoneme string: n aw + ih z + dh ax + t ay m + f ao r + ao l + g uh d + m eh n + t u
How about a nice game of chess?
Phoneme string: hh aw + ah b aw t + ey + n ay s + g ey m + ax u + ch eh s ? +
157 characters, 23.2 second message in 0 seconds.
Finished

```

### ***VoxFonts***: The “Software Only” Text-to-Speech Library

Maybe you’re working for a parts distributor. You want people to call in from a phone and find whether a certain part number is in stock. Because you carry 10,000 different parts, you don’t want to record the names of each. Simply have *VoxFonts* read the names for you as part of your interactive database application.

Or maybe you develop voice systems. Your customer will eventually have all the prompts recorded from the script that you've been handed - but in the meantime, what do you do? You don't want to record all those message yourself (just to be thrown away later)! Have *VoxFonts* generate the audio files for you from the scripts provided, without having to call in expensive outside help.



# Function Overview

```

Microsoft Visual C++ - FNTTST.MAK <5> FNTTST.C
File Edit View Project Browse Debug Tools Options Window Help
sprintf
/*****
*/
/* VoxFonts Test: FntTst.c
*/
/* Copyright (c) 1987-1996 Andrew J. Michalik
*/
/*
*/
/* You have a royalty-free right to use, modify, reproduce and
*/
/* distribute the Sample Files (and/or any modified version) in any
*/
/* way you find useful, provided that you agree that VISI has no
*/
/* warranty obligations or liability for any Sample Application Files
*/
/* which are modified.
*/
/*****
*/
#if (defined (W31)) /*****
#include "windows.h" /* Windows SDK definitions
#endif /*****
#include "..\vxflib\voxfont.h" /* Voice Fonts support defs
#include <stdlib.h> /* Standard library defs
#include <string.h> /* String manipulation funcs
#include <stdio.h> /* Standard I/O
#include <io.h> /* Low level file I/O
#include <fcntl.h> /* Flags used in open/ sopen
#include <sys\types.h> /* File status and time types
#include <sys\stat.h> /* File status types
#include <time.h> /* Time and date functions
#include <conio.h> /* DOS low-level I/O
#include <stdarg.h> /* ANSI Std var length args
00020 088

```

VISI's *VoxFonts Software Development Kit (SDK)* lets you work in both the your MS Windows and MS DOS environments. *VoxFonts* use standard "C" style function calls. The following sections describe library functions you can use to add text-to-speech capability to your programs.

This listing is divided into two parts:

- ◆ ***VoxFonts Standard Functions:*** Functions that perform the text-to-speech conversion.
- ◆ ***DOS Replacement Functions:*** Functions that may be replaced by the user to make the library compatible with your programming environment. Modify these functions if you wish to use a DOS extender or other DOS memory enhancement tool.

## VoxFonts Standard Functions

These are the basic text-to-speech conversion functions provided by the *VoxFonts* libraries. Use these function as part of your regular programming to add text-to-speech capability to your application.

### Initialization

| <u>Functions</u> | <u>Use</u>  |
|------------------|---|
| VoxFntIni        | <i>VoxFonts</i> Initialize: Initialize support for the <i>VoxFonts</i> library. This function must be called before using any processing functions. |
| VoxFntTrm        | <i>VoxFonts</i> Terminate: Terminate support for the <i>VoxFonts</i> library. This function should be called prior to program termination.          |
| VoxFntVer        | <i>VoxFonts</i> Version: Return the <i>VoxFonts</i> library version number. This function may be called at any time.                                |

### Basic

#### Text-to-Speech

| <u>Functions</u> | <u>Use</u>   |
|------------------|--|
| VxFFntOpn        | <i>VoxFonts</i> Font Open: Open a “voice font” for subsequent use. Voice fonts are libraries of voice data that contain the rules and low-level voice data necessary for performing the text-to-speech conversion. |
| VxFTxtVox        | <i>VoxFonts</i> Text to Vox: Convert an ASCII text string to a digital audio file. Use this function for simple “one-step” conversion.   |
| VxFFntCls        | <i>VoxFonts</i> Font Close: Close a previously open “voice font” and release resources. Must be called prior to termination to release the allocated system resources.   |

### Advanced

#### Text-to-Speech

| <u>Functions</u> | <u>Use</u>  |
|------------------|---|
| VxFTxtWrd        | <i>VoxFonts</i> Text to Word: Convert an ASCII text string to a International Phonetic Alphabet (IPA) string. Use this function |

|           |   |
|-----------|---|
|           | as part of a more advanced program that can identify and exception-process individual words.  |
| VxFWrdIPA | <b>VoxFonts</b> Word to IPA: Convert an individual ASCII word string to an International Phonetic Alphabet (IPA) string. Use this function as part of a more advanced program that can identify and exception-process phonetic pronunciations.                      |
| VxFIPAVox | <b>VoxFonts</b> IPA to Vox: Convert an International Phonetic Alphabet (IPA) string into a digital audio file. Use this function as part of a more advanced program that can identify and exception-process phonetic pronunciations.                                |
| VxFRulMrg | <b>VoxFonts</b> Rule Merge: Merge a user-specified rule file with an open “voice font”. These files contain phonetic pronunciation rules as described earlier. Use a these files to add translation rules or specify pronunciations for difficult or foreign words. |

## File Types

**VoxFonts** supports two different basic file types: Pure files and Wave files.

- ◆ Pure files contain only the digitized sound.
- ◆ Wave files contain raw audio data plus additional format, control, and annotation information.

### File I/O

| <u>Function</u> | <u>Use</u> |
|-----------------|------------|
|-----------------|------------|

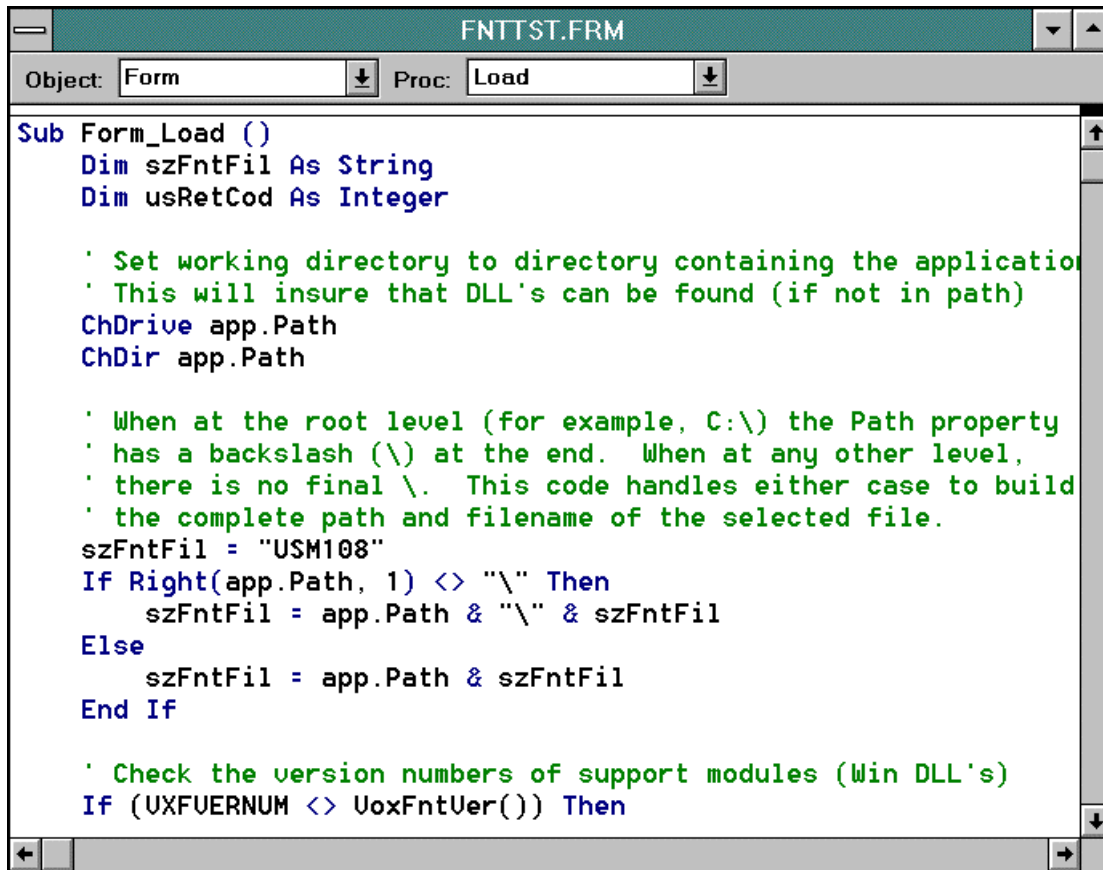
|           |   |
|-----------|---|
| VxFHdrOpn | <b>VoxFonts</b> Header Open: Opens an audio file that contains header information as well as raw audio information. A Multimedia Wave file is an example of this type of file. These files must be opened using this function in order to initialize the appropriate header block values. |
| VxFHdrCls | <b>VoxFonts</b> Header Close: Closes an audio file that contains header information as well as raw audio information. A Multimedia Wave file is an example of this type of file. These files must be closed using this function in order to set the appropriate header block values.      |

## ***VoxFonts* DOS Replacement Functions**

Most programmers will not need to use the DOS replacement functions. Some DOS-based high-performance interactive voice response systems and sophisticated game products do use specialized memory allocation routines. For those programmers, these functions may be replaced if using a DOS extender or memory enhancement product. Use the Microsoft “C” compiler “LIB” utility to replace these functions if you wish to use a DOS extender or other DOS memory enhancement tool.

| <b><u>Function</u></b> | <b><u>Use</u></b>                                  |
|------------------------|--|
| VISMemMax              | Return size of largest block of contiguous memory. |
| VISMemSiz              | Return size of designated memory block.            |
| VISMemAlo              | Allocate block of memory.                          |
| VISMemAloChg           | Change size of previously allocated memory.        |
| VISMemAloRel           | Release previously allocated memory.               |
| VISMemLck              | Lock allocated memory.                             |
| VISMemLckRel           | Unlock allocated memory.                           |

# Function Reference



```
Sub Form_Load ()
    Dim szFntFil As String
    Dim usRetCod As Integer

    ' Set working directory to directory containing the application
    ' This will insure that DLL's can be found (if not in path)
    ChDrive app.Path
    ChDir app.Path

    ' When at the root level (for example, C:\) the Path property
    ' has a backslash (\) at the end. When at any other level,
    ' there is no final \. This code handles either case to build
    ' the complete path and filename of the selected file.
    szFntFil = "USM108"
    If Right(app.Path, 1) <> "\" Then
        szFntFil = app.Path & "\" & szFntFil
    Else
        szFntFil = app.Path & szFntFil
    End If

    ' Check the version numbers of support modules (Win DLL's)
    If (UXFUERNUM <> UoxFntVer()) Then
```

## Overview

VISI's *VoxFonts Software Development Kit (SDK)* lets you work in both the your MS Windows and MS DOS environments. The *VoxFonts SDK* contains sample programs that use the libraries and DLLs. These programs were created using Visual Basic and Microsoft C version 8.0, and demonstrate usage under both Windows and DOS. The sample program performs a simple text-to-speech conversion in both .WAV and .VOX formats.

## The FntTst Program

The sample program performs a simple text-to-speech conversion. To build the Microsoft Visual C CvtTst programs:

- ◆ Make sure the Microsoft C 8.0 tools are installed and accessible.
- ◆ Run the batch file MakeMe.bat to build both MS Windows and MS DOS versions of the sample program.

## Visual Basic Interface

To build the Microsoft Visual Basic FntTst project:

- ◆ Make sure Microsoft Visual Basic 3.0 or above is installed and accessible.
- ◆ Use “File... Open Project” to open “MVBTTST\FNTTST.MAK”.
- ◆ Use “File... Make Exe File...” to build the sample program.

## VoxFonts SDK Functions

The following pages describe, in alphabetical order, the functions in the *VoxFonts SDK*. The discussion of each function is divided into the following sections:

- ◆ Description: Illustrates the function calling syntax and summarizes the routine’s effect.
- ◆ Parameter: Describes function arguments and purpose.
- ◆ Returns: Specifies the function return value, if any.
- ◆ Remarks: Provides more detailed information on side effects and usage.

## VoxFonts Standard Functions

These are the basic text-to-speech conversion functions provided by the *VoxFonts* libraries. Use these function as part of your regular programming to add text-to-speech capability to your application.

The following descriptions provide detailed information for calling these functions from your application. The definitions of these functions, structures and constants are defined in the files VOXFNT.H (for “C/C++” programmers) and VOXFNT.TXT (for Visual Basic programmers).

---

## VoxFntIni

**unsigned short pascal far VoxFntIni (unsigned short usReqTyp, unsigned long ulPrm001, unsigned long ulPrm002);**

*VoxFonts* Initialize: Initialize support for the *VoxFonts* library. This function must be called before using any processing functions.

| <b>Parameter</b> | <b>Description</b>                   |
|------------------|--------------------------------------|
| usReqTyp         | Reserved for future use. (Set to 0)  |
| ulPrm001         | Reserved for future use. (Set to 0L) |
| ulPrm002         | Reserved for future use. (Set to 0L) |

### **Returns**

0 if successful. The following pre-defined values can be used to determine the return code status:

| <u>Value</u> | <u>Meaning</u>                    |
|--------------|-----------------------------------|
| SI_VXFNO_ERR | Success                           |
| SI_VXFKEYERR | Key/License error, 100 word limit |
| SI_VXFVERERR | Module version mismatch           |

---

## VoxFntTrm

**unsigned short pascal far VoxFntTrm (void);**

*VoxFonts* Terminate: Terminate support for the *VoxFonts* library. This function should be called prior to program termination.

### **Parameters**

None

### **Returns**

0 if successful.

---

## VoxFntVer

**unsigned short pascal far VoxFntVer (void);**

*VoxFonts* Version: Return the *VoxFonts* library version number. This function may be called at any time.

### **Parameters**

None

### **Returns**

The version number as hexadecimal 0xAABB where AA equals the major version, and BB equals the minor version number.

## **VxFFntCls**

**FNTHDL pascal far VxFFntCls (FNTHDL fhFntHdl, void far \* lpRsv001);**

*VoxFonts* Font Close: Close a previously open “voice font” and release resources. Must be called prior to termination to release the allocated system resources.

| <b><u>Parameter</u></b> | <b><u>Description</u></b>              |
|-------------------------|--|
| fhFntHdl                | Previously opened font handle.         |
| lpRsv001                | Reserved for future use. (Set to NULL) |

### **Returns**

Zero if successful, original handle if failed.

## **VxFFntOpn**

**FNTHDL pascal far VxFFntOpn (char far \*szFntNam, void far \* lpRsv001);**

*VoxFonts* Font Open: Open a “voice font” for subsequent use. Voice fonts are libraries of voice data that contain the rules and low-level voice data necessary for performing the text-to-speech conversion.

| <b><u>Parameter</u></b> | <b><u>Description</u></b>              |
|-------------------------|--|
| szFntNam                | File name of font to open.             |
| lpRsv001                | Reserved for future use. (Set to NULL) |

### **Returns**

Non-zero font handle if successful.

---

## VxFHdrCls

**short pascal far VxFHdrCls (short sDstHdl, HDRHDL mhHdrHdl);**

*VoxFonts* Header Close: Closes an audio file that contains header information as well as raw audio information. A Multimedia Wave file is an example of this type of file. These files must be closed using this function in order to set the appropriate header block values.

| <u>Parameter</u> | <u>Description</u>                    |
|------------------|---------------------------------------|
| sDstHdl          | Destination file handle.              |
| dtDigTyp         | Digitizer output type constant.       |
| mhHdrHdl         | Previously file header memory handle. |

### Returns

0 if successful.

---

## VxFHdrOpn

**short pascal far VxFHdrOpn (short sDstHdl, DIGTYP dtDigTyp, HDRHDL far \*lpHdrHdl);**

*VoxFonts* Header Open: Opens an audio file that contains header information as well as raw audio information. A Multimedia Wave file is an example of this type of file. These files must be opened using this function in order to initialize the appropriate header block values.

| <u>Parameter</u> | <u>Description</u>                    |
|------------------|---------------------------------------|
| sDstHdl          | Destination file handle.              |
| dtDigTyp         | Digitizer output type constant.       |
| lpHdrHdl         | Pointer to file header memory handle. |

### Returns

0 if successful.

---

## VxFIPAVox

**char far \* pascal far VxFIPAVox (FNTHDL fhFntHdl, char far \*szIPATxt, short sDstHdl, DIGTYP dtDigTyp, void far \* lpRsv001);**

**VoxFonts** IPA to Vox: Convert an International Phonetic Alphabet (IPA) string into a digital audio file. Use this function as part of a more advanced program that can identify and exception-process phonetic pronunciations.

| <b>Parameter</b> | <b>Description</b>  |              |                |        |                              |        |                              |        |                              |        |                              |           |                                      |           |                                       |
|------------------|---|--------------|----------------|--------|------------------------------|--------|------------------------------|--------|------------------------------|--------|------------------------------|-----------|--------------------------------------|-----------|---------------------------------------|
| fhFntHdl         | Previously opened font handle.  |              |                |        |                              |        |                              |        |                              |        |                              |           |                                      |           |                                       |
| szIPATxt         | IPA text to convert.  |              |                |        |                              |        |                              |        |                              |        |                              |           |                                      |           |                                       |
| sDstHdl          | Destination file handle.  |              |                |        |                              |        |                              |        |                              |        |                              |           |                                      |           |                                       |
| dtDigTyp         | Digitizer output type constant.   |              |                |        |                              |        |                              |        |                              |        |                              |           |                                      |           |                                       |
|                  | <table border="1"> <thead> <tr> <th><u>Value</u></th> <th><u>Meaning</u></th> </tr> </thead> <tbody> <tr> <td>DLG24K</td> <td>Dialogic 24 kilobit/ second.</td> </tr> <tr> <td>DLG32K</td> <td>Dialogic 32 kilobit/ second.</td> </tr> <tr> <td>DLG48K</td> <td>Dialogic 48 kilobit/ second.</td> </tr> <tr> <td>DLG64K</td> <td>Dialogic 64 kilobit/ second.</td> </tr> <tr> <td>WAVDIG111</td> <td>Wave Multimedia 8 bit at 11.025 kHz.</td> </tr> <tr> <td>WAVDIG112</td> <td>Wave Multimedia 16 bit at 11.025 kHz.</td> </tr> </tbody> </table> | <u>Value</u> | <u>Meaning</u> | DLG24K | Dialogic 24 kilobit/ second. | DLG32K | Dialogic 32 kilobit/ second. | DLG48K | Dialogic 48 kilobit/ second. | DLG64K | Dialogic 64 kilobit/ second. | WAVDIG111 | Wave Multimedia 8 bit at 11.025 kHz. | WAVDIG112 | Wave Multimedia 16 bit at 11.025 kHz. |
| <u>Value</u>     | <u>Meaning</u>  |              |                |        |                              |        |                              |        |                              |        |                              |           |                                      |           |                                       |
| DLG24K           | Dialogic 24 kilobit/ second.  |              |                |        |                              |        |                              |        |                              |        |                              |           |                                      |           |                                       |
| DLG32K           | Dialogic 32 kilobit/ second.  |              |                |        |                              |        |                              |        |                              |        |                              |           |                                      |           |                                       |
| DLG48K           | Dialogic 48 kilobit/ second.  |              |                |        |                              |        |                              |        |                              |        |                              |           |                                      |           |                                       |
| DLG64K           | Dialogic 64 kilobit/ second.  |              |                |        |                              |        |                              |        |                              |        |                              |           |                                      |           |                                       |
| WAVDIG111        | Wave Multimedia 8 bit at 11.025 kHz.  |              |                |        |                              |        |                              |        |                              |        |                              |           |                                      |           |                                       |
| WAVDIG112        | Wave Multimedia 16 bit at 11.025 kHz.   |              |                |        |                              |        |                              |        |                              |        |                              |           |                                      |           |                                       |
| lpRsv001         | Set to zero for future compatability.   |              |                |        |                              |        |                              |        |                              |        |                              |           |                                      |           |                                       |

### **Returns**

Pointer to first unconverted IPA character. Normally points to string terminator.

## **VxFRuIMrg**

**short pascal far VxFRuIMrg (FNTHDL fhFntHdl, char far \*sSrcHdl, void far \* lpRsv001);**

**VoxFonts** Rule Merge: Merge a user-specified rule file with an open “voice font”. These files contain phonetic pronunciation rules as described earlier. Use these files to add translation rules or specify pronunciations for difficult or foreign words.

| <b>Parameter</b> | <b>Description</b>                     |
|------------------|--|
| fhFntHdl         | Previously opened font handle.         |
| lpRsv001         | Reserved for future use. (Set to NULL) |

**Returns**

0 if successful.

**VxFTxtWrd**

**char far \* pascal far VxFTxtWrd (FNTHDL fhFntHdl, char far \*szInpStr, char far \*szIPATxt, unsigned short usOutMax, void far \*lpRsv001);**

*VoxFonts* Text to Word: Convert an ASCII text string to a International Phonetic Alphabet (IPA) string. Use this function as part of a more advanced program that can identify and exception-process individual words.

| <b>Parameter</b> | <b>Description</b> |
|------------------|--------------------|
|------------------|--------------------|

|          |   |
|----------|---|
| fhFntHdl | Previously opened font handle.                                    |
| szIPATxt | Destination buffer for IPA text.                                  |
| usOutMax | Size of destination IPA buffer (including termination character). |
| lpRsv001 | Reserved for future use. (Set to NULL)                            |

**Returns**

Pointer to first unconverted IPA character. Normally points to string terminator.

**VxFWrdIPA**

**char far \* pascal far VxFWrdIPA (FNTHDL fhFntHdl, char far \*szInpStr, char far \*szIPATxt, unsigned short usOutMax, void far \*lpRsv001);**

*VoxFonts* Word to IPA: Convert an individual ASCII word string to a International Phonetic Alphabet (IPA) string. Use this function as part of a more advanced program that can identify and exception-process phonetic pronunciations.

| <b>Parameter</b> | <b>Description</b> |
|------------------|--------------------|
|------------------|--------------------|

|          |   |
|----------|---|
| fhFntHdl | Previously opened font handle.                                    |
| szIPATxt | Destination buffer for IPA text.                                  |
| usOutMax | Size of destination IPA buffer (including termination character). |
| lpRsv001 | Set to zero for future compatability.                             |

**Returns**

Pointer to first unconverted IPA character. Normally points to string terminator.

**VxFTxtVox**

**char far \* pascal far VxFTxtVox (FNTHDL fhFntHdl, char far \*szInpTxt, short sDstHdl, DIGTYP dtDigTyp, void far \*lpRsv001);**

*VoxFonts* Text to Vox: Convert an ASCII text string to a digital audio file. Use this function for simple “one-step” conversion.

| <b>Parameter</b> | <b>Description</b> |
|------------------|--------------------|
|------------------|--------------------|

|          |                                   |
|----------|-----------------------------------|
| fhFntHdl | Previously opened font handle.    |
| szInpTxt | Natural language text to convert. |
| sDstHdl  | Destination file handle.          |
| dtDigTyp | Digitizer output type constant.   |

| <u>Value</u> | <u>Meaning</u> |
|--------------|----------------|
|--------------|----------------|

|           |                                       |
|-----------|---------------------------------------|
| DLG24K    | Dialogic 24 kilobit/ second.          |
| DLG32K    | Dialogic 32 kilobit/ second.          |
| DLG484K   | Dialogic 48 kilobit/ second.          |
| DLG64K    | Dialogic 64 kilobit/ second.          |
| WAVDIG111 | Wave Multimedia 8 bit at 11.025 kHz.  |
| WAVDIG112 | Wave Multimedia 16 bit at 11.025 kHz. |

|          |                                       |
|----------|---------------------------------------|
| lpRsv001 | Set to zero for future compatability. |
|----------|---------------------------------------|

**Returns**

Pointer to first unconverted natural language text. Normally points to string terminator.

**VoxFonts DOS Replacement Functions**

Most programmers will not need to use the DOS replacement functions. Some DOS-based high-performance interactive voice response systems and game products do use specialized memory allocation routines. For those programmers, these functions may be replaced if using a DOS extender or memory enhancement

product. Use the Microsoft “C” compiler “LIB” utility to replace these functions if you wish to use a DOS extender or other DOS memory enhancement tool.

The following descriptions provide detailed information for calling these functions from your application. The definitions of these functions, structures and constants are defined in the files USRSTB.H (for “C/C++” programmers).

## VISMemAlo

**VXFMEMHDL cdecl far VISMemAlo (unsigned long ulReqSiz);**

VxF Memory Allocate: Allocate block of memory.

| <u>Parameter</u> | <u>Description</u> |
|------------------|--------------------|
|------------------|--------------------|

|          |  |
|----------|--|
| ulReqSiz | Size of requested memory block in bytes. |
|----------|--|

### Returns

|          |                                       |
|----------|---------------------------------------|
| mhMemHdl | Non-zero memory handle if successful. |
| 0        | Failure to allocate memory.           |

## VISMemAloChg

**VXFMEMHDL cdecl far VISMemAloChg (VXFMEMHDL, unsigned long ulReqSiz);**

VxF Memory Allocation Change: Change size of previously allocated memory.

| <u>Parameter</u> | <u>Description</u> |
|------------------|--------------------|
|------------------|--------------------|

|          |                                    |
|----------|------------------------------------|
| mhMemHdl | Original memory handle.            |
| ulReqSiz | New size of memory block in bytes. |

### Returns

|          |                                       |
|----------|---------------------------------------|
| mhMemHdl | Non-zero memory handle if successful. |
| 0        | Failure to re-allocate memory.        |

## VISMemAloRel

**VXFMEMHDL cdecl far VISMemAloRel (VXFMEMHDL mhMemHdl);**

VxF Memory Allocation Release: Release previously allocated memory.

| <u>Parameter</u>      | <u>Description</u>               |
|-----------------------|----------------------------------|
| mhMemHdl              | Handle of memory to be released. |
| <b><u>Returns</u></b> |                                  |
| 0                     | Memory released.                 |
| mhMemHdl              | Memory in use by other process.  |

## VISMemLck

**void far \* cdecl far VISMemLck (VXFMEMHDL mhMemHdl);**

VxF Memory Lock: Lock allocated memory.

| <u>Parameter</u> | <u>Description</u>             |
|------------------|--------------------------------|
| mhMemHdl         | Handle of memory to be locked. |

### **Returns**

Non-zero pointer to locked memory if successful, zero if failed.

## VISMemLckRel

**VXFMEMHDL cdecl far VISMemLckRel (VXFMEMHDL mhMemHdl);**

VxF Memory Lock Release: Unlock allocated memory.

| <u>Parameter</u> | <u>Description</u>               |
|------------------|----------------------------------|
| mhMemHdl         | Handle of memory to be unlocked. |

### **Returns**

|          |                             |
|----------|-----------------------------|
| mhMemHdl | Failure to unlock memory.   |
| 0        | Memory successful unlocked. |

## VISMemMax

**unsigned long cdecl far VISMemMax (void);**

VxF Memory Maximum: Return size of largest block of contiguous memory.

| <u>Parameter</u> | <u>Description</u> |
|------------------|--------------------|
|------------------|--------------------|

none

**Returns**

Size of memory in bytes.

---

**VISMemSiz**

**unsigned long cdecl far VISMemSiz (VXFMEMHDL mhMemHdl);**

VxF Memory Size: Return size of designated memory block.

**Parameter      Description**

none

**Returns**

Size of memory in bytes.



# ◆ Appendix

Appendix A, *Setup and Installation* tells you how to set up *VoxFonts* on your computer. This chapter covers system requirements, and prepares you for a sample editing session.

Appendix B, *Troubleshooting & Recovery*, contains information to help diagnose and correct problems if something unexpected occurs.

Appendix C, *Licensing*, lists options for licensing and using this software package.



# Installation & Setup

This program will install VoxFonts Version 3.0 on your computer system and verify the integrity of the distribution disk(s). You may press the [Esc] key at any time to abort the installation. INSTALL will ask you several questions about your computer and then give you the option of installing VoxFonts for your system.

Each question has a default answer. If the default answer is correct, press the ENTER key in response to the question. Otherwise, type the answer and then press the ENTER key.

If you make a mistake while typing, just press the BACKSPACE key and then retype the answer.

Press [Esc] to quit, any other key to continue . . .

Initial Install program screen

This section provides basic instructions for installing *VoxFonts*<sup>™</sup>. You must use the *VoxFonts* Setup program to install *VoxFonts*. The files on the program disks are compressed; you can't just copy them to your hard drive.

## *VoxFonts* Software Installation

Insert the diskette labeled *VoxFonts SDK* in the A: drive. Change to the A: drive by entering:

```
A: <ENTER>
```

To install *VoxFonts* on your system, enter the following:

```
INSTALL <ENTER>
```

*VoxFonts SDK* supports many popular data formats. The install program, *INSTALL*, will ask you to specify the data format you use. Just follow the instructions on the screen!

After running *INSTALL*, store your original *VoxFonts SDK* disks in a safe place. You're now ready to work with the *VoxFonts SDK*.

## The *VoxFonts SDK* Directories

The *VoxFonts SDK* directories contains all the files necessary to convert your files to and from a variety of audio file formats. In addition, the distribution diskette contains audio files you can use for sample conversions. The *VoxFonts* installation directory is organized into subdirectories. The following briefly describes the files and their uses:

- ◆ The MSCTst directory contains a sample MS Windows and DOS program “FNTTST.C”. To build both the Windows and DOS versions, run the batch file “MAKEME.BAT”. The associated makefile assumes that Microsoft C 8.0 tools are installed and accessible.
- ◆ The VxFLib directory contains the Visual Basic header file along with the “C” header and library files for various environments and memory models. MME is for Microsoft Medium memory model with floating point emulator, MLE for Microsoft Large memory model with floating point emulator, and MWE for Microsoft Windows model with floating point emulator.
- ◆ The MVBTst directory contains a sample MS Windows Visual Basic project, “MVBTST.MAK”.
- ◆ The main directory contains the DLLs necessary for the MS Windows executable versions.

After installing *VoxFonts* store your original disks in a safe place. You’re now ready to run *VoxFonts*.

# Troubleshooting & Recovery



## System Configuration

*VoxFonts*<sup>™</sup> is tested using a wide variety of system configurations (both hardware and software), and every attempt is made to ensure that *VoxFonts* will work correctly with popular displays, TSR's, memory boards, and related products.

If you experience difficulty in running *VoxFonts*, you may want to double-check your system configuration. Because of the number and variety of add-on devices and software packages, it is possible to get side effects resulting from the interaction of these entities.

The most unusual errors occur as a result of conflicts between the audio board and the hardware/software interrupts of other system components. Double (and triple)

check to ensure that the audio boards and drivers are not contending for these important resources. Peculiar and bizarre system behaviors have been reported when this occurs—and is often not detected by diagnostic software because of the intermittent nature of the device conflicts.

Quickest results are obtained by removing any extraneous boards, TSR's, and device drivers. Double check your system documentation to assure that the system boards do not conflict with any other audio hardware interrupts. This usually solves the problem. If the problem persists, please provide us with as much information as possible so that we can reproduce and help you eliminate the problem.

## Error and Warning Messages

### Error Messages

- ◆ (VxFFntOpn) Error: Cannot allocate font memory. Cannot allocate sufficient memory for font storage.
- ◆ (VxFFntOpn) Error: Cannot lock font memory. Cannot allocate and lock sufficient memory for font storage.
- ◆ (VxFFntOpn) Error: Cannot load rule table. Cannot load the font rule file. Check your path and directory settings.
- ◆ (VxFRulMrg) Error: Cannot load merge table. Cannot load the pronunciation rule merge file. Check your path and directory settings.
- ◆ (VxFFntOpn) Error: Cannot load vocters “font file”. Cannot load the font “voice characters” (vocters) file. Check your path and directory settings.
- ◆ (VxFFntOpn) Error: Too many fonts already open. Insufficient resources available to open additional fonts. Close a previously opened font before opening the new one.
- ◆ (VxFFntOpn) Error: Cannot load vocters “font file”. Cannot load low level “voice characters” (vocters) from the specified file.
- ◆ (SelVocIdx) Error: Bad vocter index [rule#]. The requested “voice characters” (vocters) does not have a valid index. Check your user font file and rule listing.
- ◆ (GetVocIdx) Error: Bad vocter \vocter string\. The low level “voice character” (vocter) is out of range.

- ◆ (OpnFntVoc) Error: Cannot open font file “font file”: Cannot perform file open. Check your path and directory settings.

## Warning Messages

- ◆ (GetTtPTab) Warning: Bad rule \rule string\. The text-to-phoneme convertor encountered a bad rule.

## Technical Support

As easy as our products are to use, there are times when you just can't manage on your own. Or maybe you don't have the time to search the manual for answers. Or perhaps you just have a general problem that's frustrating you.

That's why we offer comprehensive support for all of our products, as well as any voice or telephony problems that may arise. We offer help in several ways, so you can choose the plan that's right for you. Let our extensive knowledge of this technology and troubleshooting expertise work for you.

VISI offers a number of comprehensive technical support options for users. For more information, call 1-310-392-8780. VISI support services are subject to VISI prices, terms, and conditions in place at the time the service is used.

### When you call...

When you call, you should be at your computer and have the appropriate product documentation at hand. Be prepared to give the following information:

- ◆ The product name and version number of the VISI product you are using
- ◆ The type of hardware you are using
- ◆ The exact wording of any messages that appeared on your screen.
- ◆ What happened and what you were doing when the problem occurred.
- ◆ How you tried to solve the problem.

### Setup and Installation

For setup and installation assistance with VISI products, dial 1-310-392-6266. This service is designed to get you up and running quickly with our products.

## How-to and in-depth support

For programming “how to” questions, user training, and support for non-VISI products, dial our credit card line at 1-800-469-4874. Please have your Visa, MasterCard, or American Express card ready.

## Download Service

Access technical notes and supplementary files covering common product support issues from our website at [www.vinfo.com](http://www.vinfo.com) and via modem at 310-392-6610. These services are available 24 hours a day, seven days a week.

## Free and Paid Support

Our free technical support covers setup, installation, bug reporting and product menu usage. Our free technical support does not cover programming “how to” questions, user training, and support for non-VISI products.

Why not? The time required to answer these questions depends on the customer’s technical background and specific needs. These factors vary dramatically from customer to customer, and there is no way for us to charge for these services at a fixed rate.

The solution? We provide paid consulting services for customers that need help with programming “how to” questions, user training and support for non-VISI products.

And for your protection, all of our consulting contracts are pre-paid, so we will not perform any work until authorized by you.

VISI offers a number of comprehensive technical support options for users. For more information, call 1-310-392-8780. VISI support services are subject to VISI prices, terms, and conditions in place at the time the service is used.





# Licensing



## Licensing

The following sections summarize licensing for *VoxFonts*<sup>™</sup>. Please contact VISI for additional product pricing and licensing information. Licenses are subject to VISI prices, terms, and conditions in place at the time the product is purchased.

### The VOXFNT.LIC License File

Your license and registration information is contained in a file called “VoxFnt.Lic”. This file must be placed in the same directory as your executable file (DOS), or in the same directory as the *VoxFonts* DLLs (Windows).

This file entitles you to one *VoxFonts* Single Application Environment License as described below.

### Single Application Environment Distribution License

This allows distribution of the *VoxFonts* software with a program that is distributed by your company or government agency. Private labeling is available.

The “PRODUCT” shall herein refer exclusively to the Voice Information Systems *VoxFonts* version 3.XX (where XX is any number between 00 and 99) Single Application Environment Files specified herein. This Agreement applies only to this specific version of the PRODUCT distributed by Voice Information Systems, and shall not be construed to require VISI to deliver any subsequent or future versions. The Product consists only of the following files which can be found in the retail edition of Voice Information Systems *VoxFonts*™ version 3.XX for the DOS and MS-Windows™ operating systems:

USM108.VOC  
USM108.RUL  
DLGVXF.DLL  
DLGPCM.DLL  
VOXFNT.LIC

VISI grants to Licensee a royalty-free, non-exclusive, non-transferable, worldwide license, during the term of this Agreement, and subject to the terms and conditions of this Agreement, to reproduce and distribute copies of the PRODUCT, PROVIDED that the PRODUCT is used, modified, reproduced and distributed only in conjunction with, linked to, incorporated into, and as a part of Licensee’s Program. This Agreement limits the Licensee to distribution of the files listed above, a single Licensee’s Program, and one or more related subprograms provided by the Licensee. Related subprograms are those programs provided by the Licensee that are included in the same package or stock-keeping unit as Licensee’s Program, and are not distributed separately from Licensee’s Program during the term of this Agreement.

Licensee represents and warrants the following: Licensee’s Program(s) may load and run, through either a menu or a programmatic interface (i.e., through macros, DDE, etc.), USM108.Voc, USM108.Rul, DlgVxF.DLL, DlgPCM.DLL, VoxFnt.Lic. If Licensee’s Program(s) allows the creation of an application, it shall include documentation that clearly states that any application developer using Licensee’s Program(s) to create an application must obtain a license agreement from VISI in order to distribute any VISI files or functionality included in the Licensee’s Program(s).

This Agreement does not grant Licensee the right to use, reproduce, or distribute the PRODUCT by itself or in any other form or manner except as expressly authorized above. Licensee acknowledges that the PROGRAM is confidential and is a trade secret belonging exclusively to VISI. Neither Licensee nor any employee or agent thereof shall reverse compile or disassemble the PRODUCT, and Licensee

shall take such steps as may be reasonably necessary or prudent to insure that no employee or agent engages in such conduct.

Distributing, repackaging, or reselling of the software to third parties is not allowed. All licenses are prepaid.

Licensee's rights and obligations under this Agreement may not be assigned, sublicensed, or otherwise transferred except by the written consent of VISI. All rights not expressly granted herein are reserved by VISI.



# ◆ Glossary

**adaptive delta pulse code modulation (ADPCM):** An audio compression algorithm for digital audio based on describing level differences between adjacent samples.

**aliasing:** Undesired frequencies that are produced when harmonic components of an audio signal being sampled by a digital recording device (or harmonic components generated within a digital sound source) lie above the Nyquist frequency. Aliasing differs from some other types of noise in that its pitch changes radically when the pitch of the intended sound changes.

**amplitude:** The amount of a signal. Amplitude is measured by determining the amount of fluctuation in air pressure (of a sound), voltage (of an electrical signal), or numerical data (in a digital application). When the signal is in the audio range, amplitude is perceived as loudness.

**analog:** Of, relating to, or being a mechanism in which data is represented by continuously variable physical quantities

**analog-to-digital converter (ADC):** A device that changes the continuous fluctuations in voltage from an analog device (such as a microphone) into digital information that can be stored or processed in a sampler, digital signal processor, or digital recording device.

**application:** A sub-program in the Windows environment, such as *VFEdit*<sup>®</sup>, or Microsoft Word for Windows.

**attenuate:** To reduce the level of a signal.

**b:** An abbreviation for byte.

**byte:** A group of eight binary digits processed as a unit by a computer and used especially to represent an alpha-numeric character; also see word.

**CCITT:** Consultative Committee on International Telephone and Telegraph. The CCITT was an international standards committee now known as the ITU.

**C-ITU:** Committee of the International Telecommunication Union. The International Telecommunication Union (ITU) is group that sets international communications standards. They sponsor committees to provide detailed technical specifications for those standards.

**companding:** A type of signal processing in which the signal is compressed on input and expanded back to its original form on output. Digital companding enables a device to achieve a greater apparent dynamic range with fewer bits per sample word.

**default:** A selection automatically used by a computer program in the absence of a choice made by the user.

**Diallogic:** Trade name of a telephony board manufacturer.

**digital:** Relating to an audio recording method in which sound waves are represented digitally (as on magnetic disk) so that in recordings wow and flutter are eliminated and background noise is reduced; also see analog. (Informal definition: Digital sound is what you get when you electronically chop up a continuous analog sound.)

**digital-to-analog converter (DAC):** A device that changes the sample words put out by a digital audio device into analog fluctuations in voltage that can be sent to a mixer or amplifier. All digital synthesizers, samplers, and effects devices have DAC at their outputs to create audio signals.

**digital signal processing:** Broadly speaking, all changes in sound that are produced within a digital audio device, other than changes caused by simple cutting and pasting of sections of a waveform, are created through DSP. A digital echo is a typical DSP function.

**DSP:** See digital signal Processing.

**file format:** A description of the disk file format or contents; also see pure, indexed, wave.

**file type:** A description of the disk file format or contents; also see pure, indexed, wave.

**filter:** A device for eliminating selected frequencies from the sound spectrum of a signal and perhaps (in the case of a resonant filter) increasing the level of other frequencies.

**formant:** A resonant peak in a frequency spectrum. For example, the variable formants produced by the human vocal tract are what give vowels their characteristic sounds.

**G:** for giga, i.e., a billion, as in 1 GHz = 1,000,000,000 ( $10^9$ ) hertz, or Gb, a billion bytes; [both 'g' in giga are pronounced as in 'go', *not* as in jelly].

**direct-to-disk recording:** A computer-based form of tapeless recording in which incoming audio is converted into digital data and stored on a hard disk.

**harmonic:** A frequency that is a whole-number multiple of the fundamental frequency. For example, if the fundamental frequency of a sound is 440 Hz, the first two harmonics are 880 Hz and 1,320 Hz (1.32 kHz). See overtone; also see inharmonic.

**hertz:** A unit of frequency equal to one cycle per second. Named for Heinrich R. Hertz who did the original scientific investigations.

**high-pass filter:** A filter that attenuates the frequencies below its cutoff frequency.

**Hz:** abbreviation for hertz

**interactive voice response (IVR):** Computer systems that answer (or dial) a phone, play or record a message, often under control of the user's touch-tone phone. A voice mail system is an example of a basic IVR system.

**ITU:** International Telecommunication Union. A group that sets international communications standards. The ITU was previously known as the CCITT.

**k:** abbreviation for kilo (thousand): 1,000 or  $10^3$ , as in 1 kHz.

**m:** abbreviation for milli (one thousandth): 1/1000 or  $10^{-3}$ , as in 1 mW (0.001 W).

**M:** abbreviation for mega (million): 1,000,000 or  $10^6$ , as in 1 MHz.

**Nyquist frequency:** The highest frequency that can be reproduced accurately when a signal is digitally encoded at a given sample rate. Theoretically, the Nyquist frequency is half of the sampling rate. For example, when a digital recording uses a sampling rate of 11 kHz, the Nyquist frequency is 5.5 kHz. If a signal being sampled contains frequency components that are above the Nyquist limit, aliasing will be introduced in the digital representation of the signal unless those frequencies are filtered out prior to digital encoding; see aliasing, low-pass filter.

**OEM:** Original equipment manufacturer. A company that produces an end product as opposed to a sub-assembly.

**pole:** A portion of a filter circuit. The more poles a filter has, the more abrupt its cutoff slope will be. Each pole causes a slope of 6 dB per octave; typical filter configurations are two-pole (12 dB/oct) and four-pole (24 dB/oct). See rolloff.

**pot:** See potentiometer.

**prompts:** Common term for outgoing messages recorded for your voice messaging system. These messages typically “prompt” the user to press a key for a specific person or service.

**pure file:** Files containing only the digitized sound.

**quantization noise:** One of the types of errors introduced into an analog audio signal by encoding it in digital form. The digital equivalent of tape hiss, quantization noise is caused by the small differences between the actual amplitudes of the points being sampled and the bit resolution of the analog-to-digital converter.

**quantized:** Set up to produce an output in discrete steps.

**SDK:** Software development kit.

**sine wave:** A signal put out by an oscillator in which the voltage, or equivalent, rises and falls smoothly and symmetrically, following the trigonometric formula for the sine function. Sub-audio sine waves are used to modulate other waveforms to produce vibrato and tremolo. Audio-range sine waves contain only the fundamental frequency, with no overtones, and thus can form the building blocks for more complex sounds.

**sound card:** Add-on hardware typically placed inside the computer and used to record and play digital audio. These devices usually interface to standard audio equipment such as microphones and speakers.

**telephony card:** Add-on hardware typically placed inside the computer and used to record and play digital audio. These devices usually interface to standard telephone lines and often can handle multiple phone lines per board; also see voice board.

**VISI:** Voice Information Systems, Inc. The really great company that makes this product.

**voice board:** Add-on hardware typically placed inside the computer and used to record and play digital audio. These device usually interface to standard telephone lines and often can handle multiple phone lines per board; also see telephony card.

**vox:** A file extension specifying a generic telephony audio format. Typically encountered as FILENAME.VOX

**wav:** A file extension specifying an industry standard multimedia audio format. Typically encountered as FILENAME.WAV

**wave:** An industry standard file format that contains raw audio data plus additional format, control, and annotation information.

**waveform:** A signal, either sampled (digitally recorded) or periodic, being generated by an oscillator. Also, the graphic representation of this signal, as on a computer screen. Each waveform has its own unique harmonic content. See oscillator.

**word:** A number of bytes processed as a unit and conveying a quantum of information in communication and computer work. Also, a single number (sampled word) that represents the instantaneous amplitude of a sampled sound at a particular moment. In 8-bit recording, a sample word contains one byte; in 16-bit recording, each word is a two-byte number.

# ◆ Index

## —A—

adaptive delta pulse code modulation (ADPCM), 59  
 aliasing, 59  
 Allophones, 18  
 amplitude, 59  
 analog, 59  
 analog-to-digital converter (ADC), 59  
 application, 59  
 attenuate, 59

## —B—

b, 59  
 binary code, 23  
 binary format, 24  
*bits*, 23  
 byte, 59

## —C—

CCITT, 60  
 C-ITU, 60  
 Coarticulation, 19  
 Command Buttons, Sample Program, 13  
 Command Line Options, Sample Program, 11  
 companding, 60

Configuration, System, 49

## —D—

DAC, 23  
 default, 60  
 Dialogic, 60  
 digital, 60  
 digital signal processing, 60  
 digital to analog converter, 23  
 digital-to-analog, 24  
 digital-to-analog converter, 60  
 direct-to-disk recording, 61  
 DOS, 6  
 DOS, Sample Program, 10  
 DOSTst, 11  
 DSP, 60

## —E—

Error and Warning Messages, 50

## —F—

Features, 7  
 file format, 60  
 File I/O Functions, 31  
 File Name, Sample Program, 11  
 file type, 60  
 File Types, 31

filter, 61  
FntTst Program, 34  
formant, 61

## —G—

G, 61

## —H—

Hardware Requirements, 5  
harmonic, 61  
hertz, 61  
high-pass filter, 61  
Hz, 61

## —I—

IBM PC, 5  
input signal, 24  
Installation, 47  
Installation Directories, 48  
interactive voice response, 61  
International Phonetic Alphabet, 19  
IPA, 19  
ITU, 61

## —K—

k, 61

## —L—

License File, 55  
Licensing, 55  
low-pass filter, 24

## —M—

m, 61  
Math Coprocessor, 5  
Multimedia Player, Sample Program, 13

## —N—

Notational Conventions, 6  
Nyquist frequency, 61

## —O—

OEM, 62

## —P—

Phonemes, 17  
Phonetic Translation, Sample Program,  
11  
pole, 62  
pot, 62  
Program Name, Sample Program, 11, 13  
prompts, 62  
Pronunciation Rules, 20  
pure file, 62  
pure files, 31

## —Q—

quantization noise, 62  
quantized, 62

## —S—

sampled voltage, 24  
SDK, 62  
sine wave, 62  
Software Requirements, 6

sound card, 62  
switching noise, 24

## —T—

Technical Capabilities, 7  
Technical Support, 51  
Text String, Sample Program, 13  
Timing Message, Sample Program, 11

## —V—

VISI, 63  
voice board, 63

vox, 63  
VOXFNT.LIC, 55

## —W—

wav, 63  
wave, 63  
waveform, 63  
Windows 3.x, 6  
Windows 95, 6  
Windows NT, 6  
Windows, Sample Program, 12  
word, 63