Libraries

COLLABORATORS

	TITLE : Libraries	

ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		March 28, 2025	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

Libraries

1.1 Amiga® RKM Libraries: 36 Translator Library

This chapter describes the translator library which, together with the narrator device, provides the Amiga's text-to-speech capability. To fully understand how speech is produced on the Amiga, you should also read the "Narrator Device" chapter of the Amiga ROM Kernel Reference Manual: Devices.

The translator library provides a single function, Translate(), that converts an English language string into a phonetic string. You may then pass this phonetic string to the narrator device which will say the string using the Amiga's audio hardware. The two subsystems may also be used individually. You don't have to use the narrator to say the phonetic strings; you could use them instead for phonetic analysis or some other special purpose.

1.2 36 Translator Library / Opening the Translator Library

To use the Translate() function, you must first open the translator library. Setting a global variable, TranslatorBase, to the value returned from the call to OpenLibrary() enables the Amiga linker to correctly locate the translator library:

```
    struct Library *TranslatorBase;

    TranslatorBase = OpenLibrary("translator.library",REVISION);
    if(TranslatorBase != NULL)
        {
        /* use translator here -- library open */
        }


    LIBS: must contain translator.library.
    ---------------------------------------
```

```
    Since translator is a disk-based library, the call to
    OpenLibrary() will work only if the LIBS: directory contains
    translator.library.
```

1.3 36 Translator Library / Using the Translate Function

Once the library is open, you can call the translate function:

```
    #define BUFLEN 500

    STRPTR EnglStr;                 /* pointer to sample input string */
    LONG EnglLen;                   /* input length */
    UBYTE PhonBuffer[BUFLEN];       /* place to put the translation */
    LONG rtnCode;                   /* return code from function */

    EnglStr = "This is Amiga speaking.";    /* a test string */
    EnglLen = strlen(EnglStr);
    rtnCode = Translate(EnglStr, EnglLen, (STRPTR)&PhonBuffer[0], BUFLEN);
```

The input string will be translated into its phonetic equivalent and can be used to feed the narrator device. If you receive a non-zero return code, you haven't provided enough output buffer space to hold the entire translation. In this case, the Translate() function breaks the translation at the end of a word in the input stream and returns the position in the input stream at which the translation ended. You can use the output buffer, then call the Translate() function again, starting at this original ending position, to continue the translation where you left off. This method will sound smoothest if the ending position ends on sentence boundaries.

```
    Translate() returns negative values.
    ------------------------------------
    To get the proper character offset, you must use -(rtnCode) as
    the starting point for a new translation.
```

1.4 36 Translator Library / Closing the Translator Library

As with all other libraries of functions, if you have successfully opened the translator library for use, be sure to close it before your program exits. If the system needs memory resources, it can then expunge closed libraries to gain additional memory space:

```
    struct Library *TranslatorBase;

    if(TranslatorBase) CloseLibrary(TranslatorBase);
```

1.5 36 Translator Library / Additional Notes About Translate

The English language has many words that do not sound the same as they are spelled. The translator library has exception rules that it consults as the translation progresses. It also provides for common abbreviations such as Dr., Prof., LB., etc. Words that are not in the exception table are translated literally. This translation allows unrestricted English text as input, and uses over four hundred and fifty context sensitive rules. It automatically accents content words, and leaves function words (e.g. of, by, the, and at) unaccented. It is possible, however, that certain words will not translate well. You can improve the quality of translation by handling those words on your own.

The phoneme table that the narrator uses is listed in the "Narrator Device" chapter of the Amiga ROM Kernel Reference Manual: Devices. You will also find other important information about the Amiga's speech capability in the narrator chapter including a working example which shows how to use the translator library together with the narrator device.