Calling TECkit from VB and VBA
Version 0.4 June 2002

Created by Bob Hallissy,
SIL Non-Roman Script Initiative (NRSI)

When using SIL International's TECkit converter on the Microsoft Windows platform, both the engine and the compiler, which are implemented as Windows DLLs, can be called from Visual Basic or Visual Basic for Applications. This makes it possible, for example, to write macros for Microsoft Office applications (e.g., Word, Excel) that can do data conversions "on the fly" inside documents. This document template contains documentation and examples on how to call the TECkit engine from VB or VBA. (I have not included any information about calling the compiler, however, so this code assumes you have compiled modules available.)

Please note that this is not a "finished product", but rather a proof of concept. Feel free to use the included code to develop tools to meet your own requirements.

Finally, this document assumes familiarity with VB- or VBA-based software development.

Copyright information

Installation

In order to execute the example Word macros, the TECkit engine DLL (`TECkit_x86.dll`) must be available somewhere on your PATH.

Additionally, some of the sample code uses the SIL Greek converter (supplied with TECkit). The macros need to know the location, on your file system, of the compiled converter. This is hardcoded in the constant `GREEK_MAPPING_FILE` near the top of the module named `TKWRD_ConvertData` – you will need to modify this constant to match your installation. (You will also need to compile the Greek mapping description to create the mapping file.)

In order for the sample data (below) to display properly *before* conversion you will need the *SIL Galatia* font installed (available here) and for proper display *after* conversion you will need the *Arial Unicode MS* font (a component of Microsoft Office) installed.

Definitions

Within the context of this document, the following definitions apply:

- Mapping Description – a file (usually .MAP extension) containing the source for a TECkit mapping.
- Mapping File – a file (usually .TEC extension) containing a compiled version of a Dapping Description.
- Mapping – an in-memory image of the contents of a Mapping File.
- Converter – an instance of a TECkit engine converter object, created from a Mapping plus some auxiliary information specifying various options (e.g., conversion direction, normalization)
- Legacy text – text that, although stored in Visual Basic strings or Word document objects (which are stored as UTF-16), consists of data encoded in a non-Unicode, 8-bit "legacy encoding".

This last term is somewhat tricky to understand, so is probably worth a little extra explanation.

As an example, suppose you have a Word document that includes text formatted in an SIL Encore-based font (i.e. one built by TypeCaster). You are probably used to thinking about that font in terms of the 8-bit character numbers (32 – 255) that you see in a font chart or perhaps in the Encore CST editor. The relationship between the numbers 32 – 255 and the characters that you see when you use this font represent a custom or *legacy* encoding. It isn't a Unicode encoding because the characters don't exactly match the Unicode charts.

Within VB or VBA, when you type in 8-bit data (or open up 8-bit text files), the application automatically converts this to Unicode by mapping the data through a codepage, usually the system codepage. For your legacy encoded data, this is not a correct Unicode representation (or else you wouldn't need TECkit!). So we call this a "hacked Unicode encoding" – it has the *form* of Unicode (UTF-16) but it is really a legacy encoding. This is what the definition "legacy text" refers to.

For the VB/VBA developer, then, note that:

- When converting between legacy encodings and Unicode, the TECkit engine requires the legacy data to be in the form of a sequence of bytes – not the hacked UTF-16 form that is native to VB/VBA.
- Once legacy text has been converted to correct Unicode (using TECkit), the font it was originally formatted with is no longer suitable – that font was based on the legacy encoding and not Unicode.

Both of these problems are addressed in the code contained in this template.

Template module overview

More information about each code module will be included later in this document, but as an overview, this template contains two VBA modules:

TECkit_VBA

The `TECkit_VBA` module contains two types of code: declares for the functions and constants that make up the TECkit engine interface and a group of utility routines that VB/VBA programmers will find handy. Note: The `TECkit_VBA` module is written to be usable in any Microsoft Office application or in Visual Basic 6. That is, there is nothing in the module that refers to specific application objects or document object models.

TKWRD_ConvertData

The `TKWRD_ConvertData` module is a Microsoft Word-specific module containing macros that demonstrate how to use the `TECkit_VBA` module to either:

- Process an SFM-formatted file (e.g., a Shoebox database), invoking different TECkit converters for data marked with different SFMs.
- Search the currently selected for runs of "legacy text" (so identified because they are formatted in a given font), then use TECkit to convert each such run to Unicode and place the modified text back into the document, changing the font in the process.

As mentioned previously, the `TKWRD_ConvertData` module uses the SIL Greek converter "`silgreek.tec`" to demonstrate the conversion of Greek data into Unicode.

TECkit_VBA module details

After the opening comments, this module contains declares for the TECkit engine. These are derived from the `TECkit_engine.h` header file.

After the TECkit engine declares there are several utility routines that make using the engine from VB a little easier. These are:

- `TKVBA_GetMappingFromFile` - read a mapping file into memory (into a resizable Byte array)
- `TKVBA_CreateConverterToUnicode` - create a converter for converting legacy text to Unicode
- `TKVBA_ConvertStringToUnicode` - convert legacy text to Unicode via a TECkit converter.
- `TKVBA_GetMappingName` - returns a name string from a mapping
- `TKVBA_GetConverterName` - returns a name string from a converter
- `TKVBA_GetStatusDescription` - return a string description of a `TECkit_Status_enum` value

The `TKVBA_CreateConverterToUnicode` and `TKVBA_ConvertStringToUnicode` are the work-horse routines. The first accepts a mapping and creates a convert instance for the purposes of converting legacy data into Unicode. A parameter determines whether you want the Unicode result to be *normalized,* and if so, whether in Normal Form Composed (NFC) or Normal Form Decomposed (NFD). (See here for more information on normalization.) Using the converter thus created, the second function accepts a VB/VBA string (assumed to be legacy text), converts the string back to a string of bytes for the TECkit engine, pumps the string through the TECkit converter, and converts the result back to a proper (now Unicode) VB/VBA string.

The `TKVBA_GetMappingName` and `TKVBA_GetConverterName` are wrappers around the similarly named TECkit APIs which take care of converting the UTF-8 result from TECkit into proper VB/VBA strings.

All of the TECkit engine functions return a small integer that identifies the success or failure of the function. The possible values are identified in the `TECkit_Status_enum` constants. The

TKVBA_GetStatusDescription function simply returns an English-language string describing the various values.

TKWRD_ConvertData module details

The TKWRD_ConvertData module shows how you might use the routines from TECkit_VBA module to do useful work from within a Word document. Remember, this code is intended as an example and not a finished product. The following functions are implemented:

- TKWRD_ConvertRangeToUnicode – A utility function that searches a range of text for a runs of legacy text (identified by being formatted in a given font), pumping each such run through a converter and then reformatting the result with a different font.
- TKWRD_ConvertGalatiaToNFC – an example using TKWRD_ConvertRange.
- TKWRD_ConvertSFMParagraphs – demonstrates how one might process a Shoebox SFM file to convert records to Unicode based on their SFM marker.

The first routine, TKWRD_ConvertRangeToUnicode, is a general purpose one and the most directly reusable. You supply parameters for the document range to search, the target font name to search for, the converter to use on the data that is found, and the font name with which to reformat the now-Unicode data.

The TKWRD_ConvertGalatiaToNFC demonstrates how to use the first routine for a specific job – converting text formatted in SIL Galatia to Unicode. To see it work, select some of the Free-form sample text from below, press Alt-F8, select TKWRD_ConvertGalatiaToNFC, and click "Run".

Finally, TKWRD_ConvertSFMParagraphs demonstrates how one might process a Shoebox SFM file. It loops through the entire document to locate paragraphs that start with an SFM code. Inside the loop it has a place to customize what happens based on the SFM marker. An example is shown of using the SILGreek converter to convert all the text after a "\gr" marker to Unicode and then reformat it with Arial Unicode MS. To see it in operation, press Alt-F8, select TKWRD_ConvertSFMParagraphs, and click Run. Note the changes to the SFM Data sample below.

Sample data

Here is some sample data that you can use to test the functionality:

Free-form data:

᾽Αρχη῾ του῀ ᾽ευαγγελι΄ου ᾽Ιησου῀ Χριστου῀ [῾υιου῀ θεου῀].

Καθω῾σ γε΄γραπται ᾽εν τω῍ ᾽Ησαι¨΄α῍ τω῍ προφη΄τη῍,
᾽Ιδου῾ ᾽αποστε΄λλω το῾ν ᾽α΄γγελο΄ν μου προ῾ προσω΄που σου,
῾ο῾σ κατασκευα΄σει τη῾ν ῾οδο΄ν σου·

φωνη῾ βοω῀ντοσ ᾽εν τη῍ ᾽ερη΄μω῍,
῾Ετοιμα΄σατε τη῾ν ῾οδο῾ν κυρι΄ου,
᾽ευθει΄ασ ποιει῀τε τα῾σ τρι΄βουσ ᾽αυτου῀,

SFM Data:

\eng This is English

\gr ᾽Αρχη῾ του῀ ᾽ευαγγελι΄ου ᾽Ιησου῀ Χριστου῀ [῾υιου῀ θεου῀].

\gr Καθω῾σ γε΄γραπται ᾽εν τω῟ ᾽Ησαι¨΄α῟ τω῟ προφη΄τη῟,
᾽Ιδου῾ ᾽αποστε΄λλω το῾ν ᾽α΄γγελο΄ν μου προ῾ προσω΄που σου,
῾ο῾σ κατασκευα΄σει τη῾ν ῾οδο΄ν σου·

\gr φωνη῾ βοω῟ντοσ ᾽εν τη῟ ᾽ερη΄μω῟,
῾Ετοιμα΄σατε τη῾ν ῾οδο῾ν κυρι΄ου,
᾽ευθει΄ασ ποιει῟τε τα῾σ τρι΄βουσ ᾽αυτου

\eng And this is the end

Known Issues

When replacing legacy text with Unicode text in a Word document, it may also be necessary to set the other font names (e.g., the Bidi Font) as well as the language of the replacement text in order to obtain the correct rendering. This example code does not address these issues.

Revision History

Version 0.4, 2002-06-18

- First public release