

## Questions and comments

Comments on the whole document would be much appreciated. I need feedback on the following points in particular:

### ***Chapter 1: Text Services Framework support***

### ***Chapter 2: Improvements to package handling***

Point 1.2.3. Start Menu items in Redistributable Installers

### ***Chapter 3: European keyboard support (positional/mnemonic layouts)***

Point 7. Supporting Win9x, ANSI

Point 9. Allowing developer-defined workarounds

### ***Chapter 4: TIKE projects***

### ***Chapter 5: TIKE Debugging***

### ***Chapter 6: Keyman language improvements***

#### **Part A. Compiled keyboard files to remain backwardly compatible**

##### **Part B. optany**

Point 1. Do you really really want this?

##### **Part C. named code constants**

Point 2. Do we want Unicode names?

##### **Part D. regexp**

Not in 5.1... Sorry to get your hopes up. We need something for Keyman 6!

##### **Part E. context(n)**

##### **Part F. Improved character position matching**

##### **Part G. Deadkeys and Virtual Keys in stores**

### ***Chapter 7: Add-in support***

I don't particularly want to implement this in Keyman 5.1 -- it's a case of too many new features. Is it very desirable? The primary need is for compatibility with old applications anyway. It probably (hopefully!!) will not be required in the future.

#### **Part A. Add-in manager in KMShell**

##### **Part B. Wordlink as an add-in**

For now, we are not implementing this unless it is very much desired. WordLink will stay in its current form (although we will update the documentation!)

## ***Chapter 8: Minor features***

Part A. Unicode .kmn files

Part B. "Show Character" toolbar

Part C. Regression testing

Part D. Keyboard installation wizard

## ***Chapter 9: General cleanup***

Part A. Character map cleanup

Part B. Edit window cleanup

## Chapter 1

### Text Services Framework support

Title	Text Services Framework Support
Module	Keyman32
Scheduled for	Version 5.1
Scope	New Feature
Importance	High
Type	Functionality Enhancement
Status	■ Not yet started

Adds support for working through the Text Services Framework to retrieve context and input text.

Some requirements:

- \* Keyboards cannot output Virtual Keys.
- \* Keyboards will work in Unicode only

## Chapter 2

### Improvements to package handling

Title	Package Improvements
Module	KMShell, TIKE
Scheduled for	Version 5.1
Scope	Improvement
Importance	High
Type	Useability Enhancement
Status	■ Partially complete

#### 1. Keyman Developer

##### 1.1. Package Editor

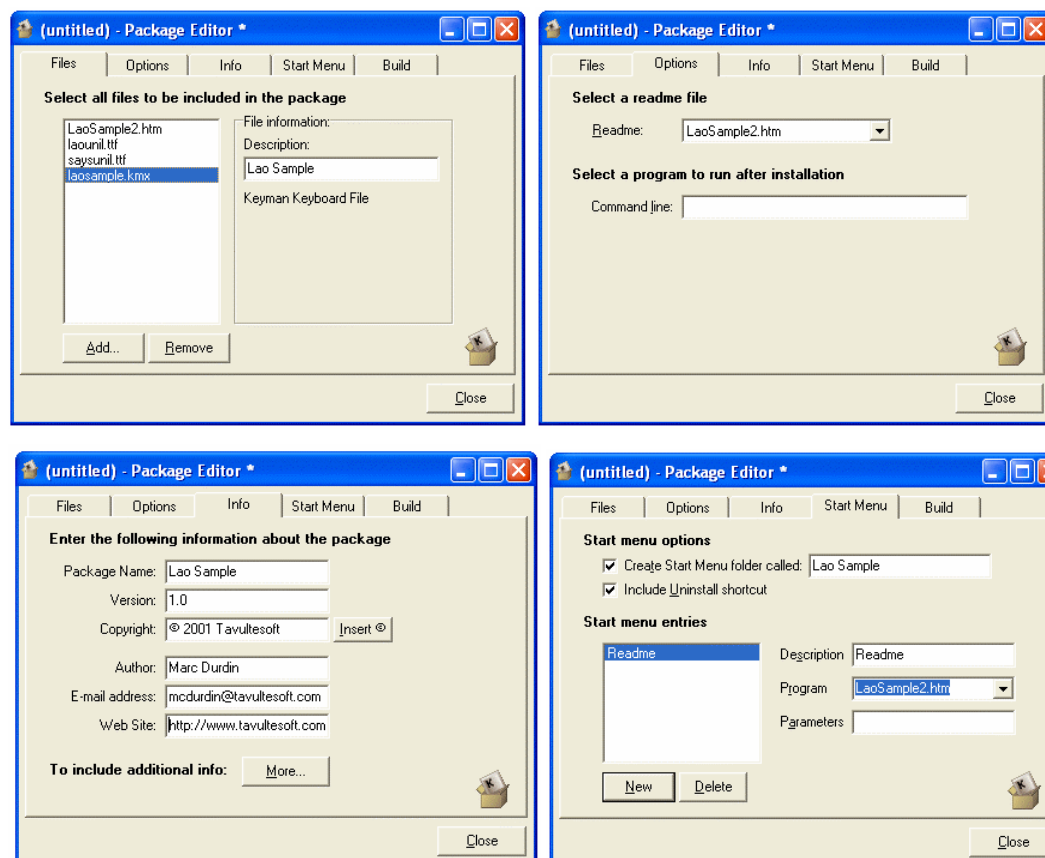
1.1.1. Support multiple keyboards in a single package

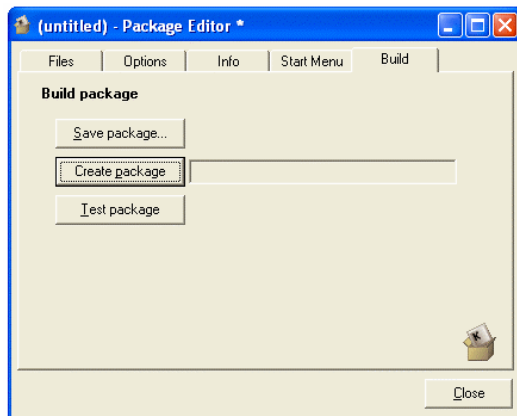
1.1.2. Include all files in a single list in the package editor

1.1.3. Decouple Start Menu Items from list of files so that one can have more control over the Start Menu.

1.1.4. Clean up file dependency and relative path problems (all files will have relative paths in source packages).

1.1.5. Screenshots:





## 1.2. Redistributable Installer Editor

1.2.1. Include all files in a single list in the package editor

1.2.2. Allow users to add only Packages (kmp) and Readme files (.txt, .html/.htm format only)

**1.2.3. Do not include support for Start Menu items in Redistributable Installers (Start Menu items should be included in the kmp packages)**

## 2. Keyman

### 2.1. Package installation

2.1.1. List all keyboards included with the package

2.1.2. Currently packages are handled identically to .kmx keyboards. In 5.1, package will be installed and uninstalled as a single entity. When multiple keyboards are installed in a single package, these will be installed in subdirectories of the installed package directory (usually under documents and settings\all users\application data\tavultesoft\packages\<package name>).

### 2.2. Redistributable installer installation

2.2.1. No significant changes

## Chapter 3

### European keyboard support (positional/mnemonic layouts)

Title	European Keyboard Support
Module	Keyman32, KMCmpDll
Scheduled for	Version 5.1
Scope	New Feature
Importance	High
Type	Functionality Enhancement
Status	■ <b>Partially complete: Win2K/XP/Unicode</b>

#### 1. Mnemonic layouts

- 1.1. Definition: a mnemonic layout uses existing key caps as a memory guide. When the key caps change position in different European keyboards, the Keyman key will move with them.
- 1.2. Recommendation: when using a mnemonic layout, use only characters and not virtual keys in the key part of the rule.
- 1.3. To find the virtual key on a European keyboard
- 1.4. Keyman finds the character associated with the unshifted virtual key on the US English keyboard; if there is no character associated with the key (e.g. function keys), Keyman will use the original virtual key.
- 1.5. Keyman then looks up the virtual key on the active layout that is associated with that character, regardless of its shift position. If the character is not found on the active layout, or the character is only available through a deadkey, Keyman will use the original virtual key.
- 1.6. Finally, Keyman applies the shift state associated with the original rule to the virtual key that it has found.
- 1.7. For example, if the user has a French keyboard, and the keyboard has this rule:

```
+ [CTRL SHIFT K_BACKSLASH] > 'boo!'
```

The backslash character on the French keyboard is located on the underscore/8 key (physically located in the same place as the US English 8/star key. So when the user presses Ctrl+Shift+underscore they will get 'boo!'. Whenever possible, use the character that would appear on the US English keyboard if the virtual key and shift combination was pressed (e.g. [SHIFT K\_BACKSLASH] is equivalent to '|', but may not be the same on other layouts).

#### 2. Positional layouts

- 2.1. Definition: a positional layout will always maintain the same key position, regardless of the system keyboard that is underlying it.
- 2.2. The key position is determined by the scan code; some keys maintain the same scan code but physically move; Keyman will not identify these situations. A common example is the backslash key on US keyboards which can be placed to the left of the backspace key, or below it.
3. Requirement: maintain compatibility with existing compiled keyboards. Keyboards compiled with earlier 5.x compilers will always be recognised as positional.
4. Limitation: Keyman will not support system keyboards that use "ligatures" (one keystroke resulting in multiple characters).
5. Specify positional vs mnemonic layouts with the following system store; if not specified, will always be positional:

```
store(&MnemonicLayout) "1"
```

6. Requirement: compiler will not change, apart from adding support for the MnemonicLayout system store.
7. **BIG QUESTION: Do we need to support Win9x and/or ANSI keyboards?**
  - 7.1. **Currently I have implemented support for WinNT/2K/XP, for Unicode only. Adding support for Win9x is certainly possible but a lot more work is involved. I do not see a great need to add support for ANSI keyboards under Win9x as Keyman 3.2 will work adequately in these situations.**
8. Specification
  - 8.1. Keyman32.dll reads the active kbdxx.dll file and uses that to setup the correlation between scan codes, characters, and virtual keys.
  - 8.2. All character-generating keystrokes do not generate WM\_KEYDOWN/WM\_KEYUP messages while Keyman is active. Instead, only the WM\_CHAR for the character will be generated. This may result in a few programs being unresponsive to control codes if a Keyman layout is active.
  - 8.3. Keyman will always let Windows generate WM\_CHAR messages for any control codes (e.g. Ctrl+C == 0x3, Ctrl+] == 0x1b; Keyman will generate WM\_KEYDOWN/WM\_KEYUP messages for these situations)
  - 8.4. When using mnemonic layouts, a minor time penalty is incurred in matching the keystroke in the kbdxx.dll file. When using positional layouts, the lookup is significantly faster.
  - 8.5. Keyman continues to use Virtual Keys internally but now remaps them as required.
  - 8.6. Keyman emulates the standard behaviour of the kbdxx.dll file; it supports deadkeys and all shifted states of normal keys. It does not support ligatures.
9. **Optional improvement:**
  - 9.1. Allowing keyboard developer to specify different paths depending on the base underlying keyboard. This would allow the developer to create workarounds when they have a keyboard that works in 99% of cases. A rule would only be matched if the keyboard is activated:
 

```
begin Unicode > use(main)

group(main)

physical_keyboard(kbdfr) > use(main_french)
pk(kbdit) > use(main_italian)
pk(kbdde) > use(main_german)

group(main_french) using keys
group(main_italian) using keys
group(main_german) using keys
...

c or another possibility is:

begin Unicode > use(main)

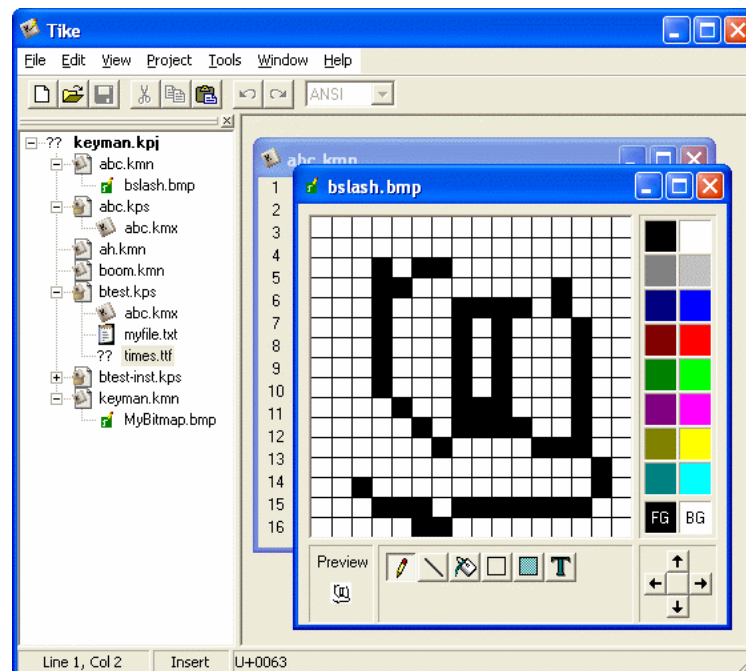
group(main) using keys
pk(kbdfr) 'a' + '^' > 'a^'
pk(kbdit) 'a' + '^' > 'a^'
pk(kbdus) + [RALT K_A] > 'a^'
```

## Chapter 4

### TIKE projects

Title	TIKE Projects
Module	TIKE
Scheduled for	Version 5.1
Scope	New Feature
Importance	Moderate
Type	Usability Enhancement
Status	■ Implementation substantially complete

1. Adds a toolbar to the IDE which allows the user to group files together for simple management.
2. Features
  - 2.1. Batch compilation: compiles keyboards, then packages, and finally redistributable installers.
  - 2.2. Lists subfiles related to each file included in the project (e.g. bitmaps with keyboards, packaged files with packages)
  - 2.3. The project is displayed in a dockable toolbar (docks on the left hand side of the screen).



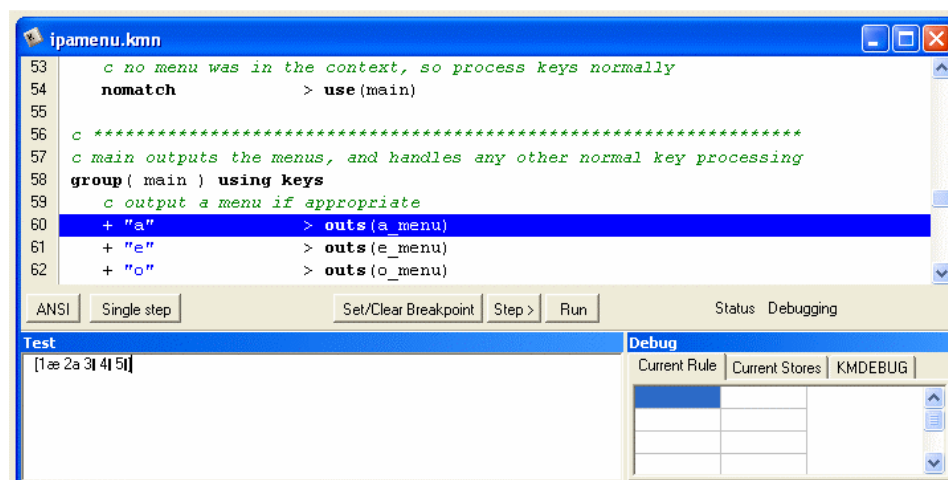


## Chapter 5

### TIKE Debugging

Title	TIKE Debugging
Module	TIKE, Keyman32
Scheduled for	Version 5.1
Scope	New Feature
Importance	High
Type	Functionality Enhancement
Status	■ Partially complete

- Enhances current keyboard debugging support for within TIKE.
- Features
  - When a keystroke is pressed, the developer can automatically run, or single step through the rules
  - Breakpoints can be set on lines which the developer wishes to debug
  - Developer can view contents and indexes of stores referenced in the current rule
  - Developer can view a call stack to understand the keyboard flow
  - Developer can view the state of current deadkeys
  - When the keystroke is pressed, it will be displayed as a keycap in the debug status area
  - This needs to work with European keyboards as well; we can probably use MapVirtualKeyEx to handle this.
  - Developer can select an underlying system keyboard to test with.
  - The test can be run in ANSI or Unicode mode




#### DETAILED STATUS

- Stepping through is complete
- ANSI does not work
- Viewing of call stack, stores does not work
- Key cap is not displayed
- Breakpoints can be set, but will not halt on them
- "Run" mode does not work

## Chapter 6


### Keyman language improvements

#### *Part A. Compiled keyboard files to remain backwardly compatible from now on?*

Title	<b>Compiled Keyboard File Backward Compatibility</b>
Module	KMCompDll
Scheduled for	Version 5.1
Scope	New Feature
Importance	Moderate
Type	Polish
Status	 <b>Complete</b>


1. Require that compiled keyboards remain backwardly compatible with earlier versions of Keyman. Obviously if the developer uses features only available in 5.1, then the keyboard will not work correctly on older versions. Primary requirement is that old keyboards compiled with 5.1 will continue to work with 5.0.
2. System Stores will be used to store any extra data with the keyboard that needs to be understood.

#### *Part B. optany*

Title	<b>optany Support</b>
Module	Keyman Language
Scheduled for	Version 5.1
Scope	New Feature
Importance	High
Type	Language Enhancement
Status	 <b>Not yet started</b>

1. **Do you really really want this? I can write it... but there are a lot of other features awaiting too!**
2. Adds support for the optany() "optional any" statement in the language. index() matching will always work on the index of the item in the rule (e.g.

#### *Part C. named code constants*

Title	<b>Named Code Constants</b>
Module	Keyman Language
Scheduled for	Version 5.1
Scope	New Feature
Importance	Moderate
Type	Language Enhancement
Description	Allows the user to define character constants for Unicode and ANSI values. The standard Unicode value names should also be available, if the appropriate Unicode name file is included in the TIKE directory.
Status	 <b>Not yet started</b>

1. suggested usage:  

```
define(a-dieresis) d225
```

c ...

+ 'a' > @a-dieresis

2. **Optional: include the standard naming file from Unicode.org in Keyman Developer directory.** The user can upgrade to later versions of the file simply by downloading it from <http://www.unicode.org/>.
3. See also Issuetrakka I114.

### ***Part D. regexp***

Title	<b>Regular Expressions</b>
Module	Keyman Language
Scheduled for	Version 5.1
Scope	New Feature
Importance	Low
Type	Language Enhancement
Description	Add support for simple regular expressions in rules.
Status	■ <b>Will not be implemented</b>

### ***Part E. context(n)***

Title	<b>context(n) Support</b>
Module	Keyman Language
Scheduled for	Version 5.1
Scope	New Feature
Importance	Moderate
Type	Language Enhancement
Status	■ <b>Complete</b>

1. Enhance the context statement to support quoting a certain character rather than the complete context, with context(n) with n as the index of the character on the LHS of the rule.
2. Enhance the existing index(store,n) and context(n) functions to work correctly on the LHS of the rule. These can only back-reference (i.e. cannot refer to characters on the right of the function).
3. See Part F for details on another feature that is really needed to make this safer.

### ***Part F. Improved character position matching***

Title	<b>Character Position Matching</b>
Module	Keyman Language
Scheduled for	Version 5.1
Scope	Improvement
Importance	High
Type	Language Enhancement
Status	■ <b>Not yet started</b>

1. At present, it is very difficult if using outs() to match characters safely.
2. We need to add the ability to match on elements, not characters, on LHS.
3. The system store store &OldCharacterPositionMatching will indicate to the compiler to use the old character position matching method.
4. Keyboards with an old version number will always use &OldCharacterPositionMatching. All other keyboards will use the new element position matching. Internally, the compiler will add a &OldCharacterPositionMatching system store when compiling if the old version number is found.

### ***Part G. Deadkeys, virtual keys in stores***

Title	Deadkeys and virtual keys in stores
Module	Keyman Language
Scheduled for	Version 5.1
Scope	Improvement
Importance	High
Type	Language Enhancement
Status	■ Complete

1. As it says... You can now use dead keys and virtual keys in stores and they will be indexed correctly in any/index/context usages.

## Chapter 7

### Add-in support

#### *Part A. Add-in manager in KMShell*

Title	<b>Add-in Manager</b>
Module	KMShell
Scheduled for	Version 5.1
Scope	New Feature
Importance	Moderate
Type	Installation Simplicity
Status	■ Will not be implemented

1. Create an add-in manager for Keyman add-ins that allows the user to simply install and uninstall add-ins (which can be created by Tavultesoft or other third parties). A current example add-in is WordLink.
2. Display a list of currently installed add-ins for Keyman
3. When uninstalling Keyman, uninstall all add-ins first
4. Add-ins are .kmp files -- but when installed, will be detected as add-in.
5. Separate subdirectory under Application Data for Tavultesoft
6. Add-ins can be installed per user?

#### *Wordlink as an add-in*

Title	<b>WordLink Add-in</b>
Module	WordLink
Scheduled for	Version 5.1
Scope	Improvement
Importance	Moderate
Type	Installation Simplicity
Description	Convert WordLink to a Keyman add-in
Status	■ Will not be implemented

## Chapter 8

### Minor features

#### Part A. Unicode .kmn files

Title	Unicode .kmn Files
Module	Keyman Language, TIKE, KMCmpDll
Scheduled for	Version 5.1
Scope	New Feature
Importance	High
Type	File Format Enhancement
Description	Add support for UTF-8 and UTF-16 .kmn files to TIKE.
Status	<span style="color: green;">■</span> Complete

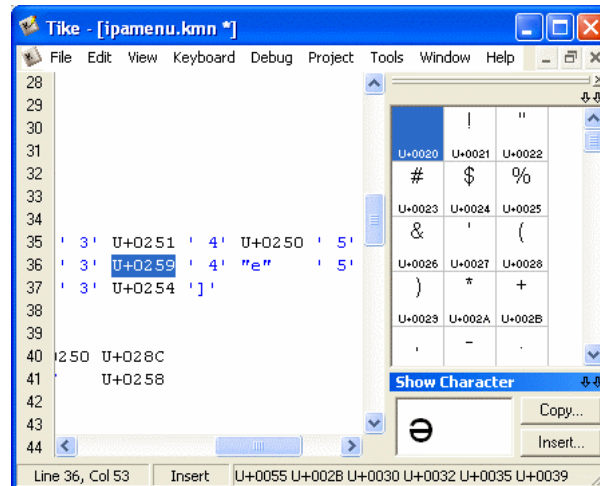
- UTF-8 must always have UTF8 signature; UTF-16 must always have UTF16 signature. Otherwise they will be recognised as ANSI.
  - This should not be a problem as Notepad already includes UTF8 and UTF16 signatures. So the Windows standard is to always include signatures.
- TIKE will allow simple conversion between file formats on the toolbar (will warn if the loss of information likely due to down-conversions).
- KMCmpDll already converted ANSI to UTF-16 in earlier versions internally. Just needed to add support for UTF-8 and UTF-16 source files. Note that ANSI values are just zero-byte-extended in UTF-16 within compiled keyboards -- but this is just a programming consideration, not important for developers or users.
- TIKE: include support for inputting Unicode characters via codenum (U+1234) and converting them to the appropriate Unicode values with Ctrl+Shift+C.

#### Part B. "Show Character" toolbar

Title	Show Character Toolbar
Module	TIKE
Scheduled for	Version 5.1
Scope	New Feature
Importance	Moderate
Type	Useability Enhancement
Description	
Status	<span style="color: green;">■</span> Complete

- Displays a little box at the bottom of the Character Map that showed the currently selected (or caretted) char code as a character in the current Character Map font.
- Optional: select the character within the CharMap. If multiple codes are selected, show all the codes in the box.**

## 3. Screenshot:

**Part C. Regression testing (language)**

Title	<b>Regression Testing</b>
Module	Regression Tester
Scheduled for	Version 5.1
Scope	New Feature
Importance	High
Type	Debugging Enhancement
Status	<span style="color: yellow;">■</span> <b>Partially complete</b>

1. A small program and a set of Keyman files that allow certain Keyman language features to be tested to ensure that they don't get broken in later versions.
2. This will probably be downloadable but will not be included in the standard Keyman Developer download as most users will never need it.
3. Whenever an issue is discovered in Keyman language, a small test keyboard that isolates the problem will be created and added to the regression testing suite.
4. Current status: we have implemented the tool, and about 15 regression keyboards so far. Need to implement a few more keyboards yet.
5. This tool will not directly cover interoperability issues with applications, compiler bugs, or user interface issues.

**Part D. Keyboard installation wizard**

Title	<b>Keyboard Installation Wizard</b>
Module	KMShell
Scheduled for	Version 5.1
Scope	New Feature
Importance	Low
Type	Useability Enhancement
Description	Simplify installation of keyboards with a wizard which displays some general info in an easy to read format, an Advanced... button, and an Install button. Include an option in Keyman Configuration to turn this off.
Status	<span style="color: black;">■</span> <b>Will not be implemented</b>

## Chapter 9

### General cleanup

#### *Part A. Character map cleanup*

Title	<b>Character Map Cleanup</b>
Module	TIKE
Scheduled for	Version 5.1
Scope	Cleanup
Importance	High
Type	Useability Enhancement
Description	Various small fixes to the Character Map to make it easier to use and more flexible.
Status	■ <b>Partially complete</b>

1. Make Character Map a dockable form.
2. Remove controls at the top of character map to reduce clutter. Use right mouse button to bring up a menu to change fonts, or a single option button to change fonts.
3. Include an option to use the font associated with the active window?
4. Fix I208 -- selecting last line of charmap causes annoying scroll
5. Fix I292 -- character preview does not work for Unicode

#### *Part B. Edit window cleanup*

Title	<b>Edit Window Cleanup</b>
Module	TIKE
Scheduled for	Version 5.1
Scope	Cleanup
Importance	Low
Type	Useability Enhancement
Description	Various small fixes and additions to the editor window to make it slightly more powerful.
Status	■ <b>Partially complete</b>

1. Include a separate "quoted text" font so that users can show their custom language in the quoted characters.
2. Include HTML colour coding for the UltraEdit control.

### **OTHER THINGS**

3. Add checkbox to turn on "Check File Associations" to Setup for TIKE, Keyman Configuration.