Unichars for Windows
v 1.4

for Windows 2000, XP, Vista and 7

FREEWARE

TODO : DejaVu
tooltip configuration
Conflicts

1. Introduction

UniChars for Windows is a Keyboard extender which allows an easy way to enter characters non present on your keyboard. Unicode has open access to hundreds or thousands of characters, and we need a way to access them directly from our keyboard without having to remember complex codes or shortcuts. The basic idea of Unichars was taken from the excellent AllChars program (http://allchars.zwolnet.com) which added this functionality years ago. But unfortunately it is still missing Unicode support.

This is the reason for which we decided to write Unichars with AutoHotkey (www.autohotkey.com). Thanks a lot to the author of AutoHotkey.

UniChars for Windows is very easy and intuitive to use. If you have any question, first read this document again. If you still don't find your answer, find a friend who is expert in computers, he should find the explanation easily. If he confirms there is a bug, then add a bug report on the site.

2. Installing UniChars on your system

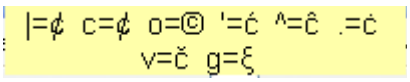
If you have downloaded Unichars_x.xx_setup.exe, the program installs like most Windows programs.

If you have downloaded the zip file, you can install UniChars on your system by copying all UniChars files to any directory on your harddisk. To activate UniChars, just run the unichars_x.x.exe file.

UniChars does not make changes to your system files.

3. Quick Start

- Install UniChars and run it.
- You see the UniChars icon in the system tray.
- Go to your wordprocessor
- Press and release the CTRL key, then type a c.
  A tooltip appear :
  It indicates you the characters you can type and the result it will produce.



- Let us select o : You type o and then, you see: ©
- It is that simple ! Let us try a little further.
- Press and release the CTRL key, then type u and g. The greek letter υu appears. Now type " : the letter gets an accent : ϋ. The tooltip still offers you two choices. Type ' and you get ΰ, the symbol in the Unichars icon.
- Now Press and release the CTRL key and type F1 (the F1 function key)
- The Main screen appears.
- Select a letter in the left list, then a line in the right list, double click on the line or click OK, the selected data is pasted in your text. This screen helps you become familiar with the codes, but it is faster to use directly CTRL followed by the code.
- There are other features you may discover below, but you have seen the basic idea.

4. What UniChars can do

UniChars for Windows uses the clipboard to send to the active application a Unicode string. You can use it for accentuated characters not present on your keyboard, or for Greek, Arabic, Hebrew, Chinese characters.

You can define strings yourself and they can be used for:

- Inserting often-typed strings like addresses, your name, company name, etc.
- Inserting code bits for us programmers, like ‘if then begin end else begin end’ etc.

- Inserting log-in and password information. (Not my responsibility),
- Anything you don't want to type again and again (like 'UniChars for Windows' when writing this manual).

N.B. : Greek, Hebrew, Arabic and so on will work only if you are using :

- A Unicode aware application (Word, Open Office, Atlantis, Notepad…)
- A font which contains the characters you want (Arial and Times New Roman contain thousands of characters. But for Chinese, you will need special fonts. Unichars cannot change the font, but in Word 2003 and Atlantis, if a given character is not found in the active font, the program will automatically look in the fonts installed to find a font which can display the character.

The UniChars Main screen shows the available characters / strings, those which are defined in the ini file(s). You can add characters or strings by editing this file or in the gui.

a) How does UniChars work?

When UniChars for Windows is started, the program 'hooks' itself into the Windows system and watches all keys pressed.

By pressing a compose-key (that you can define) UniChars is activated. It will now watch the following keys and if it finds a correspondence to a sequence defined in the ini file, it will copy the string defined for this sequence.

For example:

If the Ctrl key is defined as the compose-key, which is the default when you install the program,

Ctrl, c, o gives : ©

Ctrl, 1, /, 2 gives : ½

Ctrl, a, " gives : ä

Ctrl, a, h gives : ﻩ

Because the Control key is normally always used together with another key (pressed at the same time), but UniChars is activated by pressing and then releasing the Control key, you can use UniChars with all Windows programs (including system dialogs, etc.) without conflicting with these programs.

But if the use of the Control key does not meet your needs, you can use the Alt or Shift keys, or the Win keys, or one of the multimedia keys of your keyboard (Well, we hope this is

untested because we don't have a multimedia keyboard). In a next version, UniChars may support almost any key on your keyboard as compose-key (or even key combination, but this is not very handy).

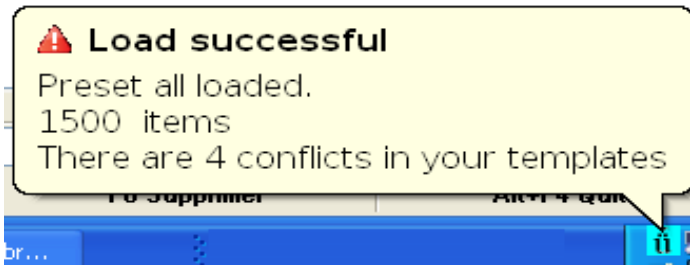
5. Terminology

First we must speak the same language and make some conventions. In the text below :

- Key : A physical key on your keyboard
- <compose> = the key you have chosen to awake UniChars (default is Control). May be referenced also as the compose-key.
- Item : a line of the configuration file which contains two parts with an = between :
  code = string
  Example : co = ©
- Code : The characters you will type after the compose-key (Example : co)
- String : The text that will be pasted in your program, once the code has been typed. Example : ©

6. Using Unichars

Once UniChars has been installed and launched, a small icon appears in the system tray. This icon contains a right click menu which allows to close the program and open the Main window and configuration window. Upon loading, Unichars will show an information message :



If conflicts in definitions were encountered, this is indicated.

Conflicts in templates are not critical, they mean that two definitions have the same code. The program will work as usual but it will use only the first of each pair of definitions in conflict. Example : If you load in the same group the templates 08 – Greek and Coptic, and 08 - Greek Extende, you will have 18 conflicts because the 18 characters with accents present in 08 - Greek and Coptic have been added also in 08 – Greek Extend and the same code is used in both case.

a) Basic use

- Type the compose-key (Control key by default) and release it (very important).
  The system tray icon turns red indicating UniChars is active and what you type

will not appear on screen.

- Type an existing code (example co). When you type the first character, a tooltip appears which indicates you the characters you can use second character for this code. Nothing appears in your document. When you have typed the full code, then the corresponding character or string is inserted in your document at the position of the cursor. In our example you will see the character © appear. UniChars icon returns to its normal color : ü. You continue to type your text.
- If you type a code which does not exist, the tray icon returns to its normal color and the last character you have typed is inserted in your text.

Some codes may need 3 or 4 characters. To see the list of installed codes, use the Main screen.

b) Overlapping codes

UniChars supports overlapping codes. Suppose you have the following codes defined :

ug = υ

ug" = ϋ

ug"' = ΰ

When you have typed <compose>ug, the corresponding character or string will appear. But UniChars knows that you are perhaps typing only the beginning of ug" ou ug"é and shows it by turning the tray icon to orange ü and not back to blue. If you continue typing a ", then the character or string corresponding to ug" will replace the previous character. And if you continue with an ' , then the character or string corresponding to ug"' will replace what was inserted before. In the above example you will see successively :

υ ϋ ΰ

c) Continuous mode

If you type the compose-key twice, Unichars enters in continuous mode which is useful if you want to enter several codes one after the other, such as for typing a word in Greek. In this mode :

- Once a valid code has been entered, instead of returning to idle state (icon blue) Unichars returns automatically to the active state (icon red).
- To get out of continuous mode, type ESC
- You can finish an overlapping code by :
  - typing the compose-key
  - typing a character which is not valid for this code (details below)

Special considerations for overlapping codes

Suppose you have the following codes defined : ab, abc, cd, ef

If you want to type ab, ef, you can just type abef. After you have typed ab, the

corresponding data will be pasted in your text and the icon becomes orange because there is still a valid overlapping code. When you type the following letter (e) the program sees that this letter does not pertain to the possible overlapping codes and stops waiting for any complement. The e is then treated normally in a new loop.

If you want to type ab, cd, once you have typed ab, the corresponding data is entered and the icon becomes orange because there is still a valid overlapping code.

Now, if you type c, the program has no way to decide if you are completing the code or starting a new one. So it will decide that you are completing the code and you want abc in our example. If this is not what you want, type ab followed by <compose-key> and Unichars will understand you want to type a new code, and "cd" will be correctly interpreted.

The language mode

This is not really a different mode, but the most interesting application of the continuous mode. Language models, such as Greek, has always in the codes a letter which identifies the language. This is necessary to prevent conflicts if you load simultaneously several language models. But the consequence is that typing a word in Greek in the normal mode is annoying, because for each letter you have to type the compose-key, and inside the code (generally in second position) the identifier. The continuous mode already prevents you to type repeatedly the compose-key. The language mode will prevent you to type the identifier. To activate it, you must create a group which name starts with * (example : * Greek) and which contains only one language. It may contain more than one template, such as for Greek which requires two templates (Greek and Greek extended). When this group is loaded, the identifier will be omitted.

Code	Value
ag/	ἀ
ag\	ἀ
ag/`	ἂ
ag\`	ἂ
ag/'	ἄ
ag\'	ἄ
ag/~	ἆ
ag\~	ἆ
Ag/	Ἀ
Ag\	Ἀ
Ag/`	Ἂ
Ag\`	Ἂ

Compare these pictures : On the left the group "Greek" which is loaded normally.

On the right, the group "* Greek" which omits the identifier "g" when it is loaded. Both groups contain the same templates.

The codes on the right are much more clear and easy to type.

Code	Value
a/	ἀ
a\	ἀ
a/`	ἂ
a\`	ἂ
a/'	ἄ
a\'	ἄ
a/~	ἆ
a\~	ἆ
A/	Ἀ
A\	Ἀ
A/`	Ἂ
A\`	Ἂ

Example : 3 ways to type the word : Ἀρχὴ

- normal mode (no fun !) : <compose>Ag`<esc><compose>rg<esc><compose>xg<esc><compose>hg`<esc>
- continuous mode (smarter) : <compose>Ag`rgxghg`<esc>
- language mode (the best) : <compose>A`rxh`<esc>

Echo

The continuous mode may be used to help typing in a language for which your keyboard has not the necessary keys, without the need of the compose key. For this, you must create a

template with just the codes you need for this language. Example : if you to type in French with an English keyboard, you will have to define : àéèùâêîôûäëïöüç and the corresponding letters in uppercase. Then you activate the continuous mode and you type normally. When the code you defined for an accentuated character is typed, the accent is automatically inserted.

But in the normal behavior of the continuous mode, each time you type a vowel, the letter will not appear at once, and this is really annoying. It is hence necessary to echo the character on screen as soon as it is typed. This is the purpose of the *Echo* feature. The table below shows in the first column what you type, in the second what appears at this stage in normal continuous mode, and in the third column what appears when Echo is activated :

You type :	Normal mode	Echo mode
e		e
e'	é	é
e't	ét	ét
e'te	ét	éte
e'te'	été	été

d) Reverse codes and Upper/Lower cases

UniChars helps your memory. You may not remember the case of the code, or the order of characters. To help you :

- If a code has two characters and the reverse combination is free, UniChars will add it automatically
- If a code is pure uppercase or lowercase and the other case is free, UniChars will add it automatically.

Example. The code co is defined for ©. But you can type co, oc, CO and OC.

These features can be disabled in the configuration screen.

e) If you don't remember well your code

UniChars can support up to 9 characters for a code. If you are not sure of remembering the codes you need, let the tooltip guide you. It can be configured to show you the remaining possible codes and their result. See below more details on tooltip.

If you don't remember at all your code, type <compose><F1> and the main window will appear, showing you all possible codes.

f) If you made a mistake

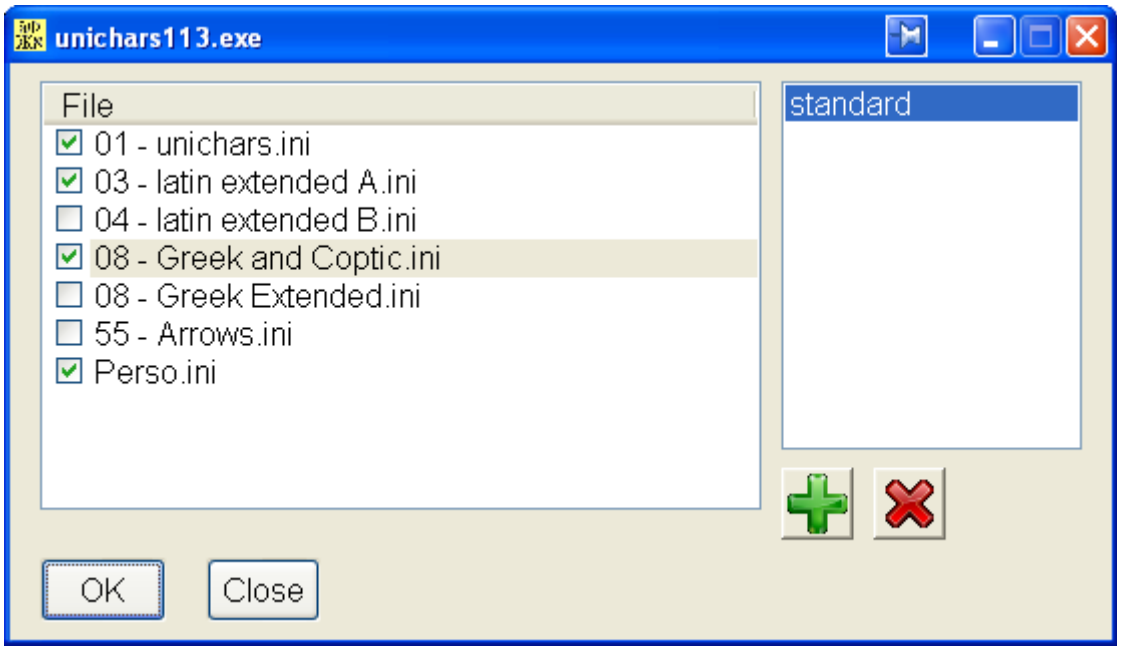
If you type an invalid code, UniChars returns to its normal state (icon blue) and you must delete the characters entered and restart your typing. You cannot (presently) go back with the Backspace key when you are inside Unichars.

7. Using the main window

a) Choosing the templates to load

Unichars may handle thousands of characters (not yet defined in version 1.1, but the library will grow) but if you load all at the same time there is a high risk of conflicts between templates if the same code is used in different models. But generally you don't need to load all templates at the same time. If you are working on a Bible study, Greek and Hebrew templates may be useful. They are less so when you write a letter to your mother. So it is better to define different groups of templates that correspond to the different type of work you are doing and loading them on demand.

To define the groups, in the main Window choose the menu File / Templates Manager.

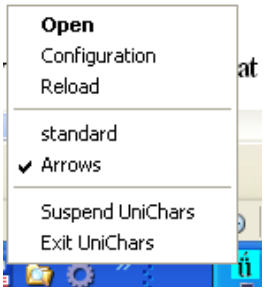


You can add groups with the + button, delete a group with the red X, and when a group is selected, you can attribute it the Templates you want in the list on the left.

Then click Close if you don't want to change the group presently loaded, or click OK and the group selected will be loaded. The changes you make in this window are saved automatically.

Let us suppose you have added the group : Arrows. Now do the following :

- Right click the Unichars icon
- Select : reload
- Once the program is reloaded, right click again on Unichars' icon : you now see :



- This means the group Arrows is loaded. To load the group Standard, just click it. It is reloaded. This feature makes switching from a group to another simple and quick.

b) The main Window

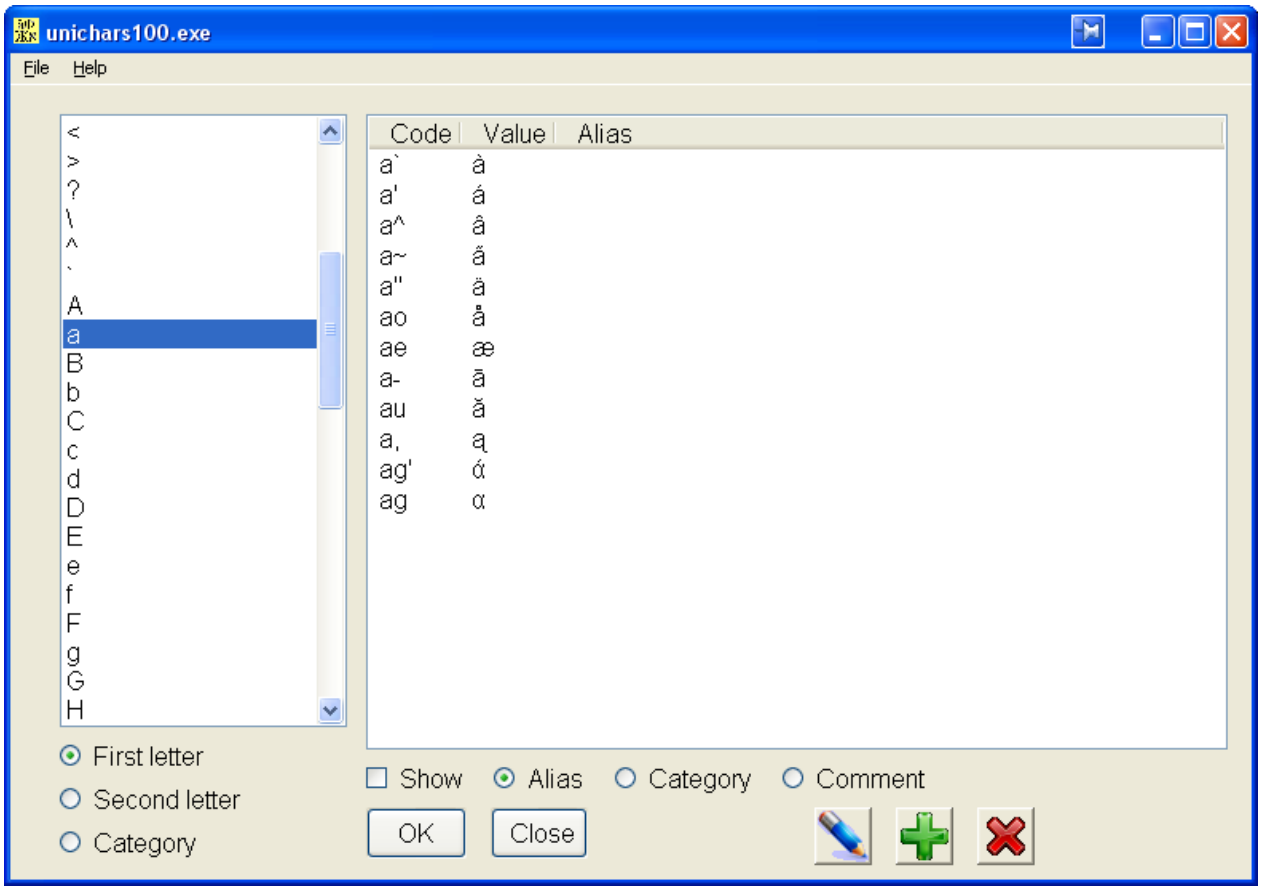
If you don't know what to type, open the Main Window. You can open it in four ways :

- Double click on the tray icon
- Right click on the tray icon, and select Open

IMPORTANT NOTICE : using the tray icon to open the Main screen will not allow you to insert a string by the **Insert** button (or double click, see below) for the following reason : When you click the **Insert** button, the Main screen is closed and the last active window is restored. But in this case, the last active window is the tray icon ! And this is this icon and not your application which will receive the string (and just discard it). For this reason, if you want to insert a string in your text, you must use one of the following ways to open the Main screen :

- Type <compose><F1>
- Type the compose-key and the beginning of the code, then <F1>. This will open the Main screen and fill the list with the codes which starts with the first letter you typed.

Suppose you typed : <compose>a<F1>, you will see :



The radio buttons will fill the left list in three different ways :

1. List of the first letters of your codes
2. List of the second letters of your codes
3. List of categories

The reason of 1. and 3. should be obvious. The reason of 2. is the following : The base template of UniChars place the accent **after** the letter because we think this is more natural. When you write, you write first the letter and then the accent. Nobody will imagine to write the accent first. The base template of AllChars does the converse. This is the behavior of old typewriters with dead keys which didn't move the carriage. This has been imitated by computers although they would be able to handle the natural way. In AllChars (and UniChars) this is not an important matter because normally both directions are allowed but it was interesting to see how sometimes computer engineers keep old habits which are no longer meaningful. Do you know that our keyboards were designed to **prevent** fast typing because old typewriters couldn't handle a high speed ? And we are still using these anti-natural keyboards because it is so difficult to change old habits... To come back to our window, since the accent is generally the second letter of the code, it is interesting to know all codes which use a particular accent.

When you click on an item in the left list, the right list is updated with corresponding data.

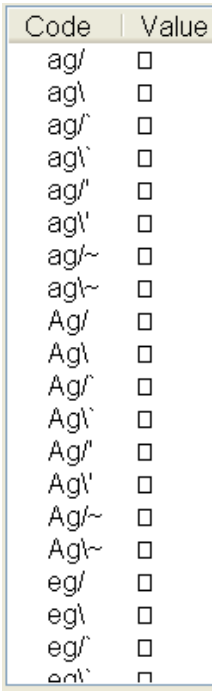
Select an item in the right list, and insert it in your text by either :

- Double clicking the item
- Clicking the OK button

You can sort the items in the right list by clicking the title of a column.

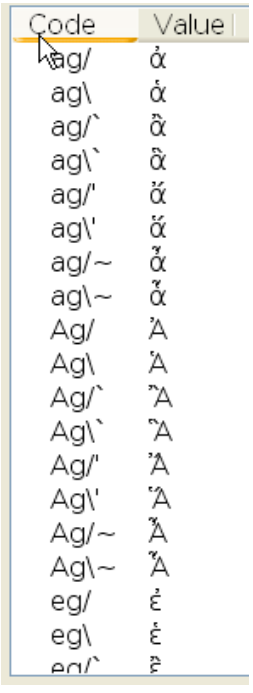
To edit an item, click on Edit.

c) Fonts issues



It may happen that in the right list you see rectangles instead of characters. This will happen if you have selected the font Arial for the gui, and you select the template Greek Extended. The reason is simple : Arial does not have most of the characters of the section greek Extended. To display these characters you must have a font which has these characters. An example is the font DejaVu Sans which contains more than 5000 characters. It is free and you can find it easily on Internet.

Install it in your system, then choose it as the font for the interface and you will see all the characters in the right list.



d) The grab function

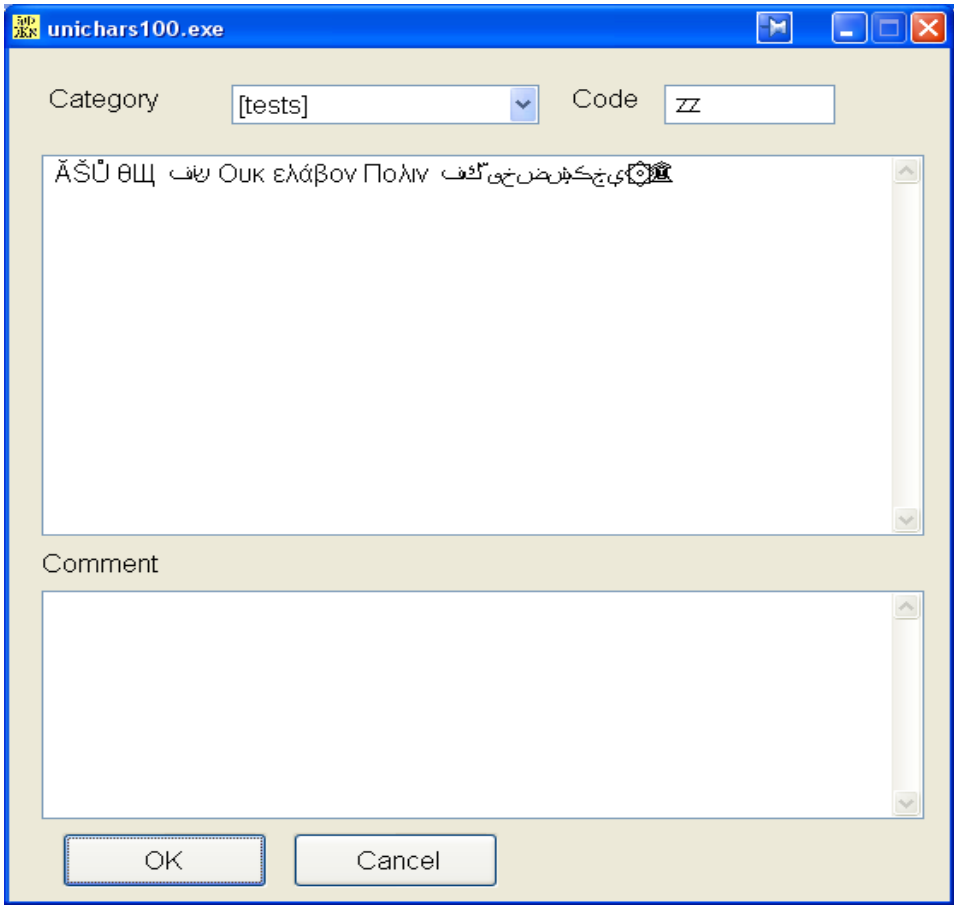
You may want to use UniChars to insert not only characters but strings or small texts. To facilitate this task, a special code that you can define in the configuration screen will launch the grab function. Here is how it works :

- Select a portion of text
- type <compose><F2> (or the code you have defined)
- The edit box opens and your text is pasted in it. You must choose the category and the code.
- Click OK
- Due to an unfixed bug your new code will not work until you reload the program. Use the Reload button or the Reload option in the tray menu for this purpose.

8. Editing items

a) The edit window

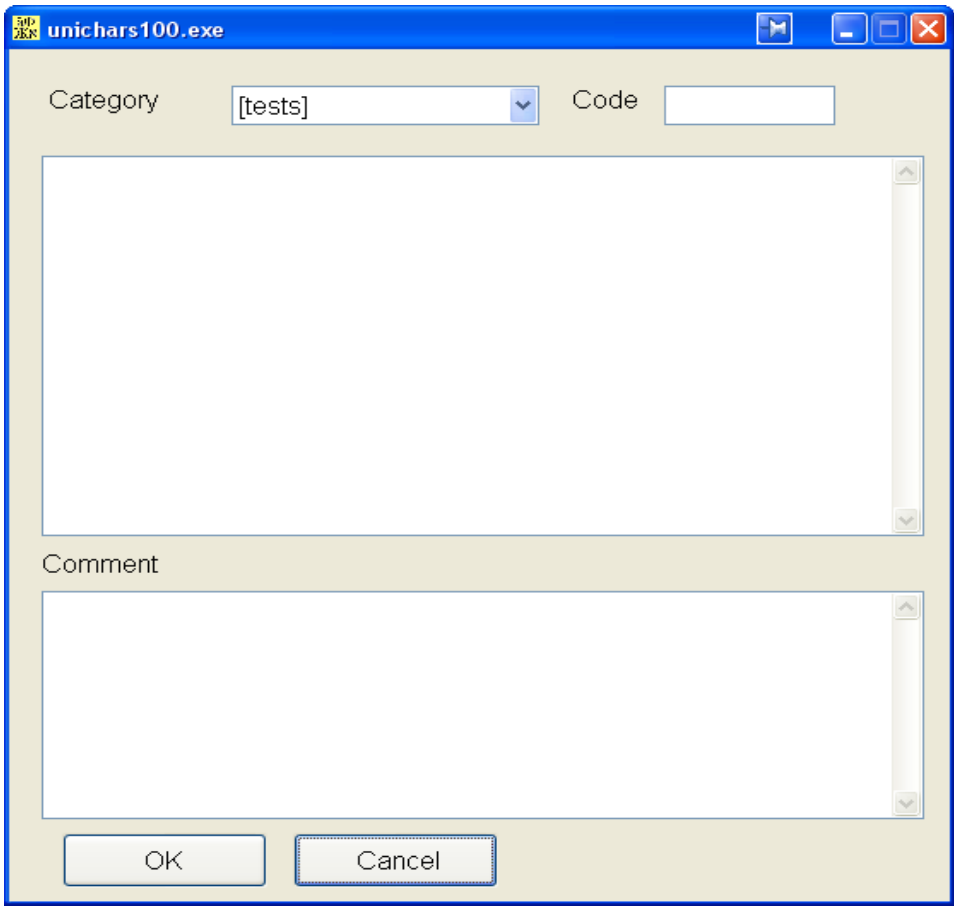
The Edit button allows you to edit the selected item.



Type or paste what you want and click OK. The right list updated.

If you have changed the Code and/or Category, a message will ask you if you want to delete the old version.

To create a new item, click on new.



A category will be automatically selected if :

- An item was selected in the right list (the category of this item will be used)
- The left list was in category mode and a category was selected. It will be used.

Otherwise you must choose yourself a category. If you don't, the program will warn you when you click OK.

If you enter a code which already exists, you will be warned when you will click OK and you will be offered the opportunity to change it.

After you have validated by the OK button, your new item will appear in the list.

The delete button will delete the highlighted item.

b) Mass edit

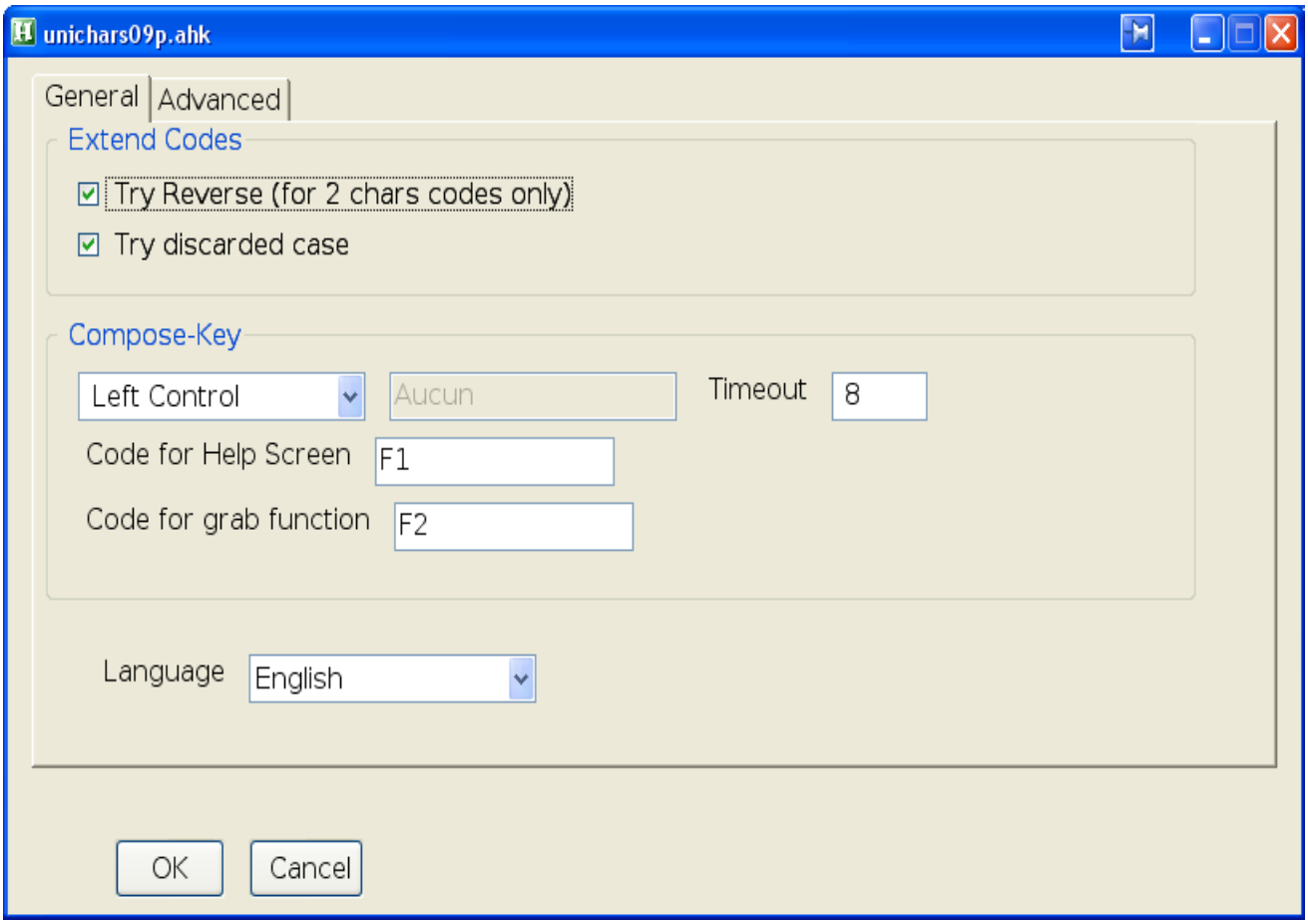
For extended editing it is recommended to open your inifile in an utf-8 capable editor [1] and to edit it directly. See below the section *Configuring the ini file.*

[1] Notepad, Scite and many others

9. Configuring UniChars

a) The configuration window

Right click on the tray icon and select Configuration, or open the Unichars window and select the menu *File / Configuration.*



Extend Codes

See the Section *Reverse codes and Upper/Lower cases* above.

Compose-key

You can choose the compose-key in the Drop Down List.

If you choose Control, both Control keys will work. If you choose Lcontrol or Rcontrol, only the left or right Control key will work. The same is true for Alt and Shift.

If the predefined choices don't meet your needs, select : *User defined* and type in the text zone in the right the character or key combination you would like to use. Almost any key of your keyboard may be used.

Tip : if you want to be able to type the character you have chosen as compose-key, add a code for it in your ini file.

Code for Main Screen

The code you will type to open the Main screen. Default is <F1>. You will type <compose><F1> to open the Main screen.

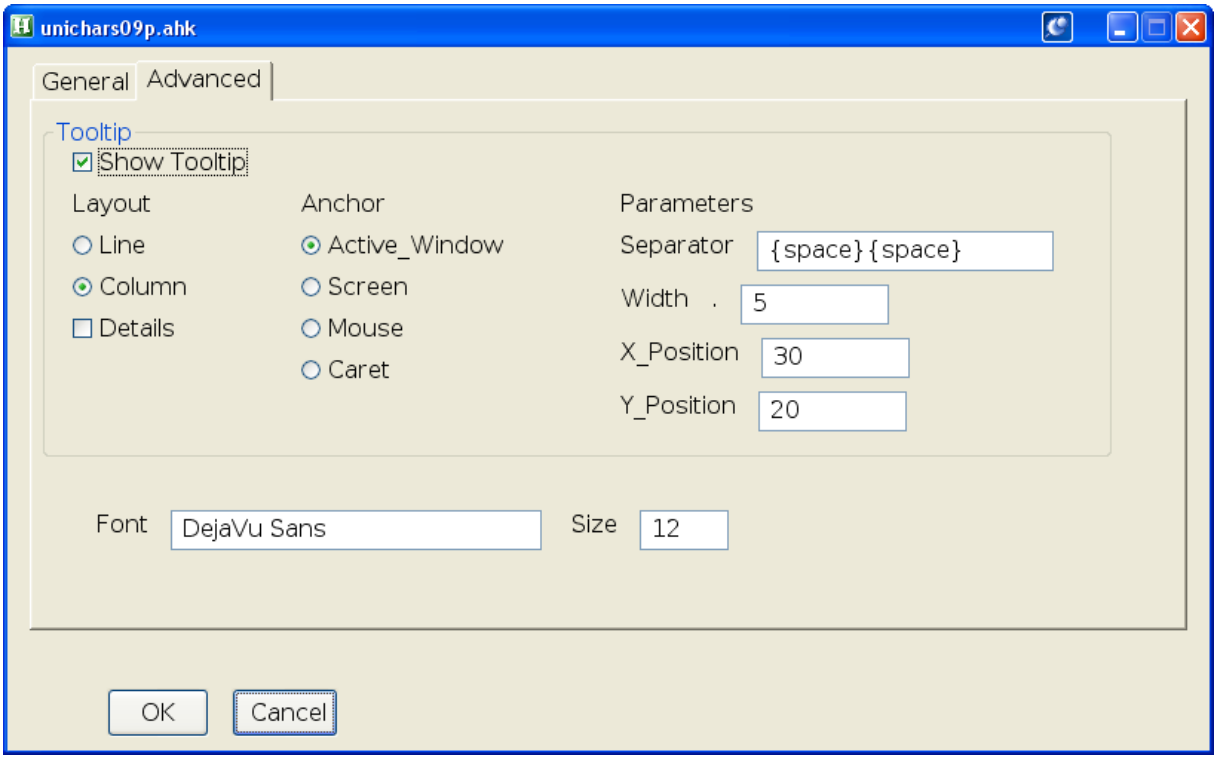
Code for Grab function

Type here the code you want to use to start the grab function. You will have to type <compose> followed by this code, <compose><F2> is the default.

Language

Choose here the language of the interface. The choices are indicated with the standard two letters code for a language.

The *Advanced* Tab shows the following dialog :



Tooltip

N.B. : The following description corresponds to Unichars 1.0. In 1.1 changes occurred, but since they will be changed again in 1.2, the documentation has not been updated yet.

When you type your codes, a tooltip may show you the following valid choices. Here you can activate or desactivate it. You can set the configuration for :

Layout : in line or in column (horizontal or vertical)

Details : if checked, the result of the codes is shown.

Anchor : where the tooltip will be shown :

1. Active Window : In the top left corner of the active window
2. Screen : In the top left corner of your screen.
3. Mouse : Near the mouse cursor
4. Caret : Near the caret (the insertion point of your text).

For 1, 2 and 4 you can select the position from the top left corner or from the caret : Enter

the values in pixel in X_position and Y_position

Note that the position near the caret (the best choice normally) does not work in some applications because they don't return the position of the caret. Unfortunately it seems that the major applications don't care about reporting. In many small programs it works, but in Word and Open Office (Now Libre Office) it does not work. It works well in Notepad (in XP) and Atlantis.

Separator : the character(s) between the items. It may be spaces, like in the example above, or whatever you want.

With : the width of the tooltip in pixels (Unichars 1.1). In Unichars 1.0 Width was the number of items in a line, if the layout is in line.

Font - Size

Font and size to use for the UniChars window. The Window will enlarge / shrink as well.

When you click OK the program will restart to apply your new settings.

10. Removing UniChars from your system

If you installed UniChars with Windows setup (using SetupUniCharsxx.exe) you can uninstall UniChars like most Windows programs. Use the 'Add/Remove Programs' option of the 'Control Panel'.

If you installed UniChars by hand (copying the files), you can remove UniChars from your system by deleting all files in the UniChars directory. And if you added UniChars to your 'Startup' group, you also have to remove it from there.

Advanced User Guide

1. Configuring the ini files

Mass update may be more practical by direct editing of the ini files. But this needs more knowledge because you must handle yourself the special characters and some rules. Reed carefully this Advanced user guide before editing an ini file.

a) The ini files

UniChars was designed to handle hundreds or thousands of codes. The free DejaVu family (find it here : http://dejavu-fonts.org) contains fonts with more than 5000 characters. It is very powerful but how give access to the characters you need ? Maintaining a single file with all these codes would have been difficult, for us and for you. So we have decided to add multi file support in UniChars.

Location of ini files

On starting, Unichars will look in the directory “templates” and load each file found there with an ini extension. The files are loaded in alphabetical order. This allow you to control the order of categories in the list of the Main window. Just add numbers in the beginning of your filenames to insure they are loaded in the order you want.

The special unichars_xx.ini

There is a special ini file which UniChars will always load first : the unichars_xx.ini file found in the subdirectory of the locale directory which corresponds to the language defined in the configuration screen.

Example : If your language is set to en (English) then the file

./locale/en/unichars_en.ini

will be loaded, if it is found.

This file contains only the standard [#help] and [#accents] sections for your language. It is not recommended to edit them because they would be overwritten by the next install. But you can modify their content in your own files. See below the details about these sections.

This file was placed in the *locale* directory because the keys used for accents and the content of the Main window are language dependent.

Editing the ini files

It is important that you can edit the ini files because the main point for a keyboard software is that you can remember the codes you need. AllChars had 200 codes, it was possible to find good combinations for all. With 5000 characters, it is no longer possible. And some persons will find a code good and others will not remember it. So the best is that you can define the code you want for the character or string you want.

You must nevertheless keep in mind that it is a little tricky : Read carefully the

recommendations below.

The syntax is the standard syntax of ini files :

Sections between [ ]

items in the form : code = string

comments in lines beginning by ;

b) Recommendations

Sections

You must not create duplicate sections. If you do, the program will warn you on loading. There is an exception : [#help] and [#accents] sections may be found in each file, but this is an advanced feature (especially for [#accents] to be used only by advanced users.

There are two special sections [#help] and [#accents] their use is described below.

Other sections have two purpose :

- grouping characters in categories and allow to find them more easily in the Main screen.
- Indicating to UniChars in which files resides a particular code, so that it can be stored in the right file when you edit a code.

Codes :

- You cannot use = in codes. You must replace it by {equal}.
- You cannot use ; in the beginning of a code, the line would be treated as a comment, you must replace it by {semicolon}.
- You cannot use spaces in the beginning or end of a code because leading and trailing spaces are ignored. You must replace them by {space}.
- You cannot use tabs inside codes. You must replace them by {tab}.
- You must not create duplicate codes. If you do, the program will warn you on loading.
- The maximum length of a code is 9 characters.
- You can use the arrow keys in codes :
    - {<} for the left arrow
    - {>} for the right arrow
    - {^} for the up arrow
    - {v} for the down arrow
    - {<<} for Home
    - {>>} for End
    - {^^} for Page Up
    - {vv} for Page Down

They will appear in the lists with the following symbols : ← →↑↓◄ ►▲▼. Presently they will appear in the tooltip as squares because the font of the tooltip does not support these characters.

All this is true only for codes, in the strings you can use everything.

Be careful not to create duplicate codes, the last definition will overwrite the first. If you do, the program will warn you on starting and the duplicate codes will be copied in the log file (unichars.log) so that you can easily edit your ini file.

Overlapping codes are allowed (example : ab, abc and abcd) but see above how they will work (Section : *Overlapping codes*).

Strings

Maximum length of strings is 62000 characters.

The ini file may start with two special sections :

Inline comments

Inline comments are allowed after a semicolon. For this reason, it you want to have a ; in your data, you must replace it by {semicolon}, otherwise everything which follows the ; will be considered a comment.

Inline comments have two purpose :

1. Comments. Obviously !
2. Send special keystrokes. If a comment starts with : Send: everything which is after the colon will be sent by Autohotkey in its particular format. This allows sending special key strokes like arrow keys. Example : if you are writing a template for Hebrew which writes from right to left, you may want to position the cursor **before** the character inserted. Just do that :

   ah = א ; Send : {left}

The full list of AutoHotkey codes to be used is found in Appendix I

If you create a new ini file

Important notice : **All files must be written in utf-8.** If you create a file from scratch don't forget this otherwise UniChars will not load. A good editor for utf-8 files is Scite because it allows you to switch from utf-8 to 8 bits, when it is necessary that you see the utf-8 codes. Notepad can work in utf-8 mode if the file contains an indicator named BOM (see on wikipedia if you want to know what this means) which tells that the file is in utf-8. But Notepad cannot switch from an encoding to another and does not tell you which encoding you are using. But Notepad has a feature which is difficult to change in Scite : you can change the font and choose a rich utf-8 font like DejaVu. In Scite you must open the global properties, look for : font.text and choose there an appropriate font. Then reload Scite and verify that *Use monospaced font* is not checked in the *Options* menu.

c) The [#help] section

In version 1.0, this is help (lower case). Help (with upper case H) will be considered as a standard section.

As explained above, the special ini file loaded from the *locale* directory contains a

section [#help]. This section contains text which will be displayed in the Short Help window. It is useful to summary the basic principles of the ini file, especially the characters used for accents.

This section is special because it may be found in each ini file. The text of each [#help] section will be added after the previous one. This allows you to add data to this section without modifying the file in the *locale* directory which is not recommended.

But what if you don't want to *add* data but replace it totally ? This will be handled in a future version of Unichars by distinguishing between two names : [#help] will replace totally all previous data, [#help+] will add data. This is not implemented yet.

d) The [#accents] section

In version 1.0, this is accents (lower case). Accents (with upper case A) will be considered as a standard section.

This section is destined to allow an easy tuning of UniChars for users of non English keyboards. The default English definitions for accents uses ` and ~ for some accents. This is fine with an English keyboard, because you have keys for these letters. But this is unusable with a French keyboard because these letters are very difficult to access : For ` you must use the combination AltGr+7 twice (the first time nothing appears) and after that two ` appear (the reason is that this combination is treated as a dead key by the French version of Windows). So you have no way at all to type `, you will always get `` and UniChars will not work. So we have to change the codes that use ``. Editing the full file would be cumbersome but UniChars offers a straightforward way. In this section you find a line :

{grave} = "`"

In the French ini, it has been replaced by :

{grave} = è

because we have a key for è on our French keyboard (you don't have it on an English keyboard) and it is practical for us.

This section is found in the special unichars_xx.ini file already mentioned above and which is located in the sub-directory of *locale* corresponding to the language you use. This file should not be modified but the section [#accents] may be present in each ini file : the new definitions will overwrite the old ones, and you can add new definitions if it is useful for the ini files you write.

The content of the unichars_en.ini file is :

[#accents]

{acute} = '

{grave} = `

{circumflex} = ^

{tilde} = ~

{diaeresis} = "

{apostrophe} = '

{cedilla} = ,

{caron} = v

{macron}= -

{breve} = u

{ring} = o

{comma-above} = ?

{dot} = .

You may add codes for your own use if you want. The basic idea is that when the program encounters one of these codes in a code below in the file, it will replace them by the character defined.

You may have a [#accents] section in each ini file, but it is not recommended unless you understand exactly what you are doing. Example : we could add :

[#accents]

{cyrillic} = c

in the cyrillic.ini file to allow easy redefining of the second letter of the main codes of this file. Keep in mind the previous {…} defined in the files already loaded are kept in memory. You can overwrite them if necessary.

IMPORTANT NOTICE : If you add an [#accent] section to a file, it will modify the definitions for this file and all files loaded after it. It will not impact the files already loaded. If you want that your definitions are used by all of your files, it is necessary that you add your [#accents] to the first file loaded in *Templates*. Since *Templates* load files in alphabetical order you know which one it is. A good practice is to place numbers in the beginning of your filenames. But remember that if you have more than 9 files, you must write 01 and not 1 if you want to control correctly the order.

e) Other sections

The remaining of the file is constituted of one or more section(s) containing items in the model :

code = string

The sections are useful to group items in the Unichars Window when you select : *category*.

Sections are also necessary to the program to identify the right file.

2. The unichar.cfg file

The file unichars.cfg contains the program configuration : Hotkeys, Font and so on. It contains two sections : [general] and [advanced]. You should normally not change the data in [general] because it is handled by the configuration window.

The data in advanced is intended for advanced users and we suppose they don't need a

gui.

- Debug = 0/1 : if set to 1, additional data is entered in the log file to help debugging
- ComposeFireOnPress = 0/1 : this changes the behavior of your compose-Key. Normally you must press **and release** the compose-key before typing the code. If this parameter is set to 1, the compose-key activates Unichars when you press it, and not when you release it. This allow typing the code before the compose-key is released.
  WARNING : This was intended for keys like the Windows keys. If you use this feature and use a Control key as Compose-key, **you will not be able to use this key for anything else** (like CTRL+I for italics).
- Beta =0/1 : Some beta features may be normally hidden and will appear if this parameter is set to 1. It depends from the version.

3. Versions info

1.4 - 8 March 2011
- Continuous mode
- Language mode
- Echo
- Window is resizable

1.3 - Never released
- Faster loading of templates (x10)
- First draft of continuous mode

1.2 - 20 February 2011
- Unicode support for tooltip

1.1 - Never released
- Templates manager

1.0 – 1 Februaray 2011
- Third column in the list
- inline comments and send command in ini files
- Lots of bug fix

0.9 – 17 January 2011
- Full rewrite of the core functions : much more stable now.
- Multiple templates support
- Languages support
- A tooltip shows the possible choices for the next characters
- The Gui grows / shrinks when the font size is increased / decreased

0.8 – 12 January 2011
- The grab function is configurable.
- Added the sections [#help] and [#accents]

- Added the {…} codes in ini files
- Upper/Lower option works with mixed case strings
- Many improvements in the Main and Edit windows.
- Added a “Reload” button and menu
- grab function rewritten

0.7 – 10 january 2011
First release

APPENDIX I
AutoHotkey Send Keys & Clicks

Sends simulated keystrokes and mouse clicks to the active window. The operating system limits SendInput to about 5000 characters (this may vary depending on the operating system's version and performance settings). Characters and events beyond this limit are not sent.

a) Parameters

Normal mode: When not in raw mode, the following characters are treated as modifiers (these modifiers affect only the very next key):

!: Sends an ALT keystroke. For example, *Send This is text!a* would send the keys "This is text" and then press ALT+a. **Note**: !A produces a different effect in some programs than !a. This is because !A presses ALT+SHIFT+A and !a presses ALT+a. If in doubt, use lowercase.

+: Sends a SHIFT keystroke. For example, *Send +abC* would send the text "AbC", and *Send !+a* would press ALT+SHIFT+a.

^: Sends a CONTROL keystroke. For example, *Send ^!a* would press CTRL+ALT+a, and *Send ^{Home}* would send CONTROL+HOME. **Note**: ^A produces a different effect in some programs than ^a. This is because ^A presses CONTROL+SHIFT+A and ^a presses CONTROL+a. If in doubt, use lowercase.

#: Sends a WIN keystroke, therefore *Send #e* would hold down the Windows key and then press the letter "e".

Key Names: The following table lists the special keys that can be sent (each key name must be enclosed in braces):

Key Name	Resulting Keystroke
{F1} - {F24}	Function keys. For example: {F12} is the F12 key.
{!}	!
{#}	#
{+}	+
{^}	^
{{}	{
{}}	}
{Enter}	ENTER key on the main keyboard
{Escape} or {Esc}	ESCAPE
{Space}	SPACE (this is only needed for spaces that appear either at the beginning or the end of the string to be sent -- ones in the middle can be literal spaces)
{Tab}	TAB

{Backspace} or {BS}	Backspace
{Delete} or {Del}	Delete
{Insert} or {Ins}	Insert
{Up}	Up-arrow key on main keyboard
{Down}	Down-arrow down key on main keyboard
{Left}	Left-arrow key on main keyboard
{Right}	Right-arrow key on main keyboard
{Home}	Home key on main keyboard
{End}	End key on main keyboard
{PgUp}	Page-up key on main keyboard
{PgDn}	Page-down key on main keyboard
{CapsLock}	CapsLock (using SetCapsLockState is more reliable on NT/2k/XP). Sending {CapsLock} might require SetStoreCapslockMode Off beforehand.
{ScrollLock}	ScrollLock (see also: SetScrollLockState)
{NumLock}	NumLock (see also: SetNumLockState)
{Control} or {Ctrl}	CONTROL (technical info: sends the neutral virtual key but the left scan code)
{LControl} or {LCtrl}	Left CONTROL key (technical info: same as CONTROL for Win9x, but on NT/2k/XP it sends the left virtual key rather than the neutral one)
{RControl} or {RCtrl}	Right CONTROL key
{Control Down} or {Ctrl Down}	Holds the CONTROL key down until {Ctrl Up} is sent. XP/2000/NT: To hold down the left or right key instead, use {RCtrl Down} and {RCtrl Up}.
{Alt}	ALT (technical info: sends the neutral virtual key but the left scan code)
{LAlt}	Left ALT key (technical info: same as ALT for Win9x, but on NT/2k/XP it sends the left virtual key rather than the neutral one)
{RAlt}	Right ALT key (or AltGr, depending on keyboard layout)
{Alt Down}	Holds the ALT key down until {Alt Up} is sent. XP/2000/NT: To hold down the left or right key instead, use {RAlt Down} and {RAlt Up}.
{Shift}	SHIFT (technical info: sends the neutral virtual key but the left scan code)
{LShift}	Left SHIFT key (technical info: same as SHIFT for Win9x, but on NT/2k/XP it sends the left virtual key rather than the neutral one)
{RShift}	Right SHIFT key
{Shift Down}	Holds the SHIFT key down until {Shift Up} is sent. XP/2000/NT: To hold down the left or right key instead, use {RShift Down} and {RShift Up}.

{LWin}	Left Windows key
{RWin}	Right Windows key
{LWin Down}	Holds the left Windows key down until {LWin Up} is sent
{RWin Down}	Holds the right Windows key down until {RWin Up} is sent
{AppsKey}	Windows App key (invokes the right-click or context menu)
{Sleep}	Computer SLEEP key.
{ASC nnnnn}	Sends an ALT+nnnnn keypad combination, which can be used to generate special characters that don't exist on the keyboard. To generate ASCII characters, specify a number between 1 and 255. To generate ANSI characters (standard in most languages), specify a number between 128 and 255, but precede it with a leading zero, e.g. {Asc 0133}. To generate Unicode characters, specify a number between 256 and 65535 (without a leading zero). However, this is not supported by all applications. Therefore, for greater compatibility and easier sending of long Unicode strings, use "Transform Unicode".
{U+nnnn}	[AHK_L 24+]: Sends a Unicode character where *nnnn* is the hexadecimal value of the character excluding the 0x prefix. Note that in the Unicode build of AutoHotkey_L, Unicode characters can be directly included in the text if the script file is saved as UTF-8 or UTF-16. If the character doesn't map to a virtual keycode, SendInput() or WM_CHAR is used to send the character and the current Send mode has no effect.
{vkXX} {scYYY} {vkXXscYYY}	Sends a keystroke that has virtual key XX and scan code YYY. For example: *Send {vkFFsc159}*. If the sc or vk portion is omitted, the most appropriate value is sent in its place. The values for XX and YYY are hexadecimal and can usually be determined from the main window's View->Key history menu item. See also: Special Keys
{Numpad0} - {Numpad9}	Numpad digit keys (as seen when Numlock is ON). For example: {Numpad5} is the digit 5.
{NumpadDot}	Numpad Period (as seen when Numlock is ON).
{NumpadEnter}	Enter key on keypad
{NumpadMult}	Numpad Multiply
{NumpadDiv}	Numpad Divide
{NumpadAdd}	Numpad Add
{NumpadSub}	Numpad Subtract
{NumpadDel}	Delete key on keypad (this key and the following Numpad keys are used

	when Numlock is OFF)
{NumpadIns}	Insert key on keypad
{NumpadClear}	Clear key on keypad (usually the '5' key when Numlock is OFF).
{NumpadUp}	Up-arrow key on keypad
{NumpadDown}	Down-arrow key on keypad
{NumpadLeft}	Left-arrow key on keypad
{NumpadRight}	Right-arrow key on keypad
{NumpadHome}	Home key on keypad
{NumpadEnd}	End key on keypad
{NumpadPgUp}	Page-up key on keypad
{NumpadPgDn}	Page-down key on keypad
{Browser_Back}	2000/XP/Vista+: Select the browser "back" button
{Browser_Forward}	2000/XP/Vista+: Select the browser "forward" button
{Browser_Refresh}	2000/XP/Vista+: Select the browser "refresh" button
{Browser_Stop}	2000/XP/Vista+: Select the browser "stop" button
{Browser_Search}	2000/XP/Vista+: Select the browser "search" button
{Browser_Favorites}	2000/XP/Vista+: Select the browser "favorites" button
{Browser_Home}	2000/XP/Vista+: Launch the browser and go to the home page
{Volume_Mute}	2000/XP/Vista+: Mute/unmute the master volume. Usually equivalent to *SoundSet, +1, , mute*
{Volume_Down}	2000/XP/Vista+: Reduce the master volume. Usually equivalent to *SoundSet -5*
{Volume_Up}	2000/XP/Vista+: Increase the master volume. Usually equivalent to *SoundSet +5*
{Media_Next}	2000/XP/Vista+: Select next track in media player
{Media_Prev}	2000/XP/Vista+: Select previous track in media player
{Media_Stop}	2000/XP/Vista+: Stop media player
{Media_Play_Pause}	2000/XP/Vista+: Play/pause media player
{Launch_Mail}	2000/XP/Vista+: Launch the email application
{Launch_Media}	2000/XP/Vista+: Launch media player
{Launch_App1}	2000/XP/Vista+: Launch user app1
{Launch_App2}	2000/XP/Vista+: Launch user app2
{PrintScreen}	Print Screen
{CtrlBreak}	Ctrl+break
{Pause}	Pause

{Click [Options]} [v1.0.43+]	Sends a mouse click using the same options available in the Click command. For example, {Click} would click the left mouse button once at the mouse cursor's current position, and {Click 100, 200} would click at coordinates 100, 200 (based on CoordMode). To move the mouse without clicking, specify 0 after the coordinates; for example: {Click 100, 200, 0}. The delay between mouse clicks is determined by SetMouseDelay (not SetKeyDelay).
{WheelDown}, {WheelUp}, {WheelLeft}, {WheelRight}, {LButton}, {RButton}, {MButton}, {XButton1}, {XButton2}	Sends a mouse button event at the cursor's current position (to have control over position and other options, use {Click} above). The delay between mouse clicks is determined by SetMouseDelay. WheelLeft/Right require v1.0.48+, but have no effect on operating systems older than Windows Vista.
{Blind}	When {Blind} is the first item in the string, the program avoids releasing Alt/Control/Shift/Win if they started out in the down position. For example, the hotkey *+s::Send {Blind}abc* would send ABC rather than abc because the user is holding down the Shift key. {Blind} also causes SetStoreCapslockMode to be ignored; that is, the state of Capslock is not changed. Finally, {Blind} omits the extra Control keystrokes that would otherwise be sent; such keystrokes prevent: 1) Start Menu appearance during LWin/RWin keystrokes; 2) menu bar activation during Alt keystrokes. Blind-mode is used internally when remapping a key. For example, the remapping a::b would produce: 1) "b" when you type "a"; 2) uppercase B when you type uppercase A; and 3) Control-B when you type Control-A. {Blind} is not supported by SendRaw and ControlSendRaw. Furthermore, it is not completely supported by SendPlay, especially when dealing with the modifier keys (Control, Alt, Shift, and Win).
{Raw} [v1.0.43+]	Sends the keystrokes exactly as they appear rather than translating {Enter} to an ENTER keystroke, ^c to Control-C, etc. Although the string {Raw} need not occur at the beginning of the string, once specified, it stays in effect for the remainder of the string.

2. Repeating or Holding Down a Key

To repeat a keystroke: Enclose in braces the name of the key followed by the number of times to repeat it. For example:

```
Send {DEL 4}  ; Presses the Delete key 4 times.
Send {S 30}   ; Sends 30 uppercase S characters.
Send +{TAB 4}  ; Presses Shift-Tab 4 times.
```

To hold down or release a key: Enclose in braces the name of the key followed by the word **Down** or **Up**. For example:

```
Send {b down}{b up}
Send {TAB down}{TAB up}
Send {Up down}  ; Press down the up-arrow key.
Sleep 1000  ; Keep it down for one second.
Send {Up up}  ; Release the up-arrow key.
```

When a key is held down via the method above, it does not begin auto-repeating like it would if you were physically holding it down (this is because auto-repeat is a driver/hardware feature).

The word *DownTemp* may also be used. Its effect is the same as *Down* except for the modifer keys (Control/Shift/Alt/Win). In those cases, *DownTemp* tells subsequent sends that the key is not permanently down, and may be released whenever a keystroke calls for it. For example, *Send {Control DownTemp}* followed later by *Send a* would produce a normal "a" keystroke, not a control-A keystroke.

3. General Remarks

In addition to letters A through Z, the following letters and symbols are also supported (however, if your system's code page is something other than 1252 [US & Western Europe], this list might be different):
€❶‚ƒ„…†‡ˆ‰Š‹Œ⇋ŽƇɑʊ‘’“”•—˜™š›œɔžŸ ¡¢£¤¥¦§¨©ª«¬®¯°±²³´µ¶·¸¹º»¼½¾¿
ÀÁÂÃÄÅÆÇÈÉÊËÌÍÎÏÐÑÒÓÔÕÖ×ØÙÚÛÜÝÞßàáâãäåæçèéêëìíîïðñòóôõö÷øùúûüýþÿ

Since the operating system does not allow simulation of the CTRL-ALT-DELETE combination, doing something like *Send ^!{Delete}* will have no effect.

When SendInput sends mouse clicks by means such as {Click}, and *CoordMode Mouse, Relative* is in effect (the default), every click will be relative to the window that was active at the start of the send. Therefore, if SendInput intentionally activates another window (by means such as alt-tab), the coordinates of subsequent clicks within the same command will be wrong because they will still be relative to the old window rather than the new one.