AutoHotkey 版「かんざし」Version 0.3α　取扱説明書

2014 年 11 月 23 日

「かんざし」は、パソコン上の複数の辞書ソフトや Web 検索サイトを一括検索するための検索支援ツール（いわゆる串刺し検索ツール）です。それぞれの辞書ソフトやサイトに検索語句を入力する手間が省けます。

この文書では、AutoHotkey（AHK）版かんざし Version 0.3α の基本的な使い方、設定方法、スクリプトのカスタマイズ方法について説明します。

むやみやたらと長い文書ですが、隅から隅までまるっとお読みいただく必要はありません。目的に応じて、次のようにご利用ください。

- 基本的な使い方は「1 概要」「2 導入」「3 基本操作」までで ひととおり説明しています。
  とりあえず そこまでは通読していただくことをお勧めします。
- 設定を変更する場合は、「4 スクリプト修正のヘルパー機能」を読んでから、「6 目的別設定リファレンス」を参照のうえ、「5 設定変更の方法」の必要箇所をお読みください。
- AHK のプログラミングで細かなカスタマイズを加えたい場合は、「7 AHK スクリプトによるカスタマイズ」をお読みください。サンプルスクリプトやヘルパー関数の説明があります。

目次:

1　概要 3

AHK 版かんざしの特徴 3

使用上の注意 5

利用条件・免責事項 5

作者連絡先 5

2　導入 6

インストール 6

起動と終了 6

とりあえず検索してみよう 8

3　基本操作 10

入力ウィンドウ 10

ポップアップメニュー 11
ホットキー一発検索 12
検索したウィンドウのアクティブ化 12
その他 12
4 スクリプト修正のヘルパー機能 14
ホットキー設定ヘルパー 14
「Ctrl+Alt+中ボタン」 14
5 設定変更の方法 16
スクリプトの構成 16
GUI の設定 18
ブラウザの設定・自動起動するソフトの設定 20
辞書グループの設定 22
各辞書グループの検索先の設定 24
ホットキーの設定 31
ショートカットキーの設定 32
6 目的別 設定リファレンス 33
7 AHK スクリプトによるカスタマイズ 34
カスタマイズの概要 34
ホットキー部のカスタマイズの例 35
ショートカットキー部のカスタマイズの例 38
DictGrpXX() のカスタマイズの例 40
ヘルパー関数リファレンス 44

1　概要

AHK 版かんざしの特徴

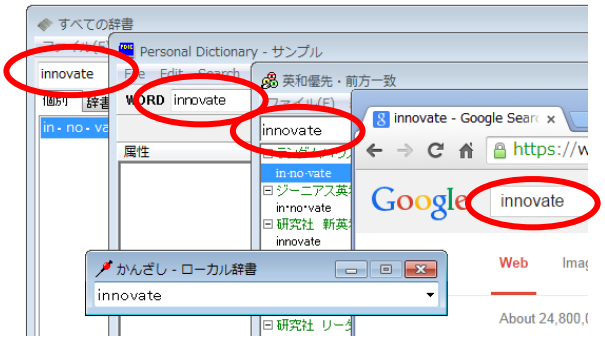
AutoHotkey（AHK）版「かんざし」Version 0.3 は、2001 年で更新が止まっていた串刺し検索ツール「かんざし」Version 0.2 の後継にあたるバージョンです。v0.2 は EXE ファイルでの公開でしたが、v0.3 は AHK 用のスクリプト（ソースコード）の形で公開しています。AHK を導入済みの Windows 環境でお使いください。

以下、本書では、以前のバージョンのかんざしのことを「スタンドアロン版かんざし」と表現します。

AHK 版かんざしの主な特徴は次のとおりです。

さまざまな辞書ソフトや Web サイトを串刺し検索

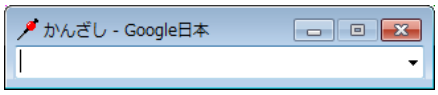
複数の辞書ソフトや Web サイトを 1 回の操作でまとめて検索でき、語句入力の手間を省きます。



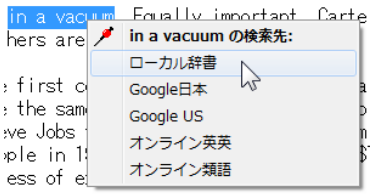
3 種類のインターフェイスを標準装備

3 種類のインターフェイスを使い分けることで、状況と目的に合った検索をすばやく実行できます。

- 入力ウィンドウ



- ポップアップメニュー



- ホットキー一発検索

（語句を選択した状態で特定のキーを押すことで検索を実行する機能です。Ctrl+C で語句をコピーできるアプリケーションのほとんどで動作します）

Web 検索機能と辞書グループ機能を強化

v0.2 に比べ、Web 検索機能と辞書グループ機能を強化しました。Web 上の検索サイトや辞書サイトを検索先として追加できます。Windows 上の辞書ソフトと Web 上の検索先を同じ辞書グループに混在させることも可能です。また、検索結果の URL に検索語句が含まれないタイプの Web サイトも検索可能です。

検索作業を効率化

アプリケーション上で選択中の語句を入力ウィンドウに取り込んだり、Web 検索した語句を引用符囲みに変換して再検索したりなど、検索作業時のキー入力と手間を減らすことができます。

また、Web 検索の結果をブラウザのどのタブで開くかを指定できる機能や、検索先の辞書ソフトを自動で起動してから検索する機能も備えています。

プログラミングの知識がなくても大丈夫

通常、AHK のスクリプトで辞書検索や Web 検索を行うためには、プログラミングの知識が必要です。しかし AHK 版かんざしは、プログラミングをご存知ない方でも、設定変更や基本的なカスタマイズを行えるようになっています。テキストエディタさえ使えれば、あとは単なるコピペや、ごく簡単な記述のみで大丈夫です。

スタンドアロン版かんざしの「★のドラッグ &ドロップ」に代わる、「Ctrl+Alt+中ボタン」という機能もあります。

プログラミングの知識を生かした細かなカスタマイズも可能

プログラミングをご存知の方は、かんざしのヘルパー関数と、AHK の組み込み関数・コマンド・ロジックを組み合わせて、単なるコピペにとどまらない、細かなカスタマイズや機能拡張を加えることができます。

たとえば、独自の検索処理のスクリプトを辞書グループの検索先として追加したり、選択語句の種類に応じてホットキー一発検索の検索先を自動で切り替えたり、かんざしを簡易ランチャーとして使ったりなど、さまざまな拡張が可能です。

本書の後半には、こうしたカスタマイズのサンプルプログラムや、ヘルパー関数のリファレンスがあります。

従来のかんざしから簡単に移行

スタンドアロン版かんざし（v0.2）の動作環境をそのまま AHK 版かんざしに移行するためのツールを用意しました。**（※α 版の段階では未公開です）**

また、スタンドアロン版よりカスタマイズ性が高まっていますので、これまで検索できなかった辞書ソフトにも対応できる可能性があります。

使用上の注意

当ツールを使うときには、以下の点にご注意ください。

- 検索処理の動作中は、キーボードやマウスの操作を控え、なるべく静かにお待ちください。プログラムで制御した擬似的なキーボード操作やマウス操作を Windows 上のアプリケーションに送り込んでいますので、外から不用意に操作を加えると、予期せぬ動作となる可能性があります。
- 場合によっては、辞書ソフトやブラウザのウィンドウ内で特定の位置にあるコントロールに対してキーボード操作やマウス操作を行うことがあります。したがって、ウィンドウの表示項目の配置やウィンドウのサイズが変わると、誤動作を引き起こす可能性があります。
- 設定変更やカスタマイズでスクリプトに手を加えるときに、修正を誤ると、正しく動作しなくなる可能性があります。元に戻せるよう、ファイルのバックアップをとったうえで修正を加えることをお勧めします。
- 設定変更やカスタマイズを加えたスクリプトは、そのパソコン上でのみお使いください。たとえ同じ辞書ソフトやブラウザを導入していても、別の環境で同じように動作するとは限りません。

利用条件・免責事項

利用条件

当ツールはフリーソフトウェアです。無償で自由にご利用いただけます。
当ツールの著作権は、作者の内山卓則にあります。

免責事項

当ツールを利用されたことによりトラブル・損害等が発生した場合でも、作者の内山卓則は一切の責任を負いません。利用者各自の責任においてご利用ください。

α 版・β 版について

α 版・β 版の段階では、大きな不具合が潜んでいる可能性もないとは限りません。お仕事などの重要なファイルを開いていない環境で、様子を見ながら少しずつお試しいただくよう、お願いいたします。

また、正式版までの間に、機能やインターフェイスに変更が加わる可能性があります。α 版やβ 版の段階でカスタマイズしたスクリプトに修正が必要になるかもしれませんが、ご了承ください。

作者連絡先

ご意見・ご感想・ご要望などございましたら、作者の内山までメールでお寄せ下さい。
アドレスは uchiyama ＠uchi-com.jp です。

2　導入

インストール

動作環境

AHK 版かんざしを使うためには、次の環境が必要です。

- Windows Vista / 7 / 8 / 8.1（各 32bit / 64bit 版）
- AutoHotkey Unicode 版 Version 1.1.13.00 以降
  http://ahkscript.org からインストーラーをダウンロードし、手順に従ってインストールしてください。

※作者は、Windows 7 Professional 64bit 上の AHK Unicode 64bit v1.1.15.04 でのみ動作を確認しています。

かんざしのインストール

zip ファイルの中身一式をお好きなフォルダに展開したうえで、「Kanzasi--sample.ahk」というファイルの名前を「Kanzasi.ahk」に変更してください。
この Kanzasi.ahk がメインのスクリプトです。設定変更などの際は、このファイルを修正することになります。

それ以外のファイルは、同じフォルダ内に置いてあるだけで OK です。

※従来のスタンドアロン版かんざしをお使いの場合、展開先はそれとは別フォルダで問題ありません。
スタンドアロン版かんざしの削除も不要です。

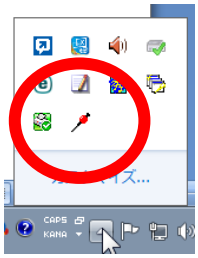
かんざしのアンインストール

特別な手順は必要ありません。展開したフォルダごと削除してください。

起動と終了

起動方法

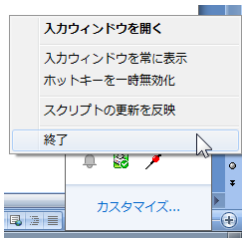
エクスプローラー上で Kanzasi.ahk をダブルクリックしてください。Windows のタスクトレイにかんざしのアイコンが表示されたら、正しく起動できています。



スタンドアロン版かんざしとは違い、デフォルトでは起動直後に入力ウィンドウが表示されません。

終了方法

タスクトレイでかんざしのアイコンを右クリックし、メニューで「終了」をクリックしてください。



スタンドアロン版かんざしとは違い、入力ウィンドウを閉じただけではプログラムは終了しません。

設定の変更方法

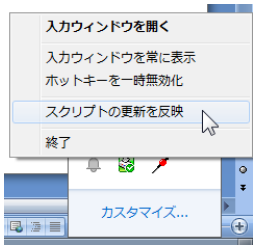
スタンドアロン版かんざしのような設定ウィンドウはありません。設定の変更や検索先の追加/削除などは、Kanzasi.ahk のスクリプトをテキストエディタで開いて、直接修正を加えていただく形となります。

基本的な設定変更には、プログラミングの知識は必要ありません。修正箇所や方法は、「5 設定変更の方法」の中で説明します。

※同じフォルダにある Kanzasi_lib.ahk というファイルは、かんざしの内部処理を「ブラックボックス化」したファイルです。通常は修正の必要はありませんので、手を加えずにそのままお使いいただくことをお勧めします。（バージョンアップの際に更新する可能性があるためです）。

設定の変更を反映する方法

かんざしを起動したままの状態でスクリプトを修正した場合、スクリプトファイルを保存しただけでは変更内容が反映されません。タスクトレイでかんざしのアイコンを右クリックし、メニューで「スクリプトの更新を反映」をクリックしてください。



※あるいは、エクスプローラー上で Kanzasi.ahk を再度ダブルクリックするというやり方でも、更新を反映できます。

とりあえず検索してみよう

検索先のローカル辞書ソフトを登録して、検索を試してみましょう。

※以下の手順では、Windows 上の辞書アプリケーションのみを登録してください。(Web ブラウザで開く検索サイトや辞書サイトは、この登録方法では検索できません)。

なお、この後の作業で「マウスの中ボタンをクリック」とあるのは、マウスのホイール部分をクリックして押し下げることです。ホイールの回転とは異なります。

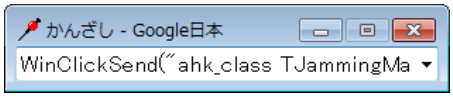
ローカル辞書ソフトの追加

ここで行うのは、スタンドアロン版かんざしの「★のドラッグ &ドロップ」に相当する作業です。

(1) Kanzasi.ahk をダブルクリックして、AHK 版かんざしを起動してください。

(2) 検索対象の辞書ソフトを起動してください。

(3) 辞書ソフトの語句入力エリアの上で、Ctrl キーと Alt キーを押しながらマウスの中ボタンをクリックしてください。かんざしの入力ウィンドウがポップアップし、クリック先の辞書ソフトを検索するためのコードが次のような形で表示されます。



(4) 表示された内容全体をクリップボードにコピーしてください(Ctrl+A で全選択し、Ctrl+C でコピー)。

(5) Kanzasi.ahk をテキストエディタで開いてください。

(6) ファイルの中盤にある次のような行を見つけてください。ファイル内を「DictGrp10」で検索すると見つかります。(エディタ上で実際に青色の表示になっているわけではありません)。

```
DictGrp10() ; ローカル辞書
{
    ; ★辞書ソフトの入力エリア上で「Ctrl+Alt+中ボタン」を押し、表示されたコードをここに貼り付け★
}
```

(7) 上記の青色の部分に、先ほどコピーした内容を上書きで貼り付けてください。

例:

```
DictGrp10() ; ローカル辞書
{
    WinClickSend("ahk_class TJammingMainForm", "TRichEdit5", "+^a^v{Enter}") ; 「英英優先・前方一致」, "x494 y67"
}
```

※行頭を半角スペース 4 つで字下げすると他の行と揃います。(なくても動作は変わりません)。

(8) 検索したい辞書ソフトが 2 つ以上ある場合は、それぞれの入力エリア上で「Ctrl+Alt+中ボタン」で取得したコードを、1 行に 1 つずつ、並べて貼り付けてください。

例:

```
DictGrp10() ; ローカル辞書
{
   WinClickSend("ahk_class TJammingMainForm", "TRichEdit5", "+^a^v{Enter}") ;「英英優先・前方一致」, "x494 y67"
   WinClickSend("PASORAMA", "Edit1", "{Home}+{End}^v") ; "ahk_class Afx:400000:b:10005:6:4037d", "x1183 y248"
   WinClickSend("ahk_class TPdicMain.UnicodeClass", "Edit1", "^a^v") ;「Personal Dictionary - サンプル」, "x352 y57"
}
```

(9) Kanzasi.ahk を保存してください。

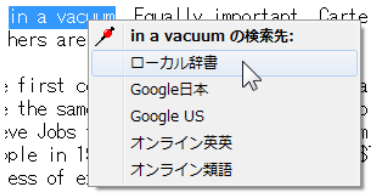
(10) タスクトレイでかんざしのアイコンを右クリックし、「スクリプトの更新を反映」をクリックしてください。

※これ以降、操作手順の説明の中では、「更新の反映」には触れませんが、スクリプトに修正を加えたときには最後に必ず実行してください。

検索の動作確認

いま登録した辞書ソフトを検索できるかどうか、ポップアップメニュー検索で試してみましょう。

テキストエディタ、Webブラウザ、Wordなどのアプリケーション上で、何か語句を選択し、マウスの中ボタンをクリックしてください。その語句を検索するポップアップメニューが表示されますので、「ローカル辞書」をクリックしてください。



うまく検索できれば、辞書ソフトの登録は完了です。

※検索先の辞書ソフトを起動した状態でお試しください。

※以前のスタンドアロン版かんざしと同じで、登録した辞書ソフトを必ずしも検索できるとは限りませんが、貼り付けたコードに修正を加えれば、正しく検索できるようになる可能性があります。その方法については後で説明します。

重要:貼り付けたコードで、カッコ内の 2 番目の項目が "x〇〇 y〇〇" という形式の場合、入力エリアの位置をウィンドウ左上隅からの座標で指定する形となっています。この場合、その辞書ソフトのウィンドウ内で入力エリアの位置が変わると、検索時に予期せぬ動作を起こす可能性が高いですので、ウィンドウのサイズや表示項目の配置を不用意に変更しないようご注意ください。

例:

```
WinClickSend("ahk_class TMainForm", "x158 y624", "{Home}+{End}^v{Enter}") ; ...
                                    ↑ここが座標指定の場合は注意！
```

3　基本操作

ここでは、デフォルト状態での機能と操作について説明していきます。
キーや検索先などはすべて設定変更が可能です。

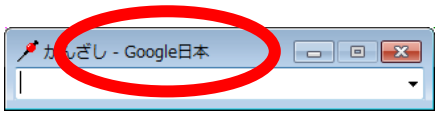
入力ウィンドウ

ウィンドウの表示

入力ウィンドウを表示する方法は次の 2 つです。

- ホットキー「Ctrl+Shift+K」
- タスクトレイアイコンをダブルクリック

検索語句を入力して Enter キーを押すと、辞書ソフトや Web サイトを検索できます。
検索先の辞書グループはタイトル部分に表示されています。



辞書グループの切り替え

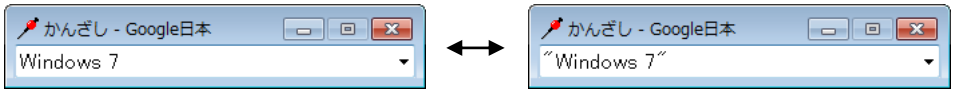
辞書グループを切り替えるショートカットキーは次のとおりです。

ショートカットキー	検索先の辞書グループ
F1、Ctrl+D	ローカル辞書
F2、Ctrl+G	Google 日本
F3、Ctrl+U	Google US
F4、Ctrl+E	オンライン英英辞典（Oxford、Merriam-Webster、Onelook）
F5、Ctrl+R	オンライン類語辞典（類語玉手箱、Weblio 類語）
Ctrl+↑、Ctrl+↓	グループを 1 つずつ切り替え

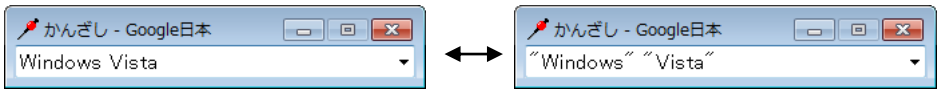
※これらのキーは、入力ウィンドウがアクティブな状態でのみ有効です。

その他の主なキー操作

- Ctrl+2：入力語句全体の " " 囲みのオン / オフ



- Ctrl+3：入力語句１つ１つの " " 囲みのオン / オフ

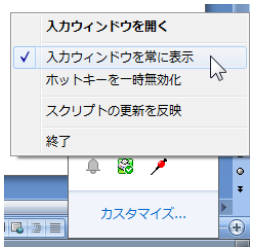


- [↓][↑]：検索履歴の表示
- Esc：ウィンドウ消去

※これらのキーも、入力ウィンドウがアクティブな状態でのみ有効です。

入力ウィンドウの常時表示

デフォルトでは、入力ウィンドウはホットキーなどで呼び出したときのみ表示され、検索を実行したら消える設定となっています。以前のスタンドアロン版かんざしのように、入力ウィンドウが常に表示されているスタイルがお好みの場合は、タスクトレイアイコンを右クリックし、「入力ウィンドウを常に表示」をクリックしてください。



ウィンドウの位置とサイズの設定

入力ウィンドウの位置とサイズを変更しても、再起動すると元の状態に戻ります。これは、スクリプトの設定を変更することで記憶可能です。また、「常に表示」モードをデフォルトでオンにすることもできます。
これらの修正方法については、「GUI の設定」セクションで説明します。

ポップアップメニュー

選択中の語句を検索するためのポップアップメニューを表示する方法は次の 2 つです。

- ホットキー「Ctrl+Shift+I」
- マウスの中ボタンをクリック

※マウスの中ボタンは、語句を選択していないときはデフォルトの動作をします。たとえば、一般的なWebブラウザには、リンクを中ボタンでクリックすると新しいタブで開くという機能がありますが、語句選択していない状態のときにはこうした機能を通常どおり使えます。

ホットキー一発検索

アプリケーションで選択した語句を、メニュー選択や入力なしで一発で検索できます。
デフォルトで登録してあるホットキーは次の 2 つです。

- ホットキー「Ctrl+Shift+D」:ローカル辞書を一発検索
- ホットキー「Ctrl+Shift+G」:Google 日本を一発検索

また、語句を選択していない状態で一発検索のホットキーを押すと、前回の検索語句と同じとみなして検索を実行します。たとえば、ローカル辞書で語句を検索したらヒットしなかったので引き続き Google で検索する、という場合、改めて語句を選択し直さなくても、「Ctrl+Shift+G」を押すだけで大丈夫です。

※「前回の検索語句」というのは、入力ウィンドウやポップアップメニューによる検索も含みます。

検索したウィンドウのアクティブ化

検索語句を送信した辞書ソフトや Web ブラウザのウィンドウを順番にアクティブ化できる機能です。

- ホットキー「Ctrl+Shift+N」:正順でアクティブ化
- ホットキー「Ctrl+Shift+P」:逆順でアクティブ化

「正順」は、最近検索した辞書グループで先頭にあるものから順にアクティブ化します。「逆順」はその逆です。

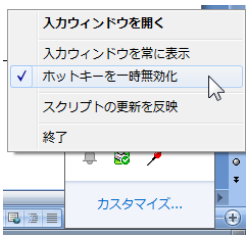
※スタンドアロン版かんざしの同様の機能とは違い、既に検索語句を送信した辞書ソフトだけがアクティブ化の対象となります。(したがって、検索を実行していない段階でこのキーを押しても何も起きません)。

※Web ブラウザはウィンドウ単位でのアクティブ化です。1 つ 1 つのタブまではアクティブ化しません。

その他

ホットキーの一時無効化

タスクトレイアイコンを右クリックし、「ホットキーを一時無効化」を選択すると、入力ウィンドウ表示・ポップアップメニュー表示・一発検索など、ホットキーをすべて無効にできます。



この状態でも、タスクトレイアイコンから入力ウィンドウを開けば、検索を実行できます。

※入力ウィンドウ上のショートカットキーも、グループ選択のキー以外は無効になります。

設定しておくと便利な機能

以下の 2 つの機能は、デフォルトでは設定してありませんが、設定しておくと便利です。
まず「Ctrl+Alt+中ボタン」機能の説明をお読みのうえ、それぞれのリンク先の説明に従って設定してください。

- 検索対象の辞書ソフトやブラウザを自動起動する機能（設定方法はこちら）
- Web 検索結果をブラウザのどのタブで開くかを指定する機能（設定方法はこちら）

AHK 版かんざしにはない検索機能

スタンドアロン版かんざしにあった以下の検索機能は、AHK 版かんざしにはありません。

- 「クリップボードの語句を検索」機能
- 「クリップボード変更時に自動検索」機能
- Kanzasi.exe の起動パラメータで検索する機能

以上で、基本的な説明はひととおり終了です。これ以降は、必要な部分のみ拾い読みしていただければと思います。

ホットキーや Web 検索先などの設定を変更する場合は、「4 スクリプト修正のヘルパー機能」を読んでから、「6 目的別設定リファレンス」を参照のうえ、「5 設定変更の方法」の該当箇所をお読みください。

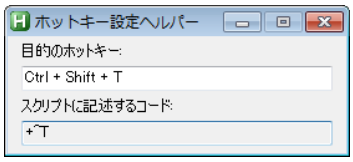
4　スクリプト修正のヘルパー機能

次の章から、スクリプトの修正方法について説明していきますが、その前に、修正作業全般で利用する 2 つのヘルパー機能について説明します。

ホットキー設定ヘルパー

スクリプトの中には、ホットキーやショートカットキーを指定する箇所がいくつかあります。こうした所では、たとえば Ctrl+V は「^v」、Alt+F1 は「!F1」というように、AHK で定められた書式に従って、キーの組み合わせを表す文字列を記述する必要があります。

AHK 版かんざしに同梱の「ホットキー設定ヘルパー」を使うと、キーの記述方法を簡単に確認できます。



使い方は簡単です。

(1) エクスプローラーで、Kanzasi.ahk と同じフォルダにある「ホットキー設定ヘルパー.ahk」をダブルクリックしてください。上のような画面が表示されます。

(2) 「目的のホットキー」の欄で、指定したいホットキーを押してください。
下の欄に、そのキーを表すコードが表示されます。

スクリプトの中でキーを指定する部分には、このコードを貼り付けてください。

ただしこのツールでは、一部の特殊なキーや、マウスのボタンを使うホットキーのコードは確認できません。
http://ahkscript.org/docs/KeyList.htm か、AHK のオンラインヘルプでご確認ください。

※このツールで表示されるコードでは、a～z の大文字・小文字が、Kanzasi.ahk のデフォルトの記述と異なる場合がありますが、通常はどちらでも問題ありません。

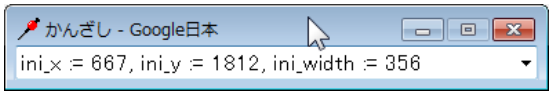
「Ctrl+Alt+中ボタン」

冒頭で行ったローカル辞書ソフトの登録もそうでしたが、スクリプトを修正するときには、「Ctrl+Alt+中ボタン」で表示されたコードを貼り付けるという形で対応する部分がいくつかあります。

この「Ctrl+Alt+中ボタン」には、次のような機能があります。

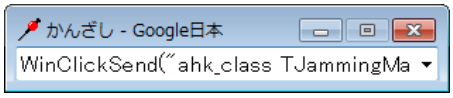
入力ウィンドウ上での「Ctrl+Alt+中ボタン」

入力ウィンドウ上で「Ctrl+Alt+中ボタン」をクリックすると、ウィンドウの x / y 座標と幅の情報が表示されます。

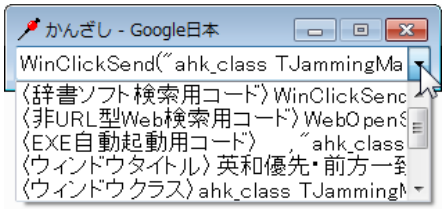


「Ctrl+Alt+中ボタン」の後のドロップダウンリスト

アプリケーションのウィンドウや入力エリアの上で「Ctrl+Alt+中ボタン」すると、ローカル辞書ソフトとして検索するためのコードが表示されます。これは冒頭のローカル辞書登録で行ったとおりです。



ここで、入力ウィンドウの右端の[▼]ボタンをクリックすると、ドロップダウン部分に、別のコードやデータがいくつか入っています。スクリプトの中では、こうしたコードやデータを貼り付けて使う箇所もあります。



リスト内の項目をクリックすると入力エリアに表示されますので、それをコピーして貼り付けてください。
ただし、行頭にある〈 〉囲みの項目名はスクリプトでは不要ですので、その部分を取り除いた形で貼り付けてください。

※これ以降、スクリプトを修正する中でこの作業を行うときには、「辞書ソフトのウィンドウ上で『Ctrl+Alt+中ボタン』し、ドロップダウン部の『〈ウィンドウクラス〉』の内容を貼り付け」といった形で、簡略化して説明します。

ドロップダウンリストに入っている項目は次のとおりです。

項目	意味
辞書ソフト検索用コード	ローカル辞書ソフトに検索語句を送信するコード （入力エリアに表示されるコードと同一）
非 URL 型 Web 検索用コード	URL に検索語句が含まれないタイプのサイトを検索するためのコード
EXE 自動起動用コード	自動起動させる辞書ソフトやブラウザを設定するためのコード
ウィンドウタイトル	マウス位置のウィンドウのタイトル部の内容
ウィンドウクラス	マウス位置のウィンドウのクラス*
マウス位置相対座標	ウィンドウの左上を基点とした、マウスの位置
マウス位置クラス	マウス位置のコントロールのクラス*
EXE フルパス	マウス位置のウィンドウの EXE ファイルのフルパス

* 「クラス」とは、プログラムの中でウィンドウやコントロールを指定するための識別子の一種です。

5　設定変更の方法

AHK 版かんざしには設定ウィンドウがありません。設定の変更や検索先の追加/削除などは、Kanzasi.ahk のスクリプトをテキストエディタで開いて、直接修正を加えていただく形となります。

ここからは、スクリプトの修正方法について説明していきます。

スクリプトの構成

スクリプト全体は次のような構成となっています。設定ウィンドウでいうタブページのようなイメージで、設定内容ごとにセクションが分かれていると捉えていただくとよいと思います。

```
; AutoHotkey 版「かんざし」Version 0.3α

; このファイルをダブルクリックすると処理が起動します。
; このファイルは自由にカスタマイズしてご利用ください。
; Kanzasi_lib.ahk のほうは手を加えずお使いいただくことをお勧めします。

#Include %A_ScriptDir%¥Kanzasi_lib.ahk

Kanzasi_Initialize:

SendMode Event
SetKeyDelay, 0

;■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■
;■■■■■■■■■■■■■■■■ ＧＵＩの設定 ■■■■■■■■■■■■■■■■
;■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■

ini_x := -100, ini_y := 0, ini_width := 300

ini_stayopen       := false
ini_stayontop      := true
ini_fontname       := "ＭＳ Ｐゴシック"
ini_fontsize       := 13
ini_activatelstwin := true

;■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■
;■■■■■■■■■■■■ ブラウザ・自動起動ソフトの設定 ■■■■■■■■■■■■
;■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■

ini_browserwin := ""

ini_winexe := { "" : ""
    , "この行にコードを貼り付け" : "" }

;■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■
;■■■■■■■■■■■■ 辞書グループの名前とキーの設定 ■■■■■■■■■■■■
;■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■

ini_grpid := { 10: "ローカル辞書    = (^d), (F1), ((^+d))"
             , 20: "Google 日本     = (^g), (F2), ((^+g))"
             , 30: "Google US       = (^u), (F3)"
             , 40: "オンライン英英 = (^e), (F4)"
             , 50: "オンライン類語 = (^r), (F5)" }

KanzasiCommonInit()

return

;■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■
;■■■■■■■■■■■■ 辞書グループごとの検索先の設定 ■■■■■■■■■■■■
;■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■

DictGrp10()
{
    ; ★辞書ソフトの入力エリア上で「Ctrl+Alt+中ボタン」を押し、表示されたコードをここに貼り付け★
}

DictGrp20()
{
    WebOpen(0, "http://www.google.co.jp/search?hl=ja&gl=jp&q=%s")
}

DictGrp30()
{
    WebOpen(0, "http://www.google.com/search?hl=en&gl=us&q=%s")
}

DictGrp40()
{
    WebOpen(0, "http://www.oxforddictionaries.com/search/english/?direct=1&multi                    ")
    WebOpen(0, "http://www.merriam-webster.com/dictionary/%s")
    WebOpen(0, "http://www.onelook.com/?w=%s&ls=a")
}

DictGrp50()
{
    WebOpen(0, "http://ruigo-tamatebako.jp/find/index?q=%s&m=match")
    WebOpen(0, "http://thesaurus.weblio.jp/content_find?query=%s&searchType=exact")
}
```

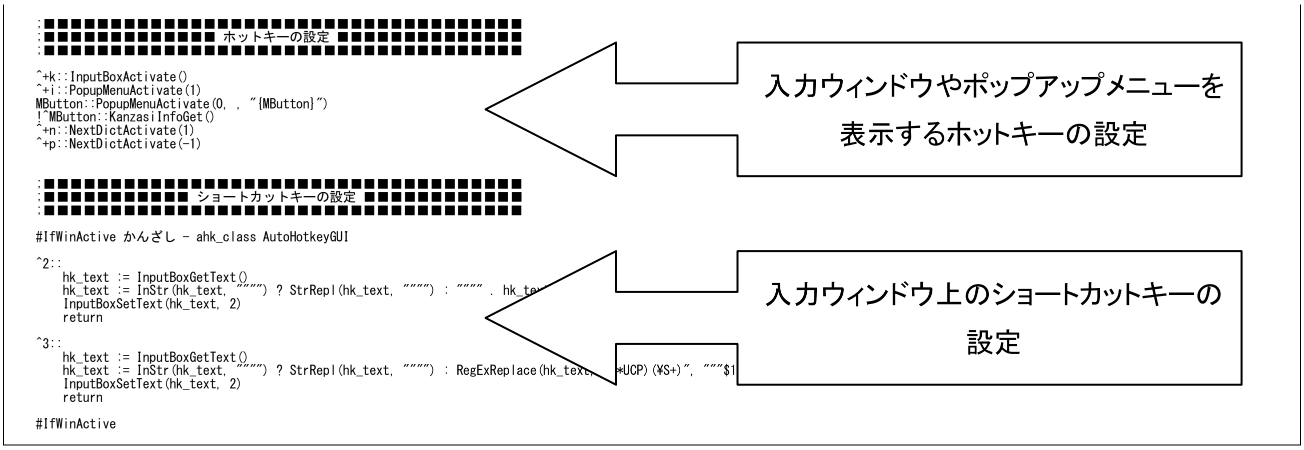
GUI の設定

Web ブラウザの設定
自動起動する EXE ファイルの設定

辞書グループの名前・ショートカットキー・ホットキーの設定

各辞書グループの検索先の設定

（続く）



以下、それぞれのセクションの設定内容について順に説明していきます。細かい話が長々と続きますので、「目的別 設定リファレンス」を参照のうえ、必要な箇所のみお読みいただくとよいかもしれません。

なお、スクリプトに手を加えるときには、修正を誤ると正しく動作しなくなる可能性があります。すぐに元に戻せるよう、Kanzasi.ahk のバックアップをとってから修正することをお勧めします。

GUI の設定

入力ウィンドウの表示などに関する設定を行うセクションです。

デフォルト設定

```
ini_x := -100, ini_y := 0, ini_width := 300

ini_stayopen      := false
ini_stayontop     := true
ini_fontname      := "MS Pゴシック"
ini_fontsize      := 13

ini_activate1stwin := true
```

設定項目

入力ウィンドウのデフォルトの位置とサイズ

項目	ini_x, ini_y, ini_width
指定値	入力ウィンドウの上で「Ctrl+Alt+中ボタン」し、表示されたコードをそのまま上書きで貼り付け
説明	実行中に入力ウィンドウの位置とサイズを変更しても、自動では記憶されず、再起動するとデフォルトの状態に戻ります。その「デフォルトの状態」を指定するための場所がここです。 入力ウィンドウを目的の位置とサイズに調整したうえで「Ctrl+Alt+中ボタン」し、表示された内容を貼り付けてください。 なお、位置に関しては、特別な設定値として次の 2 つがあります。 ・`ini_x := -100` …入力ウィンドウを画面中央に表示 ・`ini_x := -200` …入力をアクティブウィンドウの中央に表示 （これらの値を指定するときは、ini_y の値は無視されます）。

「入力ウィンドウを常に表示」をデフォルトでオンにするかどうか

項目	ini_stayopen
指定値	true / false
説明	実行中にタスクトレイで設定を変更できる項目ですが、それを起動時点でオンにするかどうかをここで指定します。

入力ウィンドウを常に前面表示するかどうか

項目	ini_stayontop
指定値	true / false

入力ウィンドウのフォントの種類

項目	ini_fontname
指定値	フォント名を " " で囲んで指定

入力ウィンドウのフォントのサイズ

項目	ini_fontsize
指定値	フォントサイズを表す数値

送信完了後に先頭の辞書をアクティブ化するかどうか

項目	ini_activate1stwin
指定値	true / false
説明	false の場合は、送信前にアクティブだったウィンドウに戻ります。 （ただし作者の環境では、入力ウィンドウが閉じた直後になぜかフォーカスが別のウィンドウに移ってしまうことがあり、false の場合には予期したウィンドウとは異なるウィンドウに戻ってしまうことがあります）。

ブラウザの設定・自動起動するソフトの設定

Web 検索に使うブラウザと、未起動時に自動起動する辞書ソフト類を指定するセクションです。

設定例

```
ini_browserwin := "ahk_class MozillaWindowClass"

ini_winexe := { "" : ""
    , "ahk_class TJammingMainForm" : "C:¥Program Files (x86)¥Jamming¥Jamming.exe,1000" }
```

設定項目

Web 検索で使用するブラウザ（設定を推奨）

項目	ini_browserwin
指定値	ブラウザのウィンドウ上で「Ctrl+Alt+中ボタン」し、ドロップダウンの「〈ウィンドウクラス〉」の内容を "" で囲んで貼り付け
説明	デフォルトでは、Web 検索の結果は、Windows 上で「既定のブラウザ」として設定しているブラウザの新しいタブで表示されますが、Chrome、Firefox、Internet Explorer の場合は、ここでブラウザを指定しておくことをお勧めします。（別の場所の設定と併用することにより、検索結果をどのタブで開くかを指定できるようになるからです）。

自動起動するソフトの指定

項目	ini_winexe
指定値	対象の辞書ソフトやブラウザのウィンドウ上で「Ctrl+Alt+中ボタン」し、ドロップダウンの「〈EXE 自動起動用コード〉」の内容を貼り付けて、行末に「}」を追加（この後の説明を参照）
説明	語句を送信しようとした時点で辞書ソフトやブラウザが未起動だった場合に、自動で起動させたい場合は、そのソフトをここで指定しておく必要があります。

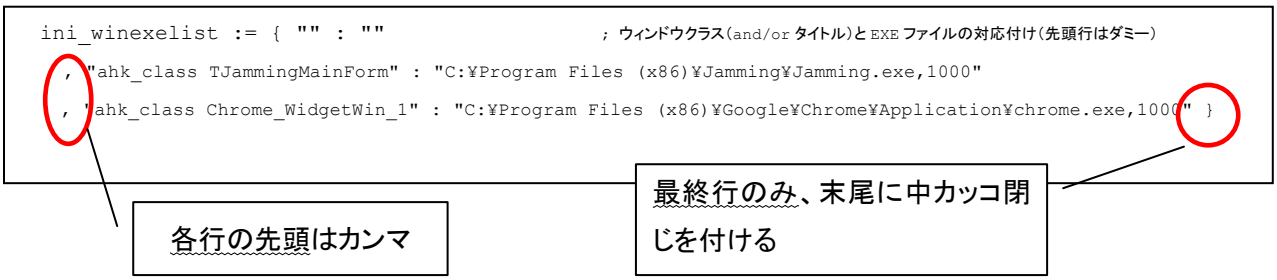
※ini_winexe の指定方法

「Ctrl+Alt+中ボタン」の「〈EXE 自動起動用コード〉」の内容を、下記の青色の部分に貼り付けてください。

```
ini_winexe := { "" : ""                    ; ウィンドウクラス(and/or タイトル)と EXE ファイルの対応付け(先頭行はダミー)
    , "この行にコードを貼り付け" : "" }
```

自動起動させたいソフトが複数ある場合は、それぞれに対する「Ctrl+Alt+中ボタン」の「〈EXE 自動起動用コード〉」を、1 行に 1 つずつ、並べて貼り付けてください。

そのうえで、最後の行の末尾にのみ「}」を加えて、次のような形としてください。



貼り付けたコードの exe のフルパスが間違っていないかどうか、念のためご確認ください。

また、exe のパスの末尾に「,1000」とあるのは、その exe を起動した後で何ミリ秒待ってから検索語句を送るかを表しています。起動の後の初期処理に時間がかかってすぐには検索できないソフトの場合は、ここの数字を大きくしてください。

辞書グループの設定

辞書グループのグループ名・ショートカットキー・ホットキーを設定するセクションです。

デフォルト設定

```
ini_grpid := { 10: "ローカル辞書  = (^d),(F1),((^+d))"
             , 20: "Google 日本  = (^g),(F2),((^+g))"
             , 30: "Google US   = (^u),(F3)"
             , 40: "オンライン英英 = (^e),(F4)"
             , 50: "オンライン類語 = (^r),(F5)" }
```

設定項目

1 行が 1 グループで、「グループ番号: "グループ名 = (ショートカットキー), ((ホットキー))"」という形式です。

- 既存の辞書グループの名前やキーを修正する場合は、該当する箇所を直接書き換えてください。
- 新しい辞書グループを作成する場合は、デフォルトの記述をコピペする形で行を追加し、各項目を書き換えてください。辞書グループはいくつでも作成できます。
- 辞書グループを削除する場合は、行ごと削除してください。

それぞれの項目の内容と指定値は次のとおりです。

グループ番号

内容	スクリプト内でグループを区別するために付ける番号
指定値	正の整数値
説明	他と重複しない値を自由に付けてかまいません。(10 番おきでなくても、2 桁でなくても OK です)。ポップアップメニューのグループ名はこの番号順に表示されます。 また、グループの検索先の指定(次のセクションで説明)の所でも、この番号を使います。

グループ名

内容	入力ウィンドウやポップアップメニューで検索先グループとして表示される名前
指定値	任意の文字列

ショートカットキー

内容	入力ウィンドウ上で辞書グループを選択するためのショートカットキー
指定値	キーを表す文字列を ( ) [半角カッコ 1 つの組]で囲んで指定。 カンマ区切りでいくつでも指定可能
説明	キーの文字列は AHK の書式どおりです。「ホットキー設定ヘルパー」を使うと簡単です。 ショートカットキーが不要なグループは、指定を省略してかまいません。

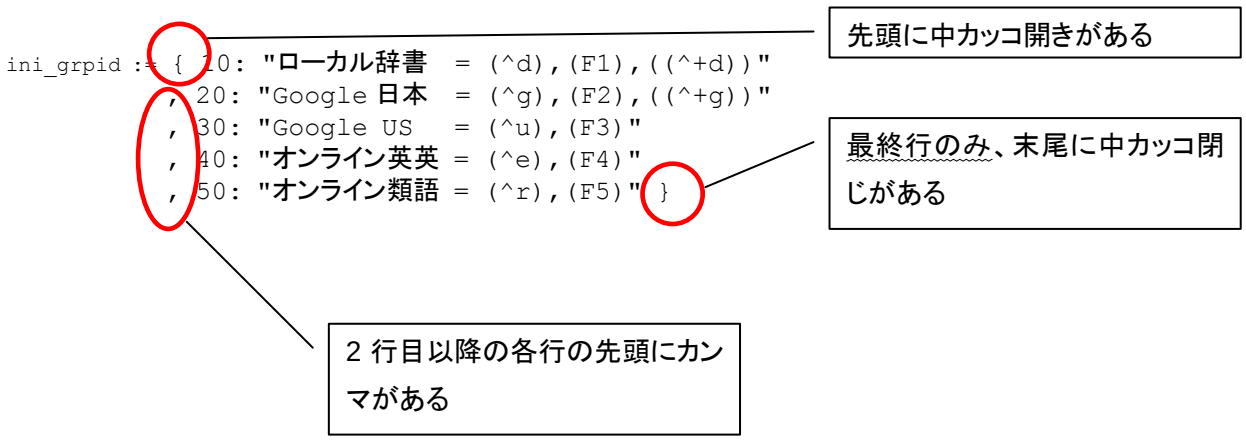
ホットキー

内容	辞書グループを一発検索するためのホットキー
指定値	キーを表す文字列を (( )) [半角カッコ 2 つの組]で囲んで指定。 カンマ区切りでいくつでも指定可能
説明	キーの文字列は AHK の書式どおりです。「ホットキー設定ヘルパー」を使うと簡単です。 ホットキー一発検索が不要なグループは、指定を省略してかまいません。

※ショートカットキーは ( ) で囲み、ホットキーは (( )) で囲みます。ホットキーは幅広いウィンドウで有効なキーだからカッコ (( )) も幅広い、と覚えるとよいかもしれません。
なお、このカッコ囲みの表記法は、かんざしで独自に定めたものであり、この部分でのみ有効です。スクリプト内の他の部分や、一般的な AHK スクリプトでは使えませんのでご注意ください。

記述方法に関する注意・補足

- "グループ名 = (ショートカットキー), ((ホットキー))" という部分は、この部分の文字列全体を " " で囲んでください。(グループ名やキーの 1 つ 1 つの項目を " " で囲む形ではありません)。
- ショートカットキーとホットキーの組み合わせは、すべてのグループを通して一意でなくてはなりません。重複がないように指定してください。
- キー自体の指定に「(」「)」「=」の各文字を使う記述方法は、AHK としては認められていますが、この部分では使えません。代わりに、「+8」「+9」「+-」を使ってください。
- ショートカットキーもホットキーも不要なグループは、たとえば「50: "オンライン類語"」のように、グループ名のみを " " 囲みで指定する形でかまいません。
- 定義の修正が終わったら、次のような形式になっているかどうかご確認ください。



なお、グループを追加した場合は、この後の説明に従って、グループの検索先も指定してください。

各辞書グループの検索先の設定

それぞれの辞書グループに属する辞書ソフトや Web サイトを設定するセクションです。

デフォルト設定

```
DictGrp10() ; ローカル辞書
{
    WinClickSend(....) ; 「Ctrl+Alt+中ボタン」で貼り付けた内容
    WinClickSend(....)
}

DictGrp20() ; Google 日本
{
    WebOpen(0, "http://www.google.co.jp/search?hl=ja&gl=jp&q=%s") ; ※ ...
}

DictGrp30() ; Google US
{
    WebOpen(0, "http://www.google.com/search?hl=en&gl=us&q=%s")
}

DictGrp40() ; オンライン英英
{
    WebOpen(0, "http://www.oxforddictionaries.com/search/english/? ...
    WebOpen(0, "http://www.merriam-webster.com/dictionary/%s")
    WebOpen(0, "http://www.onelook.com/?w=%s&ls=a")
}

DictGrp50() ; オンライン類語
{
    WebOpen(0, "http://ruigo-tamatebako.jp/find/index?q=%s&m=match")
    WebOpen(0, "http://thesaurus.weblio.jp/content_find?query=%s& ...
}
```

検索先の設定の枠組み

辞書グループの定義で指定したグループ番号ごとに、DictGrpXX() というブロックがあり、上下を { } で囲まれた中に、検索先を指定する形となっています。

基本的には、1 つの検索先を 1 行で記述します。同じ検索先を複数のグループに所属させたい場合は、行をコピペして、それぞれの DictGrpXX() の中に記述してください。

「辞書グループの設定」セクションで辞書グループを新たに追加した場合は、その番号の DictGrpXX() のブロックをご自身で作成してください。たとえば、60 番のグループを追加したときは、次のように記述し、その中に検索先を指定していきます。

```
DictGrp60()
{

}
```

また、「辞書グループの設定」で削除した辞書グループについては、DictGrpXX() のブロック全体を削除してかまいません。

※デフォルトでは、ローカル辞書検索とWeb検索の辞書グループが分かれていますが、たとえば次のような形で、1つのグループ内にローカル検索とWeb検索を混在させても問題ありません。

```
DictGrp60()
{
    WinClickSend(....)                                          ; ローカル検索
    WinClickSend(....)                                          ; ローカル検索
    WebOpen(0, "http://www.google.com/search?hl=en&q=%s")       ; Web検索
    WebOpen(0, "http://www.onelook.com/?w=%s&ls=a")             ; Web検索
}
```

検索先の種類

検索先は、主に次の3種類に分かれます。

- **ローカル辞書ソフト**:Windows 上の辞書ソフトやアプリケーションの入力エリアに語句を送信する検索です。(設定方法はこちら)
- **URL 型検索サイト**:Web上の検索サイトや辞書サイトのうち、検索結果のURLの中に検索語句が含まれているタイプのサイトに対する検索です。(設定方法はこちら)
- **非URL 型検索サイト**:Web上の検索サイトや辞書サイトのうち、検索結果のURLの中に検索語句が含まれていないタイプのサイトに対する検索です。できれば、該当の検索ページを開いた時点で、検索語句の入力フィールドにカーソルが置かれるページのほうが好ましいです。(設定方法はこちら)

※スクリプトにデフォルトで登録してあるWeb検索先は、いずれもURL型の検索サイトです。

※非URL型サイトへの検索処理は、多少ぎくしゃくした動きとなり、少し時間がかかります。URL型と非URL型のどちらの方法でも検索できるサイトは、URL型として検索することをお勧めします。

Web検索については、検索結果をブラウザのどのタブで開くかという指定も可能です。(設定方法はこちら)

以下、それぞれの設定方法について説明していきます。

ローカル辞書ソフトの検索先の指定方法

ローカル辞書ソフトを検索先に追加する場合は、検索対象のソフトの入力エリア上で「Ctrl+Alt+中ボタン」し、表示されたコードを、DictGrpXX() のブロックの中に貼り付けてください。

貼り付けたコードで正しく検索できない場合の対処法

辞書ソフトによっては、「Ctrl+Alt+中ボタン」から貼り付けたコードで正しく検索できない場合がありますが、次に示す修正方法により検索できる可能性があります。

◇ インクリメンタルサーチの辞書ソフトの場合

キー入力と並行して検索が実行される「インクリメンタルサーチ」型の辞書ソフトの場合は、最後に送信する Enter キーが不要となります。貼り付けたコードで、カッコ内の末尾にある項目から{Enter}を削除してください。

```
WinClickSend(..., "{Home}+{End}^v{Enter}")
                                  ↑この部分を削除
```

◇ 前の検索語句が残ってしまう場合

前の検索語句が消えずに次の語句が送られている場合は、貼り付けたコードでカッコ内の末尾にある {Home}+{End}^v{Enter} という部分を修正すると直る可能性があります。

```
WinClickSend(..., "{Home}+{End}^v{Enter}")
                    ↑この部分を修正
```

- 入力エリアを Ctrl+A で全選択できる辞書ソフトの場合は、"^a^v{Enter}" に変えてみてください。
- 入力エリアの右クリックメニューに「すべて選択(A)」という項目がある辞書ソフトの場合は、"+{F10}^v{Enter}" に変えてみてください。
- ウィンドウがアクティブ化した時点で入力エリアが全選択状態になる辞書ソフトの場合は、"^v{Enter}" に変えてみてください。

◇ ウィンドウ自体がアクティブにならない場合

ウィンドウ自体がアクティブにならない場合は、貼り付けたコードのカッコ内の先頭項目をウィンドウのタイトルに変更してみてください。(タイトルの一部のみでもかまいません)。
タイトルの内容は、貼り付けたコードの末尾のほうに「 」囲みで記載してあります。

```
WinClickSend("ahk_class olt1UIWindowClass", ... ) ; 「~」
                ↑この部分をウィンドウのタイトルに変更          ↑タイトルはここにあり
                (実際の内容はプログラムにより異なります)
```

※ここをタイトルによる指定に変えた場合、自動起動機能でこの辞書ソフトを起動するためには、ini_winexe の設定も、ここで記述したのと同じタイトルによる指定に変える必要があります。

◇ ウィンドウはアクティブになるがキーが送られない場合

貼り付けたコードでカッコ内の 2 番目にある項目を、入力エリアの座標による指定に変更すると、検索できる可能性があります。
座標は、貼り付けたコードの末尾に "x〇〇〇 y〇〇〇" のような形式で記述してあります。その形式のままコ

ピペしてください。

```
WinClickSend("ahk_class ～", "olt1WindowClass13" ... ) ; 「～」, "x130 y112"
                              ↑この 2 番目の項目を座標に変更          ↑座標はここにあり
                              （実際の内容はプログラムにより異なります）
```

※このように座標を直接指定する形のコードにした場合、辞書ソフトのウィンドウ内で入力エリアの位置が変わったときに予期せぬ動作となる可能性がありますのでご注意ください。

◇ 登録した直後は検索できたが、辞書ソフトを再起動したら検索できなくなった場合

辞書ソフトによっては、送信先のウィンドウと入力エリアを特定するための情報が、起動のたびに変わってしまうものがあります。その場合にこの現象が発生します。

上で説明した「ウィンドウ自体がアクティブにならない場合」「ウィンドウはアクティブになるがキーが送られない場合」の内容にならって、貼り付けたコードのカッコ内の 1 番目の項目をウィンドウのタイトルに、2 番目を入力エリアの座標に、それぞれ変更してみてください。（どちらか一方の修正のみで済む場合もあります）。

◇ その他の場合

以上の方法で修正できない場合、「独自規格の辞書ソフトの検索」というサンプルコードで示すように、辞書ソフト内の語句入力エリアの位置を直接クリックして語句を送り込む独自のコードを記述すれば、検索できる可能性があります。

ヒント：検索オプション等の指定

辞書ソフト側で、辞書グループや検索オプションなどを選択するためのキーが用意されている場合は、貼り付けたコードを修正することによって、そうした選択を行ってから検索を実行することも可能です。
カッコ内の末尾にあるキー送信の部分を修正してください。

例えば、検索ソフト「Jamming」には、ファンクションキーを使って辞書セットを切り替えられる機能があります。（[F2]キーが「すべての辞書」、[F3]以降がユーザー作成の辞書セット）。
こうしたキーで辞書セットを選択したうえで検索を実行したい場合は、次のように記述を追加してください。

```
WinClickSend("ahk_class TJammingMainForm", "TRichEdit5", "F3+^a^v{Enter}") ; ...
                                                          ↑このように目的のキーを追加
```

URL 型検索サイトの検索先の追加

検索結果の URL に検索語句が含まれている「URL 型」の検索先を追加するときには、DictGrpXX() のブロックの中に、「WebOpen(0, "http～")」という形式でコードを記述します。

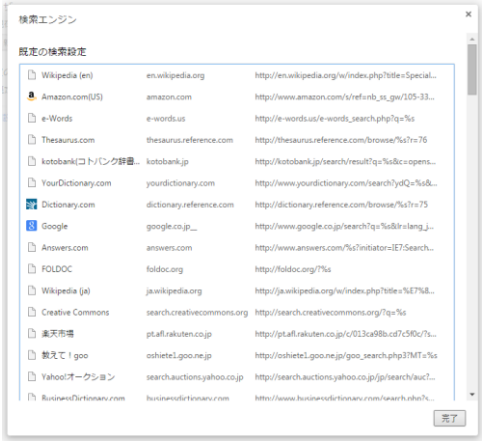
URL の部分は、検索語が入る位置を「%s」で置き換えた形で指定します。目的の Web サイトで、たとえば「sample」といった適当な単語を検索し、検索結果ページの URL をコピペしてから、「sample」とある部分を「%s」に置き換えれば、たいていは動作すると思います。

> %s の部分には、文字コード「UTF-8」で検索語句が挿入されます。もし文字化けが生じる場合、URL に文字コードの設定オプションがあれば、UTF-8 となるよう設定してみてください。

Chrome で「%s」形式の URL を簡単に入手する方法

Google Chrome をお使いの方は、検索語を「%s」に置き換えた形式の URL を、Chrome の設定画面からコピペで簡単に入手できます。

(1) Chrome のアドレスバー（URL を入力するエリア）を右クリックし、「検索エンジンの編集」をクリックしてください。次のような感じの検索エンジン設定画面が表示されます。



(2) この中には、Chrome に設定済みの検索サイトと、これまでに検索したことのある検索サイトや辞書サイトがずらっと並んでいます。目的のサイトを見つけ出してください。

(3) 目的のサイトの行で、右側の URL 欄をクリックし、URL 全体をクリップボードにコピーしてください。

(4) 「WebOpen(0, "http～")」の URL 部分にそのまま貼り付けてください。

ヒント：Google 検索のオプション指定

Google 検索では、たとえば「iphone site:apple.com」のように、検索語句と併せて「site:～」と指定すると、そのサイト内のコンテンツだけを検索できます。

こうして特定のサイトを検索することがよくある場合には、あらかじめ「site:～」を指定した形の URL を検索先として登録しておくと便利です。具体的には、「%s」の後に次の青字のように追加しておけば OK です。

```
DictGrp60()
{
    WebOpen(0, "http://www.google.co.jp/search?hl=ja&gl=jp&q=%s+site:apple.com")
}
```

Google には他にもさまざまな検索オプションがありますが、よく使うものがあれば、同じように登録しておくとよいでしょう。(たとえば、日本語のページのみ検索する場合は「&lr=lang_ja」と追加)。

非 URL 型検索サイトの検索先の追加

検索結果の URL に検索語句が含まれていない「非 URL 型」の検索サイトも、かんざしから検索可能です。できれば、該当の検索ページを開いた時点で、検索語句の入力フィールドにカーソルが置かれるタイプのページのほうが好ましいです。

> たとえば、Microsoft 製品の用語を検索する「Microsoft ランゲージポータル」のようなページが該当します。アドレスは常に「http://www.microsoft.com/Language/ja-jp/Search.aspx」のままですが、ページを開いた時点で検索語句の入力フィールドにカーソルが置かれています。

こうした検索サイトを検索先として追加するときは、Web ページ内の検索語句入力フィールドの上で「Ctrl+Alt+中ボタン」し、ドロップダウンの「〈非 URL 型 Web 検索用コード〉」を DictGrpXX() の中に貼り付けたうえで、URL を指定してください。URL は「%s」のないものをそのまま記述してください。

なお、非 URL 型検索サイトの検索は、処理の都合上、多少ぎくしゃくして時間がかかる形の実装となっております。ご了承ください。

入力フィールドにカーソルが置かれない非 URL 型検索サイトの場合

ページを開いた時点で入力フィールドにカーソルが置かれない非 URL 型サイトの場合も、フィールドの位置を座標で指定することにより一応は検索可能です。「Ctrl+Alt+中ボタン」の「〈非 URL 型 Web 検索用コード〉」から貼り付けたコードで、カッコ内の 5 番目の項目を、末尾にある "x〇〇 y〇〇" という座標に変えてください。

例:

```
WebOpenSend(0, "http://kod.kenkyusha.co.jp/demo/form.jsp", 10, "研究社辞書検索", "", "^a^v{Enter}") ; "x796 y259"
                                                                                                    ↑この青色の部分を、
                                                                                                      行末にある緑色の内容に変更
```

> ※ただし、このように座標を直接指定する形のコードにした場合、ページのサイズや配置が変わったときに予期せぬ動作となる可能性がありますのでご注意ください。

Web ページを開くタブ番号の指定

デフォルトでは、Web 検索の結果は、Windows 上で「既定のブラウザ」として設定しているブラウザの新しいタブで表示されますが、Chrome、Firefox、Internet Explorer の場合は、検索結果を開くタブを指定できます。たとえば、Google の検索結果は常にブラウザの 1 番左のタブで開く、といったように、何をどのタブを開くか固定しておくと便利です。

この機能を使うためには、次の 2 つの両方を行う必要があります。

- 検索に使うブラウザの設定。
  「ブラウザの設定」セクションで説明した「ini_browserwin」という項目で設定してください。
- 検索先を指定する箇所でのタブ番号の指定。
  検索用コードのカッコ内の先頭項目（下記の青色の部分）に、目的のタブ番号を表す数字を指定してください。
  - URL 型検索サイトの場合：　`WebOpen(0, "http://〜")`
  - 非 URL 型検索サイトの場合：　`WebOpenSend(0, "http://〜", ...)`

タブ番号として指定できる数字と動作は次のとおりです。

タブ番号の指定	動作
0	新しいタブで開く
1〜8	左から n 番目のタブで開く（1=一番左のタブ、2=左から 2 番目のタブ、…）
9	一番右のタブで開く
その他	現在のタブで開く

※1〜9 を指定した場合、該当するタブがないときにどうなるかはブラウザ次第です。なるべく、常に存在するタブの番号を指定してください。

ホットキーの設定

入力ウィンドウやポップアップメニューを表示するホットキーを設定するセクションです。

※一発検索のホットキーは、「辞書グループの名前とキーの設定」セクションで定義します。

デフォルト設定

```
^+k::InputBoxActivate()                          ; 入力ウィンドウを表示
^+i::PopupMenuActivate(1)                        ; 選択語句を検索するポップアップメニューをカーソル位置に表示
MButton::PopupMenuActivate(0, , "{MButton}")     ; 選択語句を検索するポップアップメニューをマウス位置に表示
!^MButton::KanzasiInfoGet()                      ; スクリプト修正用の情報を取得して表示
^+n::NextDictActivate(1)                         ; 次の辞書をアクティブ化
^+p::NextDictActivate(-1)                        ; 前の辞書をアクティブ化
```

設定項目

どのホットキーに対して何をするか、それぞれ 1 行ずつで記述した形となっています。

キーを変更するには、上記の青色で示した部分を、目的のホットキーを表す文字列に変更してください。キーの指定方法は AHK の書式どおりで、「ホットキー設定ヘルパー」を使うと簡単です。

※上の設定例で、3 行目の「MButton」のみ、行頭と行中の 2 カ所に出てきます。この機能を別のマウスボタンに変える場合は、同じように、行頭と行中の 2 カ所に記述してください。

補足説明

入力ウィンドウを表示するときに選択語句を取り込むには

入力ウィンドウを表示するときに、アプリケーション上で選択中の語句を取り込んで入力エリアに表示させたい場合は、上記のコードの 1 行目にある「InputBoxActivate()」という部分で、カッコの中に次のように「SelGet()」と記述してください。

```
^+k::InputBoxActivate(SelGet())
                      ↑このように記述
```

特定のウィンドウをアクティブ化するホットキーを追加するには

スタンドアロン版かんざしには、ホットキーを押すと特定のタイトルのウィンドウをアクティブ化する機能がありました。これは、AHK 自体の機能として簡単に実現できます。

次のような形で、「（ホットキー）::WinActivate, （タイトルまたはクラス名）」という形で指定してください。タイトルは一部のみで問題ありません。

例：ホットキー「Ctrl+Shift+W」で「Microsoft Word」をタイトルに含むウィンドウをアクティブ化する場合

```
+^W::WinActivate, Microsoft Word
```

※ホットキー部での検索先記述について

「Ctrl+Alt+中ボタン」で取得した「〈辞書ソフト検索用コード〉」や「〈非 URL 型 Web 検索用コード〉」）は、このホットキー部に貼り付けても動作しませんのでご注意ください。

検索処理のコードは DictGrpXX() に貼り付けてください。

ショートカットキーの設定

入力ウィンドウ上でのみ有効なキーを設定するセクションです。

デフォルト設定

```
#IfWinActive かんざし - ahk_class AutoHotkeyGUI

^2::                                                    ; 全体の " " 囲みのオン/オフの切り替え
    hk_text := InputBoxGetText()
    hk_text := InStr(hk_text, """") ? StrRepl(hk_text, """") : """" . hk_text . """"
    InputBoxSetText(hk_text, 2)
    return

^3::                                                    ; 各単語の " " 囲みのオン/オフの切り替え
    hk_text := InputBoxGetText()
    hk_text := InStr(hk_text, """") ? StrRepl(hk_text, """") : RegExReplace(hk_text, "(*UCP)(¥S+)", """$1""")
    InputBoxSetText(hk_text, 2)
    return

#IfWinActive
```

設定項目

このセクションは、独自のスクリプトによるカスタマイズのために使っていただくことを想定しているため、コピペで簡単に設定できる項目は特にありません。
デフォルトで設定されている「Ctrl+2」「Ctrl+3」というキーを変更したい場合は、上記の青色の部分を修正してください。

6　目的別　設定リファレンス

主な設定項目について、修正箇所を目的別にまとめました。

目的	設定項目
入力ウィンドウのデフォルトの位置とサイズを変更するには	ini_x, ini_y, ini_width を修正
「入力ウィンドウを常に表示」モードをデフォルトでオンにするには	ini_stayopen を修正
入力ウィンドウのフォントの種類とサイズを変更するには	ini_fontname と ini_fontsize を修正
辞書ソフトやブラウザが未起動の場合に自動で起動してから検索させるには	ini_winexe を設定
辞書グループを新規作成・修正・削除するには	ini_grpid でグループ名・ショートカットキー・ホットキーを定義し、DictGrpXX() で検索先を指定
辞書グループの名前を変更するには	ini_grpid でグループ名の定義を修正
入力ウィンドウ上で辞書グループを選択するためのショートカットキーを変更するには	ini_grpid で ( ) 囲みのキー定義を修正
ホットキー一発検索のキーを変更するには	ini_grpid で (( )) 囲みのキー定義を修正
辞書グループにローカル辞書の検索先を追加するには	DictGrpXX() に追加
既製のコードでローカル辞書がうまく検索できない場合は	DictGrpXX() 内の WinClickSend() の記述を修正
URL に検索語句が含まれるタイプの検索サイトを辞書グループに検索先として追加するには	DictGrpXX() に追加
URL に検索語句が含まれないタイプの検索サイトを辞書グループに検索先として追加するには	DictGrpXX() に追加
Web 検索の検索結果を開くタブを指定するには	ini_browserwin でブラウザを指定したうえで、DictGrpXX() 内の WebOpen() または WebOpenSend() にタブ番号を指定
入力ウィンドウやポップアップメニューを表示するホットキーを変更するには	ホットキー部のキー定義を変更
入力ウィンドウを表示するときに選択語句を取り込むには	ホットキー部の InputBoxActivate() を InputBoxActivate(SelGet())に修正

7　AHK スクリプトによるカスタマイズ

ここでは、AHK のプログラミングをご存知の方向けに、ここまで説明した内容よりも細かなカスタマイズを行う方法や注意点について説明します。

カスタマイズの概要

AHK 版かんざしには、検索語句の送信処理や入力ウィンドウの動作を制御するための、さまざまなヘルパー関数が用意されています。Kanzasi.ahk にデフォルトで記述してあるコードや、「Ctrl+Alt+中ボタン」で貼り付けるコードも、ヘルパー関数を使って実装したものです。

AHK のプログラミングによって細かなカスタマイズを加える場合、こうしたヘルパー関数のほか、AHK の組み込み関数やコマンドを使って、独自の処理を Kanzasi.ahk の中に記述していただく形となります。

> こうしたカスタマイズを行う場合も、Kanzasi_lib.ahk の中身には手を加えず、Kanzasi.ahk 側のみの修正で対応していただくことをお勧めします。

Kanzasi.ahk の中でこうしたカスタマイズを行える場所は、主に次の 3 つです。

- ホットキー部
- ショートカットキー部
- DictGrpXX() 内での検索処理

ホットキー部・ショートカットキー部と、DictGrpXX() とでは、使用できるヘルパー関数に違いがあります。

以下、それぞれの部分でどのようなカスタマイズが可能かという例をいくつか紹介したうえで、ヘルパー関数の一覧を関数リファレンスという形で示します。

※変数についての注意事項

カスタマイズの際に使用する変数について、以下の 2 点にご注意ください。

- Kanzasi_lib.ahk で使用している変数と名前が重複する場合、Warning が表示されます。次のような方法により、名前の重複を回避していただくことをお勧めします。
  - ・ ホットキー部やショートカット部で使用する変数は、名前の先頭に何らかの接頭辞を付ける（たとえば先頭を「hk_」とする）
  - ・ DictGrpXX() 内で使用する変数は、そうした接頭辞を特に付けない変数名とする
- Kanzasi_lib.ahk で内部的に使用しているグローバル変数は、本書で説明するもの以外は、Kanzasi.ahk 側で使用することは避けてください。

ホットキー部のカスタマイズの例

一般に、AHK のスクリプトで辞書検索やWeb 検索を行う場合、ホットキーの定義部に検索のコードを直接記述することが多いと思いますが、かんざしの枠組みの中では、辞書検索やWeb 検索のコードはホットキー部には直接記述せず、DictGrpXX() の中に記述するという形で統一していただくことをお勧めします。

代わりに、特定の辞書グループを検索するためのヘルパー関数として、SelSend() 関数や StrSend() 関数というものがあります。ホットキー部で独自のロジックを使って検索処理を記述したい場合は、それらの関数を呼び出す形で実装してください。

以下、ホットキー部のカスタマイズの例をいくつか示します。

例 1: 入力ウィンドウを表示するときに検索先グループを指定

ホットキー「Ctrl+Shift+K」でかんざしの入力ウィンドウを表示するとき、通常は前回検索した検索先グループがデフォルトで選択されますが、必ず「ローカル辞書」(グループ番号 10)を選択した状態で表示する例です。

```
^+k::
    InputBoxSetGrp(10)
    InputBoxActivate()
    return
```

InputBoxSetGrp() は、入力ウィンドウのグループ番号を設定するヘルパー関数です。InputBoxActivate() は、入力ウィンドウを表示するヘルパー関数です。

例 2: 前回の検索語句を " " で囲んで再検索

Google などで語句を検索した後、フレーズ検索として簡単に再検索できるようにする例です。ホットキー「Ctrl+Shift+2」を押すと、直前に検索した語句を " " で囲んで検索し直します。検索先グループは直前の検索と同じです。

```
+^2::StrSend(g_lastsentgrp, """" . g_lastsentstr . """")
```

StrSend() 関数は、指定したグループに指定した語句を送信するヘルパー関数です。g_lastsentgrp と g_lastsentstr は、前回検索したグループ番号と語句を参照するためのグローバル変数です。

例 3：語句を未選択のときには検索しない一発検索

グループ名の定義部分で (( )) 囲みで指定するホットキー一発検索は、文字列を選択していないときには前回の語句を同じとみなして検索を実行する形となっていますが、このときに検索を実行しないようにカスタマイズする例です。ホットキー「Ctrl+Shift+G」で Google 日本（グループ番号 20）を一発検索するときに、語句が未選択の場合は検索せず、代わりに音を鳴らします。

```
+^g::
    if (!SelSend(20)) {
        SoundPlay, *-1
    }
    return
```

SelSend() 関数は、選択中の語句を指定グループに送信するヘルパー関数です。この関数は、語句を未選択のときには送信を実行せずに戻り値 0 を返すことから、その場合に音を鳴らしています。

なお、こうした一発検索のコードをホットキー部に記述する場合は、グループ名の定義部分で (( )) 囲みで指定していた同じキーは削除して、重複がないようにしてください。

例 4：選択語句に応じて一発検索の送信先を振り分け

ホットキー「Ctrl+Shift+G」の Google 一発検索で、選択中の語句に応じて検索先を自動で振り分ける例です。選択語句に漢字・ひらがな・カタカナが含まれる場合は Google 日本（グループ番号 20）を、それ以外の場合は Google US（グループ番号 30）を一発検索します。語句を未選択なら何もしません。

```
^+g::
    if (hk_text := SelGet()) {
        hk_grp := RegExMatch(hk_text, "[¥p{Han}¥p{Hiragana}¥p{Katakana}]") ? 20 : 30
        StrSend(hk_grp, hk_text)
    }
    return
```

SelGet() 関数は、Windows アプリケーション上で選択中の語句を取得するヘルパー関数です。RegExMatch() 関数は AHK の組み込み関数で、正規表現によるマッチングを行うことができます。

上の例で、変数名の先頭に「hk_」と付けているのは、Kanzasi_lib.ahk で内部的に使用している変数と名前がかぶるのを回避するための策です。

例 5：選択語句に簡単な語尾補正を施してからポップアップメニューで検索

ホットキー「Ctrl+Shift+I」でポップアップメニュー検索を行うときに、選択中の語句をそのまま検索するのではなく、語尾補正を行ってから検索させる例です。単語の末尾にある「ed」「ing」を削除してからポップアップメニューを表示します。

※この補正だけでは単純すぎて使い物にならないと思いますが、このような感じで文字列を置換してからヘルパー関数に引き渡せばよい、という考え方を示した例とご理解ください。

```
^+i::
    if (hk_text := SelGet()) {
        hk_text := RegExReplace(hk_text, "(ed|ing)¥b")
        PopupMenuActivate(1, hk_text)
    }
    return
```

PopupMenuActivate() 関数は、ポップアップメニューを表示するヘルパー関数です。RegExReplace() 関数は AHK の組み込み関数で、正規表現による置換を実行できます。

ショートカットキー部のカスタマイズの例

本書およびスクリプト内では、かんざしの入力ウィンドウ上でのみ有効なキーを便宜的に「ショートカットキー」と表現していますが、AHK プログラムの処理としては、通常のホットキーと特に違いはありません。スクリプトの中で、次のようなブロックで囲むことにより、入力ウィンドウ上でのみ機能するようにしているだけです。

```
#IfWinActive かんざし - ahk_class AutoHotkeyGUI
...
#IfWinActive
```

したがって、ショートカットキー部でも、ホットキー部と同じヘルパー関数を使って、同じように処理を記述できます。

※以下に示すサンプルコードはいずれも、上記の #IfWinActive で挟まれた中に記述してください。（そうしないと、Windows 上のあらゆる部分で有効なホットキーになってしまいます）。
デフォルトで記述してある「Ctrl+2」「Ctrl+3」のコードに並べる形で配置すれば OK です。

例 1：Esc キーの機能の変更

入力ウィンドウで Esc キーを押したときに、ウィンドウを閉じるのではなく、入力内容をクリアするようにする例です。

```
#IfWinActive かんざし - ahk_class AutoHotkeyGUI
Esc::InputBoxSetText("")
#IfWinActive
```

InputBoxSetText() 関数は、指定した文字列を入力エリアに設定するヘルパー関数です。

例 2：ファンクションキーで Google を一発検索

デフォルトでは、入力ウィンドウ上で F1～F5 のファンクションキーを押すと、グループの選択が切り替わりますが、ファンクションキーを押した時点で検索自体を実行させる例です。
F6 キーを押したときに Google 日本を検索します（グループ 20 と想定）。

```
#IfWinActive かんざし - ahk_class AutoHotkeyGUI
F6::
    InputBoxHide()
    StrSend(20, InputBoxGetText())
    return
#IfWinActive
```

InputBoxHide() 関数は、入力ウィンドウを閉じるヘルパー関数です。InputBoxGetText() 関数は、入力ウィンドウの入力内容を取得するヘルパー関数です。（入力ウィンドウが非表示の状態でも取得できます）。

例 3：簡易ランチャー

かんざしの入力ウィンドウを簡易ランチャーとして使えるようにする例です。
入力内容の先頭が「//」の場合はランチャーのコマンド入力とみなし、次のように処理を実行します。

- //calc　Windows に付属の電卓を起動
- //mydoc エクスプローラーで「マイドキュメント」フォルダを開く
- //edit　メモ帳で Kanzasi.ahk を開く
- //hkhelper　かんざしに同梱の「ホットキー設定ヘルパー.ahk」を起動

先頭が「//」でなければ、通常どおりの検索となります。

```
#IfWinActive かんざし - ahk_class AutoHotkeyGUI
Enter::
    hk_cmdlist := { "calc" : "calc.exe"
        , "mydoc" : A_MyDocuments
        , "edit" : "notepad.exe " . A_ScriptDir . "¥Kanzasi.ahk"
        , "hkhelper" : A_ScriptDir . "¥ホットキー設定ヘルパー.ahk" }

    if (RegExMatch(InputBoxGetText(), "^//(.+)$", hk_m)) {
        InputBoxHide()
        InputBoxSetText("")
        if (hk_cmdlist.HasKey(hk_m1)) {
            Run, % hk_cmdlist[hk_m1]
        }
    } else {
        Send, {Enter}
    }
    return
#IfWinActive
```

入力ウィンドウ上での Enter に反応し、入力内容に応じた処理を実行しています。コマンドと実行内容は、コードの冒頭で定義している連想配列 hk_cmdlist で編集可能です。

なお、上記のように Enter キーで起動するコードは、IME の日本語入力を確定する Enter にも反応してしまいます。コードを書く際に注意しないと、予期せぬ動作となることがあります。

> たとえば、上記のコードで InputBoxSetText("") という行を削除すると、うまく動作しません。（「//～」のコマンドを実行した後で、日本語入力による検索を実行しようとした時に、IME の入力を確定したタイミングで、前回のコマンドが再び実行されてしまいます）。

また、本題とは関係ない話ですが、計算を実行したいときは、電卓を起動するより、かんざしの入力ウィンドウに計算式を直接入力して、Google 検索で答えを表示した方が簡単かもしれません。

DictGrpXX() のカスタマイズの例

「Ctrl+Alt+中ボタン」で DictGrpXX() 関数に貼り付けたコードがうまく動作しない場合や、もっと細かな制御を加えたい場合は、DictGrpXX() 関数の中に独自の検索処理のコードを記述してください。
DictGrpXX() 関数が呼び出された時点で、検索語句が Clipboard 変数に入っていますので、その値を使用して検索を行う形でコードを記述すれば OK です。

※DictGrpXX() 関数の中で Clipboard 変数の中身を変更しても問題ありませんが、変更後に WinClickSend() 関数や WebOpen() 関数を呼び出した場合、変更後の内容で検索が行われます。

DictGrpXX() 内で独自の検索処理を記述するには、主に次の 2 つの方法があります。

- **WinClickSend() 関数を使う**:ウィンドウをアクティブ化し、何らかのコントロールをクリックし、何らかのキーを送信する、という、典型的な流れの処理であれば、WinClickSend() というヘルパー関数を使うと簡単です。
- **検索処理を直接記述する**:ヘルパー関数を使わずに、AHK の組み込み関数やコマンドのみで処理を実装してもかまいません。(たとえば、EXE のコマンドライン引数による検索が可能なソフトは、AHK の Run コマンドで検索する形でコードを記述して問題ありません)。

※WinClickSend() 関数を使った場合、EXE を自動起動する機能が有効になります(つまり、対象のウィンドウの EXE ファイル名が ini_winexe 変数で設定してあれば、未起動のときは自動で起動します)。
また、WinClickSend() を使わない場合も、AHK の WinActivate コマンドでウィンドウをアクティブ化する部分を、かんざしの WinActivateOrExecute() というヘルパー関数に置き換えると、EXE の自動起動が有効になります。

※ヘルパーを関数を使わずに検索処理を直接記述した場合、その検索先のウィンドウは、「次の辞書/前の辞書をアクティブ化」の機能でアクティブ化の対象になりません。

以下にいくつか例を示します。いずれの例も、辞書グループの中に単独の検索先のみを記述した形となっていますが、「Ctrl+Alt+中ボタン」などのコードと組み合わせて、複数の検索先の中の 1 つとして使ってもかまいません。

例 1:Excel シートの検索

Excel のオートフィルタ機能をかんざしから操作して、Excel ファイルの用語集の中から、特定の単語を含む行のみを抽出する例です。「Glossary.xlsx」という用語集で、A 列に英語、B 列に日本語が並んでおり、オートフィルタの[▼]ボタンが A1 セルにあるとします。

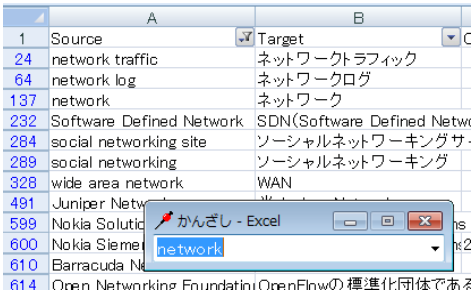
```
DictGrp60()
{
    SetKeyDelay, 100
    WinClickSend("Glossary.xlsx ahk_class XLMAIN", "", "^gA1{Enter}!{Down}fa^v{Enter}")
    SetKeyDelay, 0
}
```

※このコードは Excel 2007 の場合のものです。バージョンによっては修正が必要かもしれません。また、作者の環境では、Excel にキーを送るときには、上のコードのように SetKeyDelay コマンドでキー間隔を少し空けないと、正しく送信できません。こちらも環境によっては不要かもしれません。

このコードでは、WinClickSend() 関数を使って、用語集の Excel ファイルのオートフィルタの絞り込み条件を操作しています。

具体的には、次のような流れで処理を行っています。
用語集の Excel ウィンドウをアクティブ化 → 「Ctrl+G」で[ジャンプ]ダイアログを表示 → 「A1」と入力して[Enter]で A1 セルにジャンプ → 「Alt+↓」でフィルタのメニューを表示 → 「F」「A」で[オートフィルタ オプション]ダイアログを表示 → オートフィルタの絞り込みオプションとしてクリップボードの内容を貼り付けて[Enter]

これを実行すると、次の画像のように、検索した語句を含むセルのみを抽出できます。



例 2：PASORAMA の一括検索 / 成句検索の使い分け

セイコーインスツルの電子辞書に付属の検索ソフト「PASORAMA」で、一括検索と成句検索を使い分ける例です。検索語句を "" で囲んで指定した場合は成句検索、囲まない場合は一括検索を実行します。

```
DictGrp60()
{
    if (RegExMatch(Clipboard, "^""(.+)""$", m)) {
        btn := "Button21"
        Clipboard := m1
    } else {
        btn := "Button19"
    }

    ControlGetPos, x, y, w, h, %btn%, PASORAMA
    x := x + w/2
    y := y + h/2

    if (WinClickSend("PASORAMA", "Edit1", "{Home}+{End}^v")) {
        Click, %x%, %y%
    }
}
```

※このコードは、作者の環境（Win7 64bit）で、DF-X10001 に付属の PASORAMA を使用した場合のものです。環境やバージョンによっては修正が必要かもしれません。

検索語句が " " 囲みかどうかに応じて、一括検索ボタンと成句検索ボタンのどちらかのクリック位置を算出したうえで、PASORAMA の入力フィールドに検索語句を送信し、そのボタンをクリックしています。

作者の環境では、PASORAMA のボタンは ControlClick では動作しないため、ボタンの中央の座標を取得して Click する形にしています。ボタンの ClassNN は、「Ctrl+Alt+中ボタン」の「〈マウス位置クラス〉」で確認できます。

例 3：Firefox の検索窓の操作

作者の環境において、Firefox の検索窓で 3 番目に登録してある検索サイトで語句を検索する例です。

```
DictGrp60()
{
    WinClickSend("ahk_class MozillaWindowClass", "x1160 y77", "!{Down}", 200, "{Dow
n 3}{Enter}", 200, "^a^v{Enter}") ; ←実際には 1 行で記述
}
```

検索窓に対し、複数回に分けてキーを送り込んでいます。

具体的には、次のような流れで処理を行っています。
Firefox のウィンドウをアクティブ化 → ウィンドウの左上から「x1160 y77」の位置のコントロール（検索窓）をクリック → 「Alt+↓」→ 200 ミリ秒休止 →「↓ ×3 回、Enter」→ 200 ミリ秒休止 →「Ctrl+A、Ctrl+V、Enter」

これはあくまで、作者の環境でのコードです。検索窓の位置は場合により異なります。「Ctrl+Alt+中ボタン」のドロップダウンの「〈マウス位置相対座標〉」で確認できます。

例 1 と例 2 の WinClickSend() 関数では、キー送信を 1 回でまとめて行っていましたが、この例のように、途中に Sleep を入れながら複数回に分けてキーを送り込む処理にも対応できます。

なお、このコードのように、座標を直接指定するコードは、ウィンドウのサイズや配置が変わった場合に予期せぬ動作となる可能性がありますのでご注意ください。

例 4：独自規格の辞書ソフトの検索

作者の環境において、Oxford Learner's Thesaurus に付属の検索ソフトで語句を検索するためのコードです。

```
DictGrp60()
{
    if WinActivateOrExecute("ahk_class olt1UIWindowClass") {
        Click, 92 ,109
        Send, {Home}+{End}^v{Enter}
        Sleep, 200
    }
}
```

作者の環境では、この検索ソフトは WinClickSend() 関数で検索できません。（WinClickSend() 関数は、内部で ControlClick コマンドを使っているのですが、それが効かないようです）。

そこで、アクティブ化 → 入力エリアの座標位置をクリック →「全選択、Ctrl+V、Enter」、という処理を直接記述しています。

先ほども書きましたが、こうして座標を直接指定するコードは、ウィンドウのサイズや配置が変わると誤動作する可能性がありますのでご注意ください。

※ちなみに、この辞書は Logophile でも検索できますので、そちらを使った方が手っ取り早いと思います。

ヘルパー関数リファレンス

カスタマイズで使用できるヘルパー関数は、ホットキー部・ショートカットキー部と、DictGrpXX() 内とで異なります。以下、それぞれの一覧を、関数リファレンスという形で示します。

ホットキー部・ショートカットキー部で使用できるヘルパー関数

アクティブなアプリケーションで選択中の文字列を取得

関数名	SelGet()
パラメータ	なし
戻り値	選択中の文字列。取得できなかった場合は ""
説明	この関数は、文字列を選択中かどうかという判別にも使えます。戻り値が "" 以外なら選択中と判断できます。ただし、アプリケーションによっては、正しく取得できない場合があります。

選択中の文字列を辞書グループの検索先に送信

関数名	SelSend(grp)	
パラメータ	grp	検索先の辞書グループ番号
戻り値	送信した場合は 1、送信しなかった場合は 0	
説明	通常、ホットキー一発検索の定義は、グループ名の定義の所で (( )) 囲みでキーを指定すれば済みますが、送信処理をコードで制御したい場合は、この関数を使用してください。 なお、この関数は、文字列を選択していない時には送信を実行しません。	

指定した文字列を辞書グループの検索先に送信

関数名	StrSend(grp, text)	
パラメータ	grp	検索先の辞書グループ番号
	text	送信する文字列
戻り値	送信した場合は 1、送信しなかった場合は 0	
説明	text が空文字列の場合は送信を実行しません。	

かんざしの入力ウィンドウの現在の入力内容を取得

関数名	InputBoxGetText()
パラメータ	なし
戻り値	現在の入力内容
説明	ウィンドウが表示されていない状態でも使用できます。(その場合、前回ウィンドウが閉じた時点での入力内容が取得されます)。

かんざしの入力ウィンドウの入力エリアに文字列を設定

関数名	InputBoxSetText(text, focusmode)	
パラメータ	text	設定する文字列
	focusmode （省略可）	入力エリアのフォーカス状態をどうするか（0=全選択、1=先頭にカーソル、2=末尾にカーソル）（省略時は 0=全選択）
戻り値	なし	
説明	ウィンドウが表示されていない状態でも使用できます。（その場合、focusmode の指定は特に意味を持ちません）。	

かんざしの入力ウィンドウの履歴部分に文字列を追加

関数名	InputBoxAddListItem(text, selectthis)	
パラメータ	text	設定する文字列
	selectthis （省略可）	追加する項目を選択して入力エリアに表示するかどうか（true / false） 省略時は false。この場合、入力エリアの内容は元のまま
戻り値	なし	
説明	ウィンドウが表示されていない状態でも使用できます。	

かんざしの入力ウィンドウの送信先グループを設定

関数名	InputBoxSetGrp(grp)	
パラメータ	grp	設定するグループ番号
戻り値	なし	
説明	存在しないグループ番号を指定した場合は無視されます。 ウィンドウが表示されていない状態でも使用できます。	

入力ウィンドウを表示

関数名	InputBoxActivate(text, focusmode)	
パラメータ	text （省略可）	入力エリアに設定する文字列（省略時は前回の状態のまま。つまり、前回検索した語句 または 入力途中の語句）
	focusmode （省略可）	入力エリアのカーソルの状態（0=全選択、1=先頭にカーソル、2=文字列にカーソル）（省略時は 0=全選択）
戻り値	なし	

入力ウィンドウを消去

関数名	InputBoxHide(force)	
パラメータ	force （省略可）	「入力ウィンドウを常に表示」モードがオンのときでもウィンドウを閉じるかどうか（true / false）（省略時は false）
戻り値	なし	
説明	force が true の場合、「入力ウィンドウを常に表示」モードがオンでも、オフにしたうえでウィンドウを閉じます。false の場合、「常に表示」がオンのときは、この関数は何もしません。	

ポップアップメニューを表示

名称	PopupMenuActivate(position, text, unselsend)	
パラメータ	position （省略可）	メニューの表示位置（0=マウスの位置、1=アクティブウィンドウのカーソル付近）（省略時は 0=マウスの位置）
	text （省略可）	送信する文字列 （省略時は、アプリケーション上で選択中の語句）
	unselsend （省略可）	text が "" で、かつ語句が未選択だった場合に、代わりに Send で送るキー。（※このパラメータは text が "" の場合のみ有効）
戻り値	メニューを表示した場合は 1、表示しなかった場合は 0	
説明	語句を検索するためのポップアップメニューを表示します。対象は、text に文字列を指定した場合はその文字列、text を指定しなかった場合はアプリケーション上で選択中の語句です。unselsend は、語句が未選択の状態だった場合に、ホットキーのデフォルトの動作を実行させるためのパラメータです。マウスボタンをトリガーとしてこの関数を呼び出す場合に使用することを想定しています。特に指定しなくても問題ありません。	

スクリプトをカスタマイズするための情報を取得して表示

名称	KanzasiInfoGet()
パラメータ	なし
戻り値	なし

次/前の辞書をアクティブ化

名称	NextDictActivate(mode)	
パラメータ	mode	アクティブ化するウィンドウ（0=前回実行した検索の先頭辞書をアクティブ化、正の数=次の辞書をアクティブ化、負の数=前の辞書をアクティブ化）
説明	スタンドアロン版かんざしにあった「次の辞書/前の辞書をアクティブ化」のホットキーと同様の機能を実現するためのものです。起動以降に検索語句を送信した辞書ソフトや Web ブラウザのウィンドウを、最近検索したものから順にアクティブ化します。	

ホットキー部・ショートカットキー部で参照できるグローバル変数

ホットキー部・ショートカットキー部では、次に示すグローバル変数を参照できます。
ただし、読み取りのみ可とします。値の設定は行わないでください。

前回実行した検索の送信先辞書グループ

名称	g_lastsentgrp
戻り値	前回の検索の辞書グループ番号を表す数値

前回実行した検索の検索語句

名称	g_lastsentstr
戻り値	前回の検索語句を表す文字列

入力ウィンドウで選択している送信先辞書グループ

名称	g_inputboxgrp
戻り値	入力ウィンドウで現在選択中の辞書グループ番号を表す数値
説明	現在選択中の辞書グループを取得するための変数です。辞書グループを変更したい場合は、この変数ではなく、InputBoxSetGrp() 関数を使用してください。

DictGrpXX() 内で使用できるヘルパー関数

ウィンドウのアクティブ化（または起動）、ControlClick、Send、Sleep を順に実行する関数

関数名	WinClickSend(wintitle, clicktarget, key_or_sleep*)	
パラメータ	wintitle	アクティブ化するウィンドウのタイトル and / or クラス（AHK の WinExist()関数の第 1 パラメータと同じ形式）
	clicktarget	ControlClick でクリックする対象のコントロール（AHK の ControlClick コマンドの Control-or-Pos パラメータと同じ形式）。 空文字列（""）の場合はクリックは行わない。
	key_or_sleep*	Send で送るキーを表す文字列、または Sleep で休止する時間（ミリ秒）を表す数値をカンマ区切りで並べて指定する。 個数が可変のパラメータなので、引数をいくつでも指定可
戻り値	キーを送信した場合は対象のウィンドウの HWND、それ以外は 0	
説明	アクティブ化の対象が起動していない場合、ini_winexe 変数で該当の EXE ファイルが設定してあれば、自動で起動します。 また、一般に、辞書検索や語句送信の後は多少 Sleep が必要ですが、最後の引数が数値でなかった場合には、ある程度の Sleep を自動で入れるようになっています。	

指定のウィンドウが存在すればアクティブ化し、存在しなければ起動する関数

関数名	WinActivateOrExecute(wintitle, execwaitsec)	
パラメータ	wintitle	アクティブ化するウィンドウのタイトル and / or クラス（AHK の WinExist()関数の第 1 パラメータと同じ形式）
	execwaitsec （省略可）	EXE が起動しない状態が何秒続いたらエラーとみなすか（秒で指定） （省略時は 5 秒）
戻り値	ウィンドウが存在するか、または起動できた場合は、その HWND。それ以外は 0	
説明	アクティブ化の対象が起動していない場合、ini_winexe 変数で該当の EXE が設定してあれば、自動で起動します。	

※以上２つの関数で、EXE の自動起動を機能させるためには、ini_winexe 変数で定義しているウィンドウクラス / タイトルが、wintitle の指定内容と完全に一致する必要があります。

たとえば、ini_winexe で "ahk_class XLMAIN" の EXE を指定してあっても、上記の関数の wintitle パラメータの指定が "Microsoft Excel ahk_class XLMAIN" だった場合、EXE は起動しません。

クリップボードの内容（検索文字列または URL）をブラウザで開く関数

関数名	WebOpen(tabnum, queryurl)	
パラメータ	tabnum （省略可）	ブラウザ内のどのタブで開くか（0=新しいタブ、1～8=その番号のタブ、9=一番右のタブ、その他=現在のタブ）（省略時は 0）
	queryurl （省略可）	検索先の URL（検索語句部分を%s で指定したもの）。（省略時は ""） 空文字列の場合は、ブラウザの検索窓に文字列を送り込んで検索する
戻り値	検索できた場合はそのハンドル（HWND）。それ以外は 0	
説明	この関数は、検索ではなく普通の Web ページを特定のタブで開く目的でも使用できます。その場合は、「%s」を含まない URL を指定してください。 ini_browserwin 変数が未設定の場合、tabnum は常に 0 とみなし、また queryurl が "" はエラーとみなします。	

クリップボードに入っている URL の検索ページをブラウザで開き、そのカーソル位置に語句を送り込んで検索する関数

関数名	WebOpenSend(tabnum, queryurl, waitsec, wintitle, clicktarget, key_or_sleep*)	
パラメータ	tabnum	ブラウザ内のどのタブで開くか（0=新しいタブ、1～8=その番号のタブ、9=一番右のタブ、その他=現在のタブ）
	queryurl	検索先の URL。空文字列の場合はエラーとみなす
	waitsec	その URL のページが開き終わるまで何秒待つか
	wintitle	アクティブ化するウィンドウのタイトル and / or クラス（AHK の WinExist()関数の第 1 パラメータと同じ形式）
	clicktarget	ControlClick でクリックする対象のコントロール（AHK の ControlClick コマンドの Control-or-Pos パラメータと同じ形式）。 空文字列（""）の場合はクリックは行わない。
	key_or_sleep*	Send で送るキーを表す文字列、または Sleep で休止する時間（ミリ秒）を表す数値をカンマ区切りで並べて指定する。 個数が可変のパラメータなので、引数をいくつでも指定可
戻り値	検索できた場合はそのハンドル（HWND）。それ以外は 0	
説明	ini_browserwin 変数が未設定の場合、tabnum は常に 0 とみなします。	

どこでも使用できるヘルパー関数

次に示す関数は、DictGrpXX() 内、ホットキー部、ショートカットキー部のどこでも使用できます。

文字列を置換（StringReplace コマンドのラッパー）

関数名	StrRepl(InputVar, SearchText, ReplaceText)	
パラメータ	InputVar	置換対象が含まれる文字列（haystack）
	SearchText	置換対象の文字列（needle）
	ReplaceText （省略可）	何に置き換えるか。省略時は "" とみなす
戻り値	マッチする箇所をすべて置換した結果	
説明	AHK の StringReplace コマンドを関数形式で呼び出すための単なるラッパーです。 各パラメータは、StringReplace コマンドの同名の項目とまったく同じ意味です。 StringReplace コマンドで ReplaceAll パラメータを指定した場合と同じく、マッチする箇所をすべて置換した結果を返します。	