AutoHotkey Tricks You Ought To Do With Windows

Jack Dunning

ComputorEdge E-Books

Table of Contents

Fair Use Copyright

"You should be able to read your e-books when you want and where you want."

We believe that you should be able to peruse the e-books you buy on any reading device you own. Therefore, we don't encrypt our books or implement Digital Rights Management. We depend upon your sense of fairness to determine how, when and where you will read our e-books.

As of the Third Edition, this particular book, *AutoHotkey Tricks You Ought To Do With Windows*, is a free book which you are free to distribute to anyone. It is intended to act as a motivator for more people to start using AutoHotkey and as a reference for all the other books—including the Table of Contents and Index for each. Please share.

Fair Use Copyright licenses the buyer to load this book on any device owned by the buyer. However, if you give it to someone else, you are in technical violation of the license for this e-book. We don't know how we would know, so maybe you could go ahead and share this with a friend. With any luck at all, they will appreciate our years of work and purchase their own copy of the book, which is probably exactly like the one they already have (or just send us money for their current copy). Since this is an e-book there is a chance that a newer edition with new material will be available. If you plan to give the e-book as a gift, purchase the download at our Web site, www.ComputorEdgeBooks.com.

If you've found this e-book useful and didn't buy it, you can purchase it at the above address. We know that many of these AutoHotkey books have been pirated and are easier found on the Internet. If you're not afraid to download from one of these dodgy sites and obtain one of these pirated copies, but did not get any value from it, then you owe us nothing. (We would have refunded the price if you had purchased it and found it useless.) But, if you get value out of any of our books, then your sense of fairplay tells you that you should compensate us for the extensive work we put into them. The easiest way to do that is by actually purchasing the book. Plus, it is quite possible that there will be an updated edition available.

If you're wondering about the price of the book, you can check with the normal distribution networks or at www.ComputorEdgeBooks.com. If you're paying us directly by mailing a check, email or call for the address.

Introduction to AutoHotkey Tricks You Ought to Do

"This free book offers some very useful (mostly beginning level) ideas for AutoHotkey while providing complete indexes to Jack's other books."

> *Here are a number of simple AutoHotkey techniques that anyone can use to improve their Windows experience. Plus, if you're looking for more AutoHotkey information, the Table of Contents and Indexes for Jack's other AutoHotkey books are included in this free book.*

After a number of years of using AutoHotkey (and writing five other AutoHotkey books), I'm still convinced that it is absolutely the best free Windows utility software ever! One of my goals is to spread the word about AutoHotkey and show how easy it is for anyone to get started. That's why I regularly write a blog about AutoHotkey, post AutoHotkey help pages for beginners, create sample AutoHotkey scripts available for download, publish AutoHotkey e-books, and now I'm offering the third edition of this AutoHotkey overview e-book free. There is a lot of good information in this book for people trying to decide whether or not AutoHotkey is right for them. Since AutoHotkey's roots are in a program called AutoIt, I've added a chapter for those trying to decided which program to choose: AutoHotkey or AutoIt. (In short, other than language syntax, the major difference between AutoHotkey and AutoIt is that there is much better support for hotkeys which add action shortcuts and hotstrings—primarily text replacement—in AutoHotkey. This is what most beginners and casual users want.) But remember, regardless of which route you take, if you don't do anything else, you ought to do at least one or two of the AutoHotkey tricks offered here on your Windows computer.

AutoHotkey Tricks You Ought To Do is broken up into two parts: the AutoHotkey things you ought to do on your Windows computer and a convenient reference section which includes the Table of Contents and Indexes from my four other AutoHotkey books. If you own any of my other AutoHotkey books, then this book can help you find which book has the answer to your questions by searching the Table of Contents and indexes included here for each book. That should be easier than searching all five books separately.

Note: You might notice that this book is no longer available on Amazon. While I would like to keep it there, they won't let me give it away. It just wasn't getting into enough people's hands. Therefore, as of this edition, the book will always be free and easily available as a download at the ComputorEdge site.

The Things You Ought To Do

Most of the things that you ought to do with AutoHotkey are simple (often one-line) scripts which will immediately give you results. A few of them get a little more complicated, but you shouldn't need any other books to implement most of these Windows tricks. All that's required is the free AutoHotkey program installed on one of your Windows computers.

Installing AutoHotkey

Installing AutoHotkey is as simple as downloading the program and running it. Probably the best way to download the current version (changed from AutoHotkey_L to just plain AutoHotkey) is go directly to the AutoHotkey Web site and click the Download button. From time to time you will see references to AutoHotkey_L which took over as primarily from the original AutoHotkey package. The current download has now taken on the original AutoHotkey name rather than AutoHotkey_L.

Version 2.0 Note: AutoHotkey 2.0-a is a release of AutoHotkey in the alpha stage of development. The goal is to clean up many of the redundant and confusing aspects of the current version. However, there may be a considerable loss of backward compatibility when using 2.0. (Many scripts found on the AutoHotkey sites will not run properly.)

For now, it is recommended that new users stick with the current 1.1 release.

Once you have installed AutoHotkey, then there are plenty of free online sources of AutoHotkey information. I put together my AutoHotkey books for those people (beginners and intermediate users) who want a little more explanation and to act as references—although by no means do my books cover all of the power of AutoHotkey.

Writing Your First AutoHotkey Script

Once the main AutoHotkey program is installed, if you can open the basic text editor Windows Notepad, then you can write a script. You only need to copy one of the many scripts available on the Web, paste it into a Notepad window, and save it as an AHK file (*MyScript.ahk*). There are examples at the "Installing AutoHotkey and Writing Your First Script" Web page. Right-click on the filename and select Run Script to load the file.

If you want a portable script, then you can right-click on the same filename and select Compile Script to create a separate EXE file (*MyScript.exe*) which you can run on any Windows computer with a simple double-click—with or without the main AutoHotkey program installed. That means you don't need to install AutoHotkey on every Windows computer to use AutoHotkey scripts. Just compile it into an EXE file.

If you later need to change an AHK file, right-click on the filename and select Edit Script. The script will open with your default text editor—often Notepad.

The Power of AutoHotkey

Most of the examples included in this book are either hotstring replacements for adding text or hotkey combinations to execute tasks. These features are the basis for AutoHotkey and if this is all you do, it will serve you well. However, AutoHotkey can do so much more by building useful apps for your Windows computers. (Just like AutoIt, AutoHotkey has tremendous power for automating any Windows program.)

There is a set of Graphical User Interface (GUI) pop-up windows built into the software which can be used to create gadgets and widgets for a variety of tasks. For a taste of these capabilities, see "AutoHotkey Applications Contents and Index" in this book or the *ComputorEdge* article "A Beginner's Review of AutoHotkey Pop-up Windows" which includes images of most of the available GUI pop-ups.

The power of AutoHotkey goes far beyond the topics covered in my AutoHotkey books which are aimed at beginning and intermediate script writers. I have seen many advanced commercial quality apps posted (and free to use) on the AutoHotkey forums. These involve AutoHotkey commands and routines which access the inner workings and hidden mechanisms of Windows. I don't pretend to have the level of knowledge or skill to produce these apps, but I can certainly help newcomers. Plus, many of these same snippets of code can be used in other scripts without a full knowledge of how they work.

Reference Look Up For AutoHotkey Books

If you're wondering if any of my books might be right for you, review the Table of Contents and Index for each of the books in the back of this e-book. This should give you enough information to determine whether one of the books will be helpful.

If you decide to buy any of the books, I would prefer that you purchase them from the ComputorEdge E-Books Web site. While they are all available at Amazon.com, independent authors are charged extra based upon the size of the e-book. Since all of my AutoHotkey books have numerous graphic images, the charge per book to *ComputorEdge* is significant. (No other reseller of e-books charges authors this type of download fee, but Amazon is so big there isn't much that anyone can do about it. The authors hit the hardest are those who include photos and illustrations.) Of

course, you should purchase wherever you feel most comfortable.

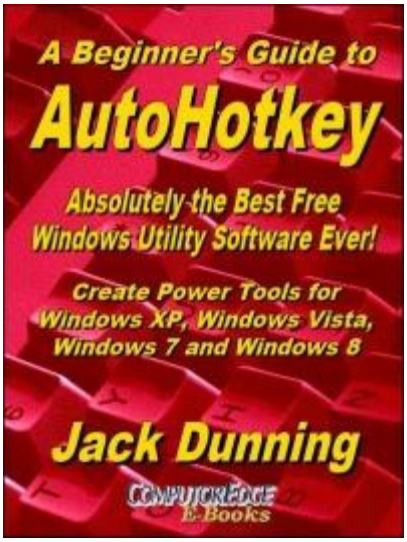
ComputorEdge E-Books also provides two downloads for each book purchase. This is done just in case there is a problem with the first download, but it also facilitates downloading the book directly to more than one device. (None of the books are copy-protected so it is relatively easy to copy the file to other computers and devices.) If you buy at ComputorEdge E-Books and run out of downloads, send us an e-mail and we will give you more downloads.

While we prefer that you not distribute copies of the book to your friends and colleagues without paying for additional books, we know that some people will do that anyway. If you're the recipient of any such copies and you find them useful to you, then we would appreciate it if you would buy your own copy at ComputorEdge E-Books. If you don't find the books of value, then there is no need to pay for them. We don't want you to waste your money and would quickly refund any money paid if you purchased the books directly from us.

I have found a few places on the Web where the books have been pirated and made available as free downloads. While this is illegal, there is little that the average author can do about it. Plus, my philosophy is that people who download from these sites would never have paid for the book in the first place. If push comes to shove, I prefer that my books be easy to use (no copy protection) rather than limit them to only people who can make a decision sight unseen.

My goal is to help publicize AutoHotkey to all Windows users (whether they use my books or not). Since it's open source and free to the world, there is no marketing arm to help promote AutoHotkey's value. (AutoHotkey is recognized for its usefulness and promoted by many of the top technical writers on the Web.) It's growing in popularity and many people use AutoHotkey extensively in their professions. In a world that is still dominated by Windows computers, AutoHotkey is a must have tool.

* * *



The third edition with more tips, updated chapters, and an index to the AutoHotkey commands found in the book is available in e-book format from Amazon (and other formats—EPUB and PDF— at the ComputorEdgeBooks Web site linked below). Jack's *A Beginner's Guide to AutoHotkey, Absolutely the Best Free Windows Utility Software Ever!: Create Power Tools for Windows XP, Windows Vista, Windows 7 and Windows 8* (preferred, EPUB format for iPad, Android, and computers; MOBI for Amazon Kindle; and PDF for printing) offers a gentle approach to learning AutoHotkey. (Also available from Amazon for the Kindle and Kindle software on other devices.)

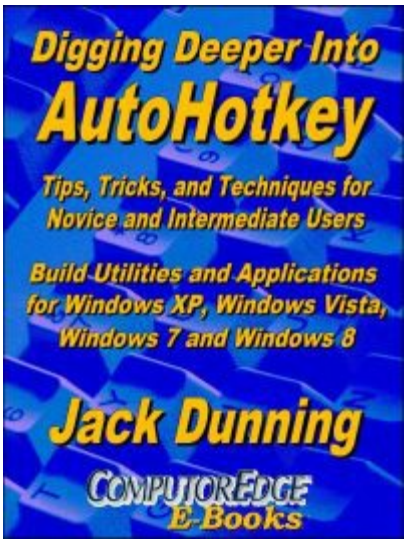
Building Power Tools for Windows XP, Windows Vista, Windows 7 and Windows 8, AutoHotkey is the most powerful, flexible, *free* Windows utility software available. Anyone can instantly add more of the functions that they want in all of their Windows programs, whether installed on their computer or while working on the Web. AutoHotkey has a universality not found in any other Windows utility—free or paid.

Based upon the series of articles in *ComputorEdge*, Jack takes you through his learning experience as he explores writing simple AutoHotkey scripts for adding repetitive text in any program or on the Web, running programs with special hotkeys or gadgets, manipulating the size and screen location of windows, making any window always-on-top, copying and moving files, and much more. Each chapter builds on the previous chapters.

For an EPUB (iPad, NOOK, etc.) version of A Beginner's Guide to AutoHotkey click here!

For a PDF version for printing on letter size paper for inclusion in a standard notebook of A Beginner's Guide to AutoHotkey click here!

* * *

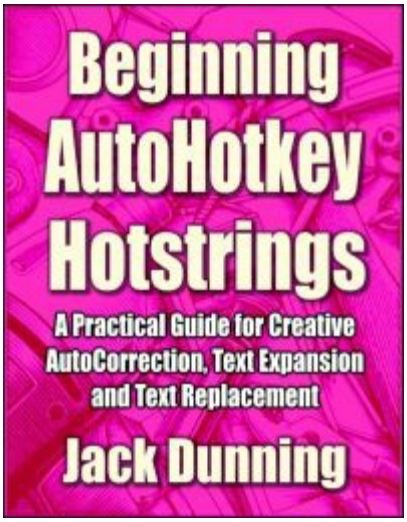


Jack's second AutoHotkey book, *Digging Deeper Into AutoHotkey* (preferred, EPUB format for iPad, Android, and computers; MOBI for Amazon Kindle; and PDF for printing) is comprised of updated, reorganized and indexed columns from *ComputorEdge* is now available. Since the columns were not all written in a linear fashion, the book has been reorganized and broken up into parts by topic. The book is not for the complete beginner since it builds on the information in *A Beginner's Guide to AutoHotkey*. However, if a person is reasonably computer literate, they could go directly to this book for ideas and techniques without the first book. (Also available from Amazon for the Kindle and Kindle software on other devices.)

For an EPUB (iPad, NOOK, etc.) version of Digging Deeper into AutoHotkey click here!

For a PDF version for printing on letter size paper for inclusion in a standard notebook of Digging Deeper into AutoHotkey click here!

* * *

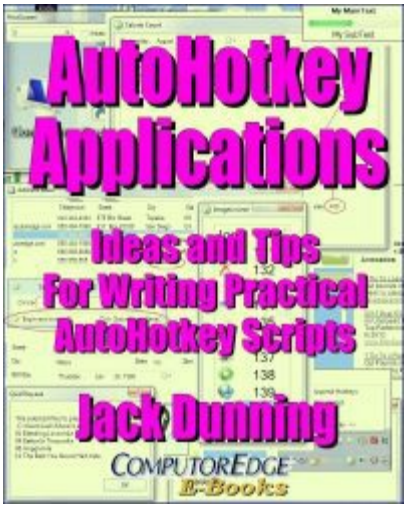


As a convenience for people who don't want to dig through the Web for individual tips, but would like to learn some cool AutoHotkey Hotstring tricks, the e-book *Beginning AutoHotkey Hotstrings* is now available on the ComputorEdge E-Books site and through Amazon. If e-books are not your thing, then it might be worth your time to peruse some of the blogs included in this book and linked at "Beginning AutoHotkey Hotstring Techniques" found under the "AutoHotkey Topics and Series" tab in the top menu bar. They just might inspire your next AutoHotkey script.

Beginning Hotstrings explores the potential of the basic AutoHotkey Hotstring option and how they can aid anyone who uses word processors, text editors, or Web input fields on Windows computers. It's surprising how this one small area of AutoHotkey can add power to your computer through Hotstring menus and the enigmatic Input commmand.

For more details about *Beginning AutoHotkey Hotstrings* (Table of Contents and the entire book index), click here!

* * *

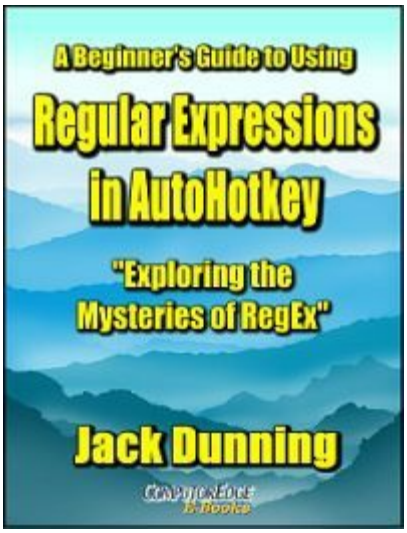


Jack's third AutoHotkey book *AutoHotkey Applications* (preferred, EPUB format for iPad, Android, and computers; MOBI for Amazon Kindle; and PDF for printing) is an intermediate level book of ideas and applications based primarily on the AutoHotkey GUI command. The book emphasizes practical applications. The book is not for the complete beginner since it builds on the information in the other two books. However, if a person is reasonably computer literate, they could go directly to this book for ideas and techniques without the other books. There is an extensive index to the ideas and techniques covered in the back of the book. (Also available from Amazon for the Kindle and Kindle software on other devices.)

For an EPUB (iPad, NOOK, etc.) version of AutoHotkey Applications click here!

For a PDF version for printing on letter size paper for inclusion in a standard notebook of AutoHotkey Applications click here!

* * *



This Beginner's Guide to Using Regular Expressions in AutoHotkey is not a beginning level AutoHotkey book, but an introduction to using Regular Expressions in AutoHotkey (or most other programming languages). To get the most from this book you should already have a basic understanding of AutoHotkey (or another programming language). Regular Expressions (RegEx) are a powerful way to search and alter documents without the limitations of most of the standard matching functions. At first, the use of RegEx can be confusing and mysterious. This book clears up the confusion with easy analogies for understanding how RegEx works and examples of practical AutoHotkey applications. "Regular Expressions in AutoHotkey" will take you to the next level in AutoHotkey scripting while adding more flexibility and power to your Windows apps. (This book is also available at Amazon.com)

Chapter One: Add Tailored Signatures to All E-mail and Documents

"If you do nothing else, use AutoHotkey to put your signature in every document, blog, and e-mail."

Add your formated signature (with e-mail and Web site) to any document, Web page response, or e-mail with easy one-line AutoHotkey scripts. Plus, instantly enter your e-mail address anywhere (no matter how long it is).

There are simple beginning AutoHotkey tricks that anyone with a Windows computer can do. If you need to regularly add your signature to documents, e-mails, or Web comments, then you're going to like this one. See Figure 1.

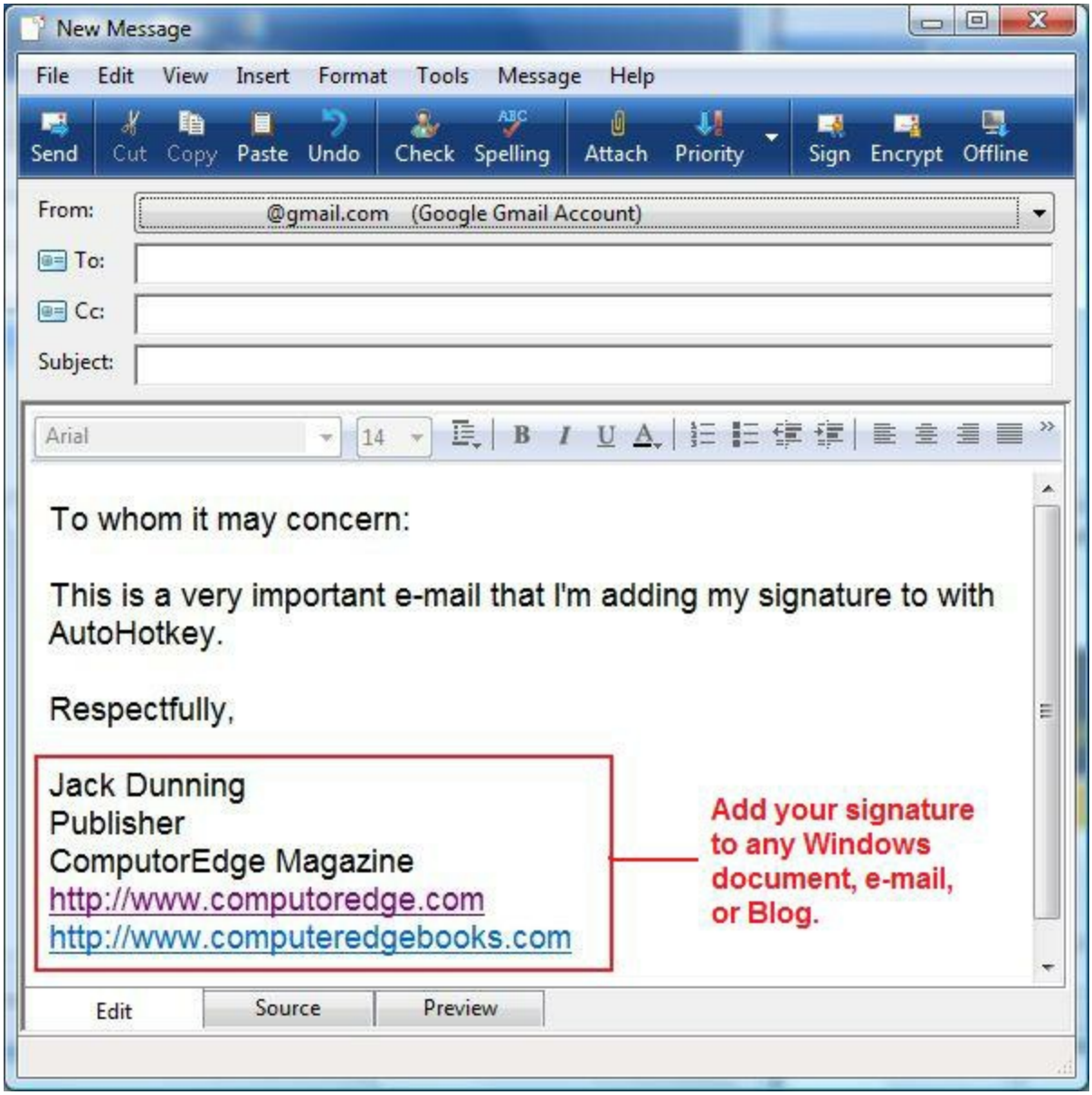


Figure 1. The signature (including company, title and Web addresses) is added to this e-mail by merely typing "jsig" followed by enter, space, or other punctuation. It also works in other Windows documents, program editing fields, and Web pages.

The AutoHotkey tips discussed in this chapter are based upon one of the basics of the free utility software—auto-replacement of text. They are simple to do and will save you a great deal of typing while personalizing all of your documents and correspondence in any form. These techniques work in any version of Windows including Windows 8 and Windows 10. Yes, they even work in the editing fields of the Windows 8 Modern Interface apps.

Adding Your Signature to Anything

The two basics of AutoHotkey (before entering the more complex uses of AutoHotkey pop-up windows) are hotkeys and hotstring replacement. Hotkeys are used to assign computer keyboard combinations (all keys pressed simultaneously) to certain operations. Hotstring replacement is primarily the substitution of strings of text entered from the keyboard with alternative text—usually to expand short abbreviations into longer input. For example, expanding "lol" into "laugh out loud." Here we use hotstring replacement to enter my signature after typing only a few characters. It can be done with one line of code in an AutoHotkey script::

```
::jsig::Jack Dunning{return}Publisher
      {Return}ComputorEdge Magazine
      {Return}http://www.computoredge.com
      {return}http://www.computeredgebooks.com
```

(Although this line appears as four lines for display purposes here, it should be one continuous line in the script.)

This one line added to an AutoHotkey script does the job. If you know how to use Notepad and have the free AutoHotkey installed on your computer, you're ready to go. Anytime I type "jsig" into an e-mail, document, Web blog or other text editing field followed by a space or other punctuation, it's replaced with the following:

```
Jack Dunning
Publisher
ComputorEdge Magazine
http://www.computoredge.com
http://www.computeredgebooks.com
```

This technique is even more useful if you use multiple signatures. For example, I can add two more alternative signatures with:

```
::dsig::Paternalistically,{enter 2}{tab}-Dad
::gsig::Entirely yours,{enter 2}{tab}-GrandDad{enter}{tab}{space}"Too Cool for School"{!}
```

which, when "dsig" is typed, it is replaced with:

```
Paternalistically,

 -Dad
```

and, when "gsig" is typed, it is replaced with:

```
Entirely yours,

   -GrandDad
    "Too Cool for School!"
```

However, you may not want to use {tab} since in Web pages it will move the cursor to the next object which could cause you to jump out of the edit field. A better option is to use {space 5}:

```
::dsig::Paternalistically,{return 2}{space 5}-Dad{return}
::gsig::Entirely yours,
     {enter 2}{space 5}-GrandDad{enter}
```

```
    {space 6}"Too Cool for School{!}"
```

(The second hotstring is wrapped into three lines for display purposes. In the script it would appears as one line.)

The number inside {space 5} indicates how many times to repeat the character.

How It Works

The hotstring replacement or substitution command has two parts: the characters which, when typed, trigger the replacement and the text that replaces the trigger text. The replacement command line always starts with two colons (::). (There can be option characters inserted between the two colons, but in its most basic form the string replacement command needs no options.) The triggering string follows the first set of two colons and is then followed by an additional set of two colons. The replacement string of text follows the second set of colons. When the trigger text is typed followed by a space, return (or enter), or other common punctuation character, that text is replaced with the substitution string. For example, the hotstring command:

```
::lol::laugh out loud
```

replaces the text "lol" with "laugh out loud" whenever "lol" is typed and followed by a terminating space, return, or most other punctuation.

Special keys such as the space bar, enter (or return) key, and tab are added by enclosing keywords in curly brackets (*{space}*, *{enter}* or *{return}*, and *{tab}* respectively). See the online documentation for other special keys and characters. Special characters such as the exclamation point (! for the ALT) also need to be enclosed in curly brackets when not acting as a hotkey modifier. Note the *{alt}* in the *gsig* hotstring above is replaced with the exclamation point (!). See the documentation for the Send command for a list of these characters.

Adding Blocks of Text

If you're adding a long block of text it's unreasonable to attempt to put it all in one line. In fact, in many cases it much easier to use this alternate method for adding signatures:

```
::jsig2::
(
Jack Dunning
Publisher
ComputorEdge Magazine
http://www.computoredge.com
http://www.computeredgebooks.com
)
```

By enclosing the entire formatted text in parentheses, AutoHotkey substitutes the raw text for the trigger string appearing as it does in the snippet of code. Notice that none of the special characters need enclosing in curly brackets. This example yields the same results as the first one-line example *jsig*.

For *gsig* signature, the block replacement command looks like this:

```
::gsig::
(
Entirely yours,

        -GrandDad
        "Too Cool for School!"
)
```

Caution: if you embed the TAB character in the text, even though you can't see it, it will still cause the cursor to jump to the next field when used in a Web page edit field.

By using parentheses to enclose blocks of text, substantial portions of commonly used boilerplate can be condensed to short hotstrings. This may be your preferred method for adding signatures to all of your Windows documents, since it is substantially less complicated than putting all the text on one line and using curly brackets for special keys and characters as shown in the first examples.

If you have any reason to add signatures to any of your Windows correspondence, blogs, and documents, then this is one AutoHotkey technique you should be using.

Note: AutoHotkey hotstrings offer much more capability than shown in these simple examples, such as, adding pop-up selection menus and running other apps. For more on the built-in hotstring options, see Beginning AutoHotkey Hotstrings.

Next time, how to instantly add e-mail addresses, user names, and possibly passwords to data fields.

Chapter Two: Use AutoHotkey to Instantly Insert Your E-Mail Address into Web Forms

"If you do nothing else, shorten your e-mail address to a simple hotkey or hotstring."

It can be a pain to type long, convoluted e-mail addresses. Use AutoHotkey to instantly add yours and maybe your password too.

Hotstrings for Entering E-mail Addresses

As I look at all the ways that I routinely use AutoHotkey, some of the most useful are the simplest. I find that since so many Web sites use the e-mail address for account sign in, I'm often entering my e-mail address into the username field. Some pages let you store the username by saving it in a cookie. Others make you re-enter it every time. Most browsers have the capability to save usernames for each Web site, yet, even though I use those built-in save features, I find that I'm often having to type in my address. No matter how long your e-mail address, it's always awkward to type. The @ sign causes the problem. For some reason I can't remember the location of each punctuation key in the top row of keys. Not only do you have to look for the @, but the SHIFT key needs to be included in the process.

To make the e-mail address problem worse, they are usually unreasonably long—often with odd combinations of characters. It takes a long time to learn to type "computoredge.com" as all one word (and put that strange "o" in it). For many people the situation is worse. That's why this simple AutoHotkey technique is one of the most used features of my scripts. With it I can convert any e-mail address of any length into a two character word which instantly expands into the full address without adding any trailing space. (It's important that no space or return be tacked onto the end of the e-mail address as many accounts won't recognize a username with a trailing character.)

Fortunately this instant e-mail address feature can be implemented with AutoHotkey in one line by adding an option to the hotstring format:

```
:*:j@::mrjackdunning@computoredge.com
```

An option is included in the hotstring line by placing the appropriate parameter between the first two colons, in this case the asterisk (:*:). This asterisk immediately activates the replacement action when hitting the @ character. With the asterisk option included, AutoHotkey requires no space or other terminating character to initiate the hotstring action. Plus, if the combination is preceded by any character other than a terminating character (space, comma, return, period), then it does not fire at all. That means if I type another e-mail address which happens to have a "j@" in it, it won't fire. For example, ritaj@wizziwig.com would not convert the "j@" to "rita*mrjackdunning@computoredge.com*wizziwig.com."

I know…I'm still using the awkward-to-type @ character, but there is a good reason for that. There are almost no words that would start with "j@" (none off the top of my head). By using @ in the two character combination, I greatly decrease the risk that the e-mail address replacement will trigger while I'm typing another word. Of course, any unusual character would do.

If you check out the documentation for hotstrings, you'll see that there are a number of possible option characters which can be inserted between the first two colons including no automatic erasure of the hotstring (b0), capitalization sensitivity (c), omit the last character (o), and a few others.

Note: For more on how AutoHotkey hotstrings options can improve your Windows life, see Beginning AutoHotkey Hotstrings.

Adding a Password

You may or may not want to add passwords to your e-mail address or username to recreate an automatic login. On the downside, passwords appear in the AHK file as plain text. Anyone who has access to your computer could conceivably read your AutoHotkey login script. Even if you compile the file, it is still possible for someone to recover passwords by opening the EXE file with a program as simple as Notepad. While most of the file will appear as garbage, anything that was entered as plain text (the passwords) will appear as such somewhere in the file.

On the plus side, people will not necessarily know where to look for the logins as long as you don't use an obvious filename such as *passwords.ahk*. Plus, even if someone happens to come across your passwords, as long as you don't indicate which passwords apply to which account, then the nefarious person accessing your computer will need to figure it out. Whether you use this technique or not is entirely a matter of how comfortable you feel about someone else getting on your computer. If you save passwords in a browser, you run the same (or greater) risk since anyone on your computer can log into your accounts by merely opening the login page. For that matter, a person can go into Settings in Chrome, or Firefox, and see both the Web site and the password by merely clicking "Show" next to the saved password—although usually required to enter your Windows password. At least with AutoHotkey the villain would need to know which hotstring (or hotkey) to use with which site (unless you use the same password for everything).

Even if you used an AutoHotkey function to encrypt the passwords, it would require a key. Anyone who could access your files would be able to decrypt the passwords using the same key found in the AHK file.

That's your warning. If you feel safe then you can do a complete login with the following type of code:

```
:*:a1@::myaccount@computoredge.com{tab}YourPassword{return}
```

In this case, the instant that "a1" is typed it will be converted to the username "myaccount@computoredge.com", jump to the next field (*{tab}*), enter the password "YourPassword", then press the RETURN (or ENTER) key. It's that simple! You can set up all your logins in this manner using a different hotstring for each. The problem is that you will still need to remember which hotstring is which. Maybe you can write them on a piece of paper in some sort of code.

This same technique can be used to fill in name and address fields on forms, as long as there are always the same number of fields in the same order. Just use *{tab}* each time you need to jump to another field.

Tip: In a browser, you can quickly move data to the proper field by highlighting it and dragging it to the new location.

Logging in with Different Web Pages

I recently received the following AutoHotkey question:

I have some sites that put the username and password on different pages. So I modified the script as follows:

```
:*:a1@::username{return}password{return}
```

However, I need to pause after the first "return" until after the second page loads (i.e. the password page). Also, I'd like to precede the username and password with a line or lines in the same script so that it will first open the browser to the username page automatically. How do I accomplish these two things?

Ron Cerrato

There are many sites which change pages between entering usernames and passwords making the first script

nonfunctional after the first page. The problem is that the script continues to run while the next page loads, thus losing the password to outer space. My first thought may be to use the Sleep command. (Remember, the *Sleep* command placed at key points in a script is used to slow down the execution of an AutoHotkey script. This prevents the execution of the lines of code from outrunning the action of the commands and the loss of subsequent functionality.) The Sleep command could be used to pause the script while the new page loads, however Web page loading times vary considerably—sometimes as long as a few seconds. If the page has not loaded by the time the *Sleep* command times out, the password won't get inserted in the second page.

The WinWaitActive command is also unreliable in this situation since a Web page may be active, yet still loading data. This will also cause the script to continue before it can enter the password in the data field.

An easy way to ensure that the Web page has completely loaded is to force the script to wait for user input with the MsgBox command:

```
:*:a1@::
  SendInput, username{return}
  MsgBox, Wait for next screen!
  Sendinput, password{return}
Return
```

This does require you to press the ENTER key once when you see the new page has loaded, but it's better than re-entering the password after the script misses it. (There may be more elegant ways to solve this problem in AutoHotkey, but this certainly works and is simple.)

Note: When running multiple commands, the hotstring format must switch from the one-line hotstring format to the command structure format starting on the second line, then terminated with the *Return* at the end of the snippet.

As for loading the Web browser first, that's as easy as copying the URL from the Web login page and loading it with the Run command at the beginning of the script:

```
:*:a1@::
  Run, https://www.website.com/loginpage.html
  MsgBox, Wait for next screen!
  SendInput, username{return}
  MsgBox, Wait for next screen!
  Sendinput, password{return}
Return
```

Again, with the variable times for loading Web pages, it may be easiest to use a *MsgBox* to suspend script execution.

Just in case the right Web page is already active, you may want to make the Run conditional with the IfWinNotActive command:

```
:*:a1@::
  IfWinNotActive, [WinTitle]  ;WinTitle as shown by Window Spy
     {
         Run, https://www.website.com/loginpage.html
         MsgBox, Wait for next screen!
     }
  SendInput, username{return}
  MsgBox, Wait for next screen!
  Sendinput, password{return}
Return
```

As discussed in Chapter Six, the window title can be found by using Window Spy after loading the Web page. This

prevents the browser from attempting to reload an active page. The cursor must land in the username field.

* * *

Ron Cerrato and I exchanged e-mails about the problem of using AutoHotkey for logging in when the operation spans two Web pages—one for the username and one for the password. While the method for using *MsgBox* command worked for me on my bank site, the loss of focus caused by *MsgBox* prevented complete input for Ron. Eventually, he went back to using the *Sleep* command. I had avoided the *Sleep* command because the Web page loading times are hard to determine. Ron solved the problem by using very long Sleep times (7.5 seconds and 5.0 seconds). Here's Ron's final e-mail:

Thanks a lot!!! This works great:

```
:*:a1@::
  Run, https://mybank.com/obc/forms/login.fcc
  Sleep, 7500
  SendInput, username{return}
  Sleep, 5000
  Sendinput, password{return}
Return
```

Third Edition Update: Chapter Fifteen, "Tracking the Original Edit Window," of *Beginning AutoHotkey Hotstrings* offers a sure-fire technique for saving the current page, then later returning to it using the WinActive() function and the WinActivate command.

Ron states:

I now have five green "H's" in the System Tray: Login, Autocorrect, Reminder, Dictionary, StripAllReturns.

This last comment about five separate AutoHotkey scripts running simultaneously brings up an important question. How many is too many? If all the green icons in the System Tray don't bother you, then there is nothing technically wrong with running a number of separate scripts. One advantage of doing this is that each app is less likely to conflict with another. However, if you want to combine scripts, then there are a few things that you need to understand about how AutoHotkey processes scripts.

In the AutoHotkey Applications book, I devoted Chapter Thirty-seven, "Combining Apps into One Script" to this problem. Rather than repeat what I said in that chapter I wrote a short piece with a slightly different take at this *ComputorEdge* AutoHotkey page How an AutoHotkey Script is Processed. This information is important because it often determines whether or not a combined script runs properly. Although beyond the scope of this book, an understanding of how AutoHotkey processes scripts when first loading helps you debug many scripts—even if they are standalone apps.

You can find even more detailed descriptions of how AutoHotkey processes scripts at Jack's AutoHotkey Blog.

Chapter Three: Use AutoHotkey to Instantly Turn Hard-to-Type Jargon into Hotstrings

"If you do nothing else, use AutoHotkey to instantly turn hard-to-type jargon into hotstrings."

> *Most people have specialized vocabulary in their occupation. Use short hotstrings to add those terms to any document without spelling or typing hassles.*

Anyone who regularly works with computers finds that frequently they must add specialized vocabulary to their writing and data fields. Often these terms—unique to the occupation—are too long to type so we abbreviate them for ease of input. However, these abbreviations may be confusing to some and meaningless to others. The answer is to add the complete terms to your documents every time you need them without typing anything more than an abbreviation. Then people will know exactly what you're talking about.

For example, typing *ComputorEdge* is a real pain. Between the capital E in the middle of the word and the silly "o" rather than an "e" it is extremely awkward to type. I could abbreviate *ComputorEdge* to *CE*, which may work find for most *ComputorEdge* readers, but to the average person *CE* may mean Consumer Electronics, CE marking, Continuing Education, or the Corps of Engineers. At a time when clarity is important, abbreviations get in the way.

I added these often used, yet specialized, words to my main AutoHotkey script which launches every time I log in:

```
::ce::ComputorEdge
```

When I enter "ce" followed by a space or other punctuation, I get *ComputorEdge*.

Here are a couple more terms I regularly use:

```
::ahk::AutoHotkey
::ahkl::ahkscript.org
```

Since I do an AutoHotkey blog, I often use the word AutoHotkey. I could use the abbreviation AHK, but that could be confused with Acid House Kings (Swedish band) or American Health Kennels. (Okay, maybe that one would work.) But even more useful is the second hotstring, "ahkl", which is replaced with "ahkscript.org" when followed by an ENTER key. I use this when I'm adding Web references to AutoHotkey commands.

Third Edition Update: Since the main AutoHotkey site has moved back from *ahkscript.org* to *www.autohotkey.com*, I no longer need the *ahkl* hotstring. Don't worry. The old *ahkscript.org* redirects to *autohotkey.com* for commands and other documentation. I no longer need to use the technique which follows, but the information is useful for other similar problems.

Before the switch back to *autohotkey.com*, if I Googled an AutoHotkey command, I would usually get a results list with the documentation for the original AutoHotkey version at the top. I could use that Web page, but command explanation may not be complete for the latest accepted version of AutoHotkey_L. I could search specifically for the AutoHotkey_L documentation, but that's time consuming compared to using the following technique.

I search for "AutoHotkey gui" with Google—usually by highlighting the words in a column I'm writing, right-clicking and selecting "Search Google for 'AutoHotkey gui'." I click "GUI - AutoHotkey" which is at the top of the search results, opening the GUI documentation on the original AutoHotkey site (*www.autohotkey.com*) (see Figure 1). Since, with the exception of the domain name, the URL for the AutoHotkey_L documentation is normally identical, I highlight the domain portion of the address and type "ahkl" followed by a return.

Figure 1. A Google search for an AutoHotkey command will bring up the Web site pages for the original version of AutoHotkey at the top of the list.

The URL is replaced with the equivalent AutoHotkey_L Web site URL (*ahkscript.org*) and the browser is redirected to that page (see Figure 2). I now have the URL I want to use in my referencing of the command.

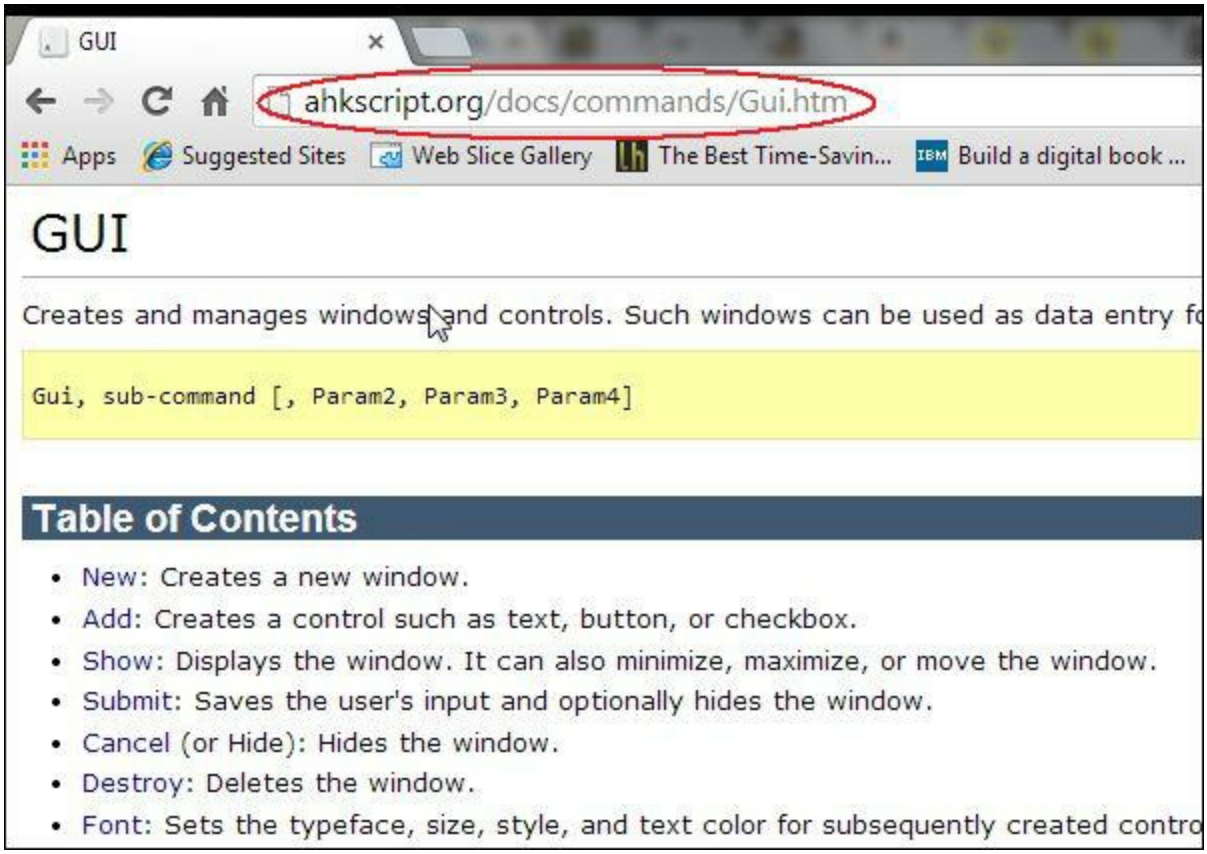


Figure 2. The browser is redirected to the Web site with the documentation for the currently accepted version of AutoHotkey (AutoHotkey_L).

I'm not sure how much use most people would have for this type of trick, but it sure saved me a lot of time when the AutoHotkey Web site confusion still existed. However, expanding commonly used abbreviations for jargon is certainly something that you should do on your Windows computer—both to save you time and add clarify for others. I'm sure that you have your own set of words that are normally a hassle to type/spell.

If you use the AutoHotkey AutoCorrect script (discussed in this *ComputorEdge* AutoHotkey page and Chapter Twenty-three, "Add AutoCorrect to Your Windows PC" of the Digging Deeper into AutoHotkey book) for commonly misspelled words (and you should), then you can just add your special vocabulary to the AutoCorrect hotstring list.

Chapter Four: Adding Currencies, Special Symbols and Fractions, Plus Today's Date

"If you do nothing else, use AutoHotkey to add the special characters you want to your keyboard."

> *While there are other ways to add foreign currency symbols and special characters to your documents, files, and Web edits, nothing is easier than AutoHotkey. Or, instantly add today's date to any document.*

Every once in a while I need to insert a special character into a document, but the key doesn't exist. My keyboard supports a dollar sign, but no British Pound (£) key or Euro (€) key. For math or casual estimates my keyboard lacks a plus or minus symbol (±). In marketing, it takes some computer gymnastics to add the registered trademark sign (®) or copyright sign (©). Would you like to be able to add the condensed ½ to any document or Blog rather that the three character 1/2? I often use the following AutoHotkey technique for adding symbols. For example, I was able to add the degree symbol (°) for range of view to an Oculus Rift VR helmet article without consulting the Windows Character Map (see Figure 1). (Run => *CharMap* to open.)

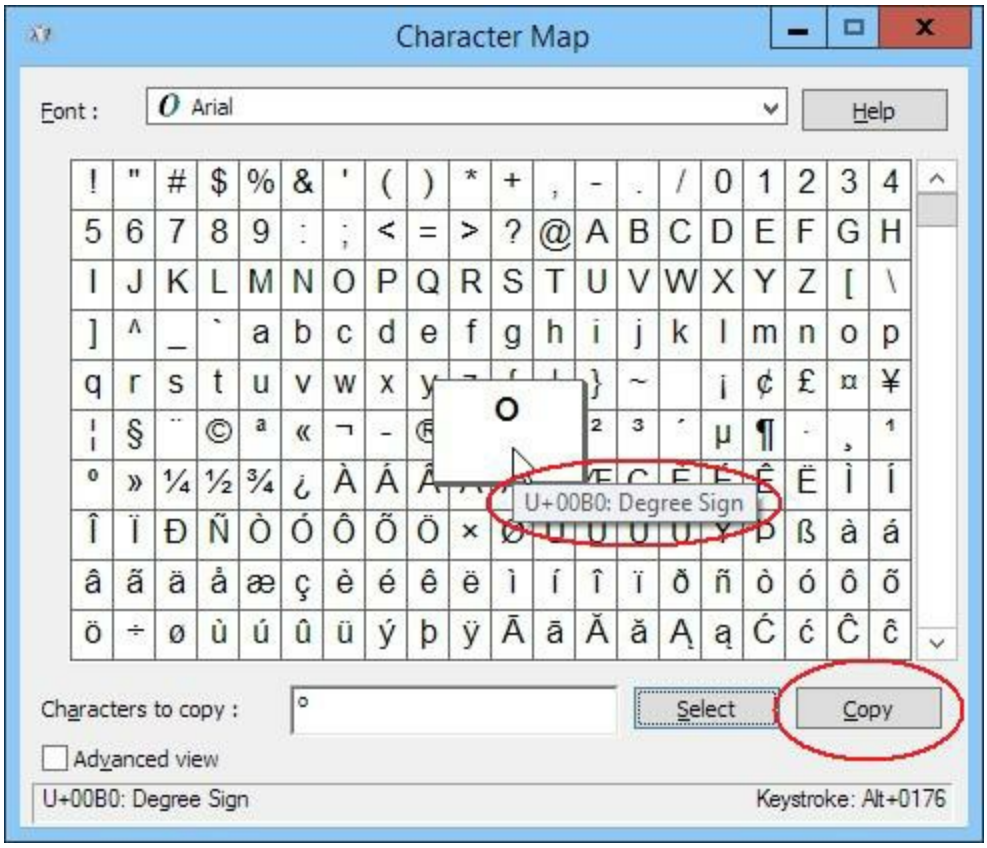


Figure 1. Windows Character Map (CharMap) can add special characters to your documents.

You can add all these special characters (and any others that you need) without making any changes to your keyboard. Use AutoHotkey's hotstrings to auto-replace short key sequences with the special characters you need. If you do nothing else with AutoHotkey, do this.

Automatically Adding the Special Characters to Documents and Editing Fields

The great thing about this AutoHotkey technique is that it is simple to do. All you need is a short text file with as little as one line of code. The following is a list of some of the special hotstrings I use. For currencies:

```
:*:pound*::£
:*:euro*::€
:*:yen*::¥
:*?:cent*::¢
```

A hotstring replacement line always contains two sets of double colons (::). The first set of colons starts the hotstring line and may contain option code letters between the first two colons such as the * and *? seen above. The hotstring and replacement characters are separated by the second set of double colons (::). After loading the script, whenever typing *pound**, £ appears; after entering *euro**, € replaces it; add the ¢ sign to any document by typing *cent**.

The asterisk * option between the first two colons forces hotstring execution immediately after pressing the last hotstring character (in this case it happens to be another asterisk *). This eliminates the need to hit a punctuation key or spacebar to activate the replacement. Since a number (which does not activate a text replacement) usually follows a currency symbol, the added * option ensures an action which would not normally occur.

The second asterisk acts merely as an activator, not to be confused with the first option. The additional asterisk at the end of each hotstring creates a unique hotstring using a character (*) you would not normally type. This helps prevent the accidental insertion of the special characters at unexpected times. For example, without the asterisk as part of the *pound** hotstring, you would get the £ symbol every time you typed, "I want a pound of flour," ("I want a £ of flour"). I'm not likely to type the word "pound*" with an asterisks unless I want the £ character.

The question mark ? option between the first two colons forces the hotstring replacement to occur even when in the middle or at the end of another word. This is especially important for the cent sign ¢ since it normally will fall at the end of a number without any intervening space. (The order of the options, *:*?:* or *:?*:*, make no difference.)

When writing the script, you can use Windows Character Map to copy the symbols and paste them directly into the script text file.

See a few of my other special hotstrings:

```
:*?:+-::±     ; plus or minus sign
:?*:(c)::©
:?*:(r)::®
:?*:(tm)::™
:*?:`.` `.` `.::…  ;  ellipsis
```

The plus or minus sign is cool because all you need to do is type the plus (+) sign then the minus (-) sign. It instantly converts to ±.

I picked (c), (r), and (tm) because those are the character strings used for copyright, registered trademark, and trademark on a typewriter.

In the last example, I use three periods with spaces between them to create the ellipsis character (…). Since both the period and space are activating characters, I must use the accent/backtick escape character (`) before each in the hotstring. I could have eliminated the spaces in between, but I didn't want ellipses to start popping up each time I typed a number of periods in a row—although I can't remember the last time I did that.

For the few fractions available and other symbols I added:

```
:*:1/4*::¼
:*:1/2*::½
:*:3/4*::¾
```

```
:?*:deg*::°
```

Since I had previously implemented the AutoHotkey AutoCorrect script for the most common misspelled words (discussed in the Digging Deeper Into AutoHotkey e-book), it was easy to add these lines to the end of the file. That way I didn't need to run a separate script for the symbols.

Tip: If you want to prevent the activation of a hotstring, click a mouse button before you complete typing the activating hotstring. The click resets the hotstring monitor.

If there's only one special character that you need readily available, then that's all you put into the AutoHotkey script. You can also compile the script into an EXE file which will run on any Windows computer. That eliminates the need to install AutoHotkey on every machine you use.

If you want to eliminate the steps involved in finding and using certain special characters, then this is an AutoHotkey trick that you really should use.

Third Edition Update: Even though the hotstring examples above are relatively easy to remember, I still find that I occasionally need to go back and remind myself how to add them. When writing the book Beginning AutoHotkey Hotstrings, I developed a script for adding special characters through a pop-up menu. See Chapter Ten, "Add Currency (and Other) Symbols with AutoHotkey Hotstring Menus." If you need to quickly add a number of different types of symbols, then a selection menu may be your best bet.

Tip for Adding Today's Date to Any Document

A similar, yet slightly more complicated, use of AutoHotkey makes a text hotstring string act as a hotkey—running a series of commmands. Valuable in those situations where the results may vary, the hotstring adapts to the current situation. In this example, AutoHotkey instantly enters today's date (formatted Month, Day, Year—e.g. June 27, 2016) into the document whenever you type the text string *Anow*:

```
:c*:Anow::
  FormatTime, CurrentDateTime,, MMMM d, yyyy
  SendInput %CurrentDateTime%
Return
```

To run commands using a hotstring, the first code must appear on the next line after the hotstring definition (*:c*:Anow::*)—treating it as a command rather than replacement text. In this case, I used *Anow* (with a capital *A*) because it is not a word and unlikely to appear in any text, especially since the first letter is capitalized. By placing the *c* option between the first two colons (*:c*:*), AutoHotkey requires the hotstring to contain the same capital letter (case sensitive). This reduces ambiguity and the likelihood of accidental firing when entering the hotstring.

The asterisks between the two colons (*:c*:*) tells AutoHotkey to execute the command immediately when pressing the last letter without any terminating keystroke such as a comma or period. This hotstring formats the current date (by default) and sends it to the active document or text editing field. The hotstring in terminated with the *Return* command on the last line. This and similar techniques are discussed in Chapter Twenty-six of the AutoHotkey Applications e-book.

Chapter Five: Web Site Searches Made Easy

"Set up a hotkey combination to make searching your favorite Web site easy with AutoHotkey."

> *Anyone can put this simple AutoHotkey script to good use for quickly searching their favorite Web site. Plus, in this chapter we build a slightly more advanced app by using a graphic user interface (GUI) and Menu command to add more bells and whistles.*

Often when researching a topic, we just open Google in our browser and do a keyword search. This is easy enough and yields a plethora of results—sometimes too many. If we already have our browser open, we can usually highlight text, right-click, selecting search, then finding the selected text with our default search engine. But what if we use an application that doesn't support a Web search or would like to find the selected text in a particular site such as Wikipedia? A simple AutoHotkey script provides us with a hotkey combination that directly executes the search function for a specified site. For example:

```
^#g::
  Send, ^c
  Sleep 200
  Run, http://www.google.com/search?q=%ClipBoard%
Return
```

After loading on a Windows computer with the free AutoHotkey software installed, this snippet of code, when saved in a simple text file (use Notepad or any other text editor) with the AHK extension (e.g. *GoogleSearch.ahk*), initiates a Google search for the highlighted text in any Windows program whenever simultaneously pressing the CTRL+WIN+G key combination (Control+  +G). If your default browser is not open, the script opens it.

The three characters (^#*g*) before the double colon set up the hotkey combination (Control+  +G) for initiating the routine. Once the script is loaded, the routine runs every time the Control key (^), the WIN key (#) and G are pressed simultaneously on the keyboard.

The AutoHotkey Send command executes the Windows key combination CTRL+C (the copy command) placing the selected text into the Windows Clipboard for later use.

Although omitted from many examples that you may find on the Web, the *Sleep 200* line of code is important. Without it, you may need to hit the hotkey combination twice. The reason is that the computer needs some time to copy (CTRL+C) anything into the Windows Clipboard—however short. Using the Sleep command allows a pause (200 microseconds) before moving on to the next step in the script. Otherwise, the script may run so fast that it attempts the search with the previous contents of the Clipboard.

The line of code with the Run command is the heart of the script. It opens Google and executes a search using the contents of the Windows Clipboard. Not coincidentally the word *ClipBoard*, a built-in AutoHotkey variable, always contains the contents of the Windows Clipboard. It can be evaluated and used in other commands by placing percent signs around it (*%ClipBoard%*). Doing this substitutes the saved text in the Windows Clipboard for the term *%ClipBoard%* in the *Run* command line .

The Return command at the end of the line merely marks the end of the routine. That's it!

If you wanted a similar hotkey combination for Wikipedia, then it might look like this:

```
^#w::
  Send, ^c
  Sleep 200
  Run, http://en.wikipedia.org/wiki/Special:Search?search=%ClipBoard%
Return
```

I've seen on the Web similar examples of this script which place all of the commands on one line. Be warned that this will not work since multiple commands must appear on separate lines after the initial hotkey combination (marked by the double colon) and enclosed with a *Return*. However, a sole command appearing immediately after the double colon (such as the *Run* command) will execute while requiring no Return command—as long as it is the only command.

Setting Up Your Own Favorite Web Site Search

The hardest part of setting up your own specialized Web site search is determining the URL (Web address plus search parameters) that you should use. Suppose you want to set up a hotkey combination to search your favorite recipe site—in this case *Food.com*. Open your browser, navigate to your favorite site and do a search using their built-in search tool. After the search, the URL with the included search parameters will show up in the navigation field (see Figure 1).

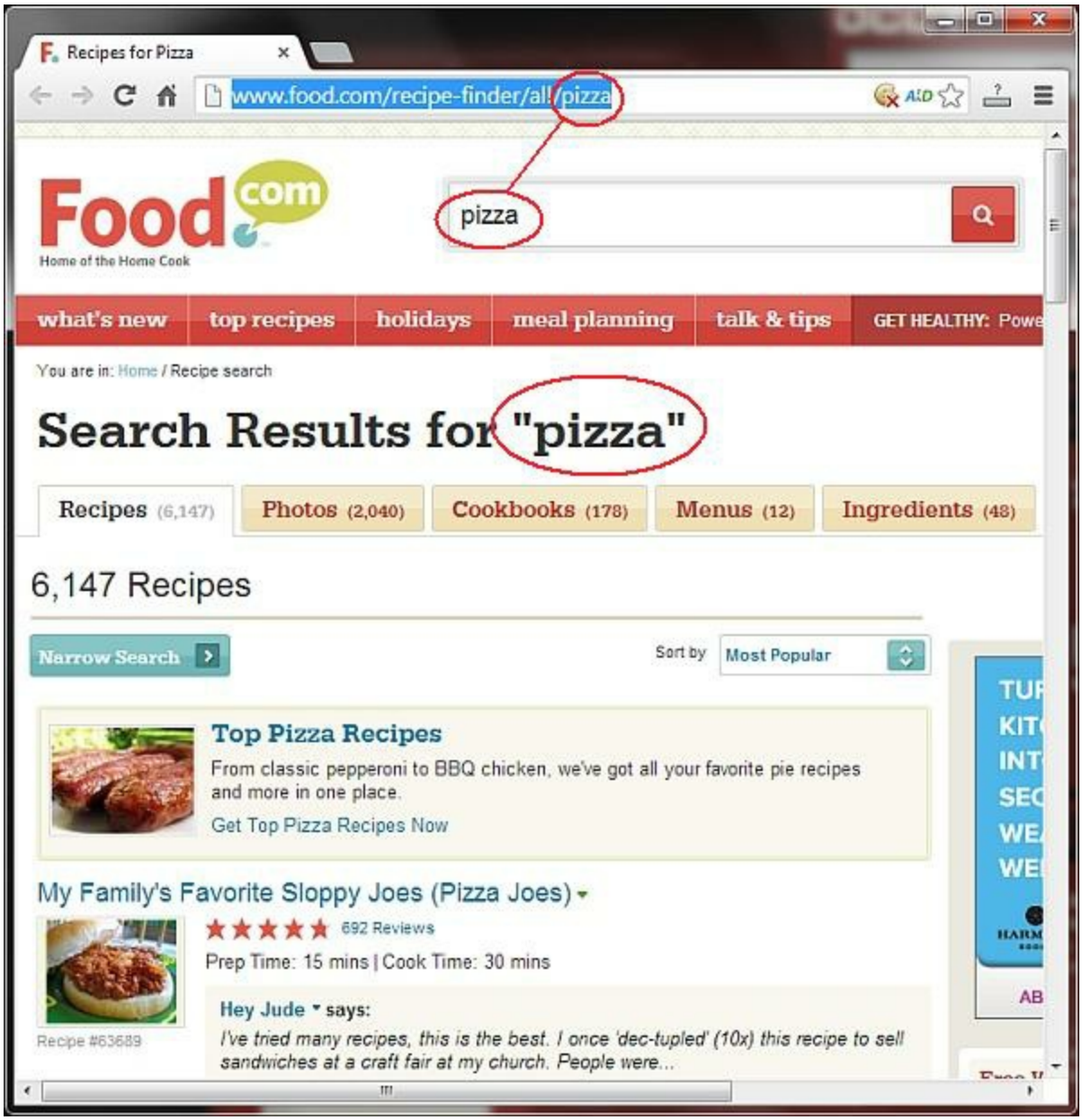
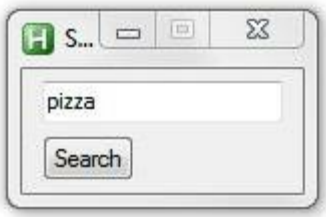


Figure 1. The navigation field of your Web browser will display the parameters needed to execute a search of the site.

In this case, *Food.com* merely adds the name of the food to the end of the search URL. The following snippet will execute a recipe search:

```
^!r::
  Send, ^c
  Sleep 200
  Run, http://www.food.com/recipe-finder/all/%ClipBoard%
Return
```

Note that the entire URL is copied (including the *http://*) with only the search keyword being replaced with *%ClipBoard%*. It's that simple. Now, after loading, whenever you highlight a food item on your Windows computer in any document or Web page and hit CTRL+ALT+R simultaneously (! is the ALT key), a recipe search of *Food.com* will be conducted using the selected food.



But it's not always convenient to highlight the food item—especially if you're not in a document or on a Web page which contains the words. Unlike Google or Wikipedia which you might want to use whenever you come across curious words in a document or on a Web page, you're not likely to have the proper words available for highlighting when you're searching for a new recipe. Wouldn't it be better just to input the food into a window (such as the one displayed on the left), then immediately search for the recipe? Next we will look at the few lines of code needed to open an input window for entering the recipe keyword search. It's simple and only takes three additional lines of AutoHotkey code.

Making a Pop-up Search Window

As novice AutoHotkey users we tend to confine ourselves to building hotstrings and hotkeys (such as found in this "Introduction to AutoHotkey") or the previous easy Web search script. But we are only one step away from adding professional pop-up windows to our quick apps. In many cases, adding an AutoHotkey pop-up control can turn a good tool into a great tool.

Above a short, easy script for launching favorite Web page searches of highlighted keywords is demonstrated. Finally, *Food.com* is the target of searches for recipes. However, it's unlikely that we will just happen to come across the right recipe keywords unless we are reading an article about a particular food. Wouldn't it be nice to pop up an input box (as shown above), type in the recipe keywords, then launch the search? Here is a new script which adds a search word input pop-up:

```
Gui, Add, Edit, vMySearch
Gui, Add, Button, Default gSearch, Search
Return

 ^!r::
  Gui, Show
 Return

 Search:
   Gui, Submit, Nohide
   Run, http://www.food.com/recipe-finder/all/%MySearch%
 Return
```

While this new script has only a few more lines than the original, it is organized in a slightly different manner. The first two lines (which must appear first in the script) create the GUI (Graphic User Interface) window which allows user input. The hotkey combination CTRL+ALT+R (^!r) now merely displays that newly created input window with the *Gui, Show* command. Then the subroutine (*label*) called *Search:* does the work of saving the user input to a

variable (*MySearch*), opening the default Web browser and running the Web page search with the user inputted keywords.

While a more detailed explanation of how this script works follows, as a beginner someone could merely substitute the URL and search parameters for their favorite Web site into the *Run* command line (as discussed in the beginning of this chapter) to add this search pop-up window capability to their favorite AutoHotkey search script.

The AutoHotkey GUI (Graphic User Interface)

The AutoHotkey GUI (Graphic User Interface) consists of a powerful set of tools for making pop-up windows. While they may be a little intimidating for the novice, they implement a new level of capability to even the simplest scripts. GUIs add editing and input fields, buttons, lists, dropdown menus, and many other objects which enhance the usability of AutoHotkey. The GUI pop-ups do have their peculiarities, but once understood they can be simple to use. The GUI shown above, which consists of a text input field and a button to start the search, is about as easy as it gets. It is a good start for any AutoHotkey beginner who wants to move to the next level.

The GUI command consists of sub-commands for setting up a pop-up window. The three most important sub-commands are *Add* (*Gui, Add*) which places objects in the pop up window, *Show* (*Gui, Show*) which makes the window pop up onto the screen, and *Submit* (*Gui, Submit*) which captures any input (text, menu selections, etc.) from the user.

When using the Gui, Add command there are various objects which can be included in the window such as *Text*, *Edit*, *Button*, etc. (To see what many of the objects do and look like, check out this *ComputorEdge* AutoHotkey page.) For our easy search script we only need an *Edit* object (*Gui, Add, Edit*) for inputting the recipe key words and a *Button* object (*Gui, Add, Button*) for starting the search:

```
Gui, Add, Edit, vMySearch
Gui, Add, Button, Default gSearch, Search
```

These two lines actually place the GUI into computer memory although they are not be immediately displayed on the screen. (For the pop-up to display as shown above, the Gui, Show command must be used.) These two lines should appear in the first part of the script before any hotkeys or subroutines. This first part of every AutoHotkey script is called the auto-execute section and runs when the script is first loaded. If they appear after any hotkeys, they will not run. (These GUI setup lines may appear within a hotkey combination, but this may cause errors as discussed below.)

Note: When we start adding more than just hotkeys and hotstrings to an AutoHotkey script, it's important to understand how AutoHotkey scripts are read by the computer. See this AutoHotkey page and these blogs for a better understanding of how AutoHotkey reads a script.

An important concept in AutoHotkey GUI windows is that once one is created it continues to exist in memory whether or not you can see it (or maybe even close it with the little "x" in the upper right-hand corner). This means that you can bring it back with the *Gui, Show* command. If you want to remove it permanently, then you must either destroy it (*Gui, Destroy*) or exit the AutoHotkey script (*ExitApp*).

There are numerous options available for every GUI command, but we don't need to deal with most of those in this beginner script. The two options of interest are *vMySearch* in the *Edit* line and *gSearch* in the *Button* line. These options demonstrate a couple of the more important AutoHotkey concepts which apply to most GUI commands and objects. An understanding of how these two work aids you later when exploring other powerful possibilities with GUI windows.

The vVariable Option

In a *Gui, Add, Edit* line, the letter "v" in front of a variable name (any name you choose) in the option area (after the third comma) creates a variable by that name. When the *Gui, Submit* command is issued, any input from the GUI objects is stored in the associated "v" variable. In this case, *MySearch* is the "v" variable where the entered search words will be stored. Rather than using the selected words stored in the Windows ClipBoard (as was done previously), the script now uses the input stored in *MySearch*.

A particular *vVariable* can only be created once when a script is running. If you try to recreate the same variable by running the *Gui, Add* line with that "v" variable more than once, you will get an error. For that reason, it is important for the *Gui, Add, Edit* line to appear outside the hotkey routine. Otherwise after the first instance, an error will occur each time you use the hotkeys as it tries to recreate an existing variable. (There are other ways to avoid this error as discussed in the *AutoHotkey Applications* e-book such as using *Gui, Destroy* at the end of the routine, but most often placing the *Gui, Add* lines in the auto-execute section at the beginning of the script will be cleanest.)

The gLabel Option

In the *Gui, Add, Button* line, the letter "g" (for goto) in front of a name (any name you choose) in the options area (after the third comma, but before the fourth comma) designates a subroutine (label) for the GUI to run when the object is clicked (or changed—depending upon the type of object). This is used to add action to a GUI. In this case the option *gSearch* calls the label (subroutine) *Search:* . (Note that the first line of the actual subroutine (label) is always its name followed by a colon ":" while the last line is the *Return* command.) In this case the label (*Search:*) first uses the Gui, Submit command to save the user input to the variable *MySearch*, then uses the Run command to execute the search using the contents of *MySearch* (*%MySearch%*). This is similar to the prior search line which uses *%ClipBoard%* rather than *%MySearch%* for the search keywords.

The *Gui, Submit* command saves all user input from all the GUI objects with a *vVariable* option to its associated variable name. The *NoHide* option merely prevents the pop-up window from disappearing from the screen after the *Submit*. If you want the window to close, remove the *NoHide* option from that line. (Remember, the pop-up will still be available in memory even if it is closed.)

If the *gLabel* option is used in a *Gui, Add* command, then the label (subroutine) must also exist. Otherwise, AutoHotkey will throw an error when loading. At a minimum, the label name (followed by a colon) must be included in the script somewhere (anywhere as long as it appears after the auto-execute section which is always in the beginning of the AHK script file).

The Hotkey Combination and Gui, Show

Note that the only line of code appears within the hotkey combination: *Gui, Show*. That's because the only thing we need to activate the GUI is to *Show* it. There is always a temptation to place all of the GUI commands inside the hotkey combination, but as pointed out above, this will likely cause errors when the hotkeys are hit a second time—unless the GUI is destroyed after the action is completed.

The logic of the script is:

1. The GUI window is loaded into memory.

2. The hotkey combination (CTRL+ALT+R) displays the pop-up GUI window.

3. Clicking the Search button, submits (saves) the user input and launches the search.

The script waits patiently between each step.

Many beginners confine themselves to creating hotstrings and hotkeys. There is nothing wrong with that as long as

AutoHotkey is doing what they want it to do. The GUI objects may look confusing and they do have some quirks which must be considered. However using these graphic window objects brings AutoHotkey scripts to an entirely new level. A complete understanding of all of the possible GUI options is not necessary. The basic functionality of AutoHotkey GUI pop-ups is built into them and automatic without using many of the possible options. But most important are the *vVariable* and *gLabel* options demonstrated in this simple script which unleash most of the power that the GUIs have to offer.

Third Edition Note: AutoHotkey GUIs quickly create useful applications. The book AutoHotkey Applications demonstrates how each GUI type (*MonthCal*, *ListView*, *TreeView*, etc.) may be used in practical applications.

What if we don't want to be forced to remember the hotkey combination? Next, add this search feature to the right-click menu of the AutoHotkey icon in the System Tray with one line of code.

Adding an Item to the AutoHotkey System Tray Right-click Menu

In the first two parts of this chapter we've put together short AutoHotkey scripts which, first, start searches of specific Web sites by selecting words in any Windows document or Web browser, then hitting the assigned hotkey combination. Next, the script was modified to use a pop-up a window for entering the keywords before the search of targeted Web sites (see Figure 3). The next step, by inserting only one line into the script, adds the search feature as an option in the right-click menu of the AutoHotkey icon in the System Tray.

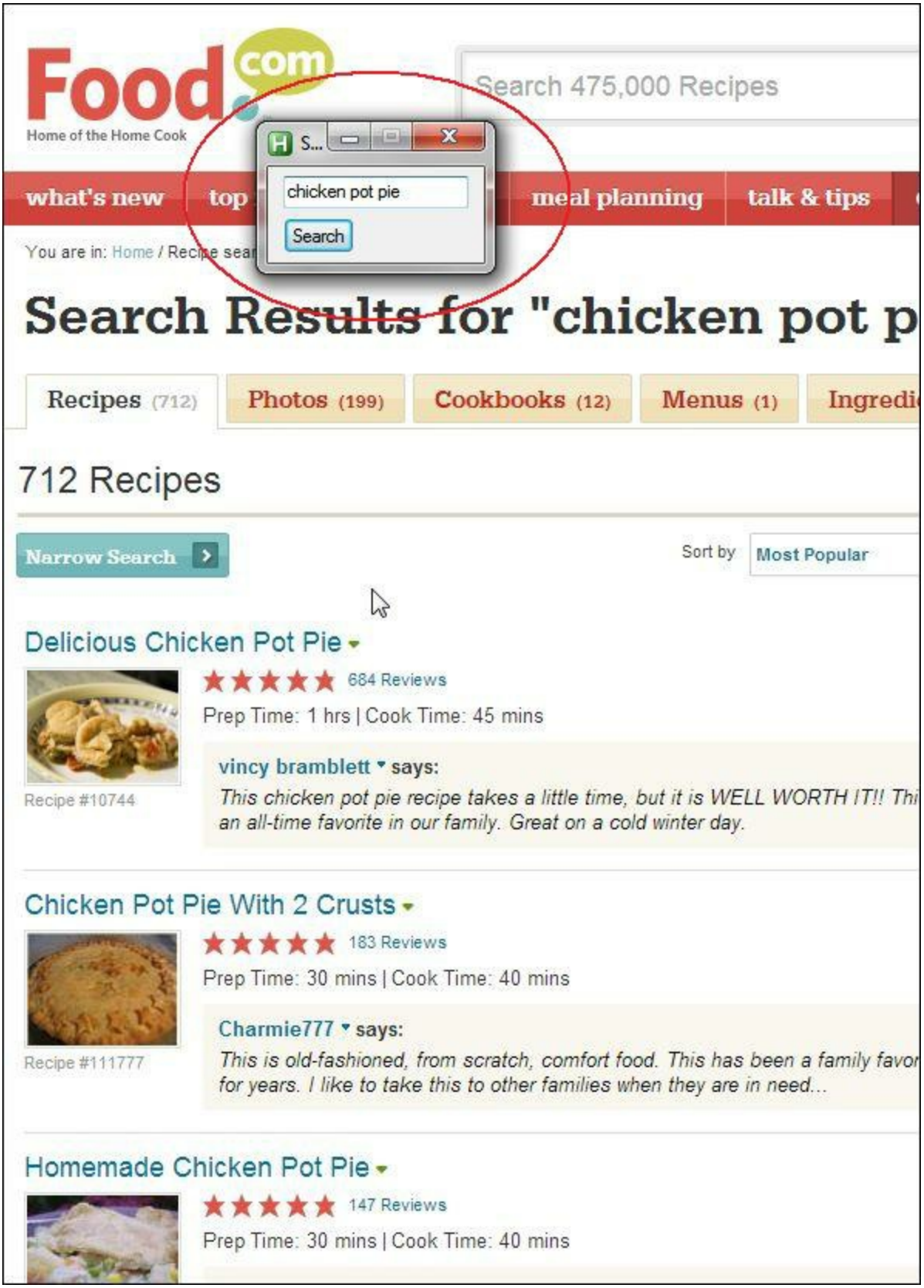


Figure 3. A pop-up was added to initiate searches of specific Web pages—in this case Food.com.

Placing items in the AutoHotkey System Tray right-click menu is easy with the Menu, Tray, Add command. Whenever this command is executed it places another item at the bottom of the menu. For example:

```
Menu, Tray, Add, Search Food.com, ^!r
```

places the text "Search Food.com" as an item at the bottom of the menu for the original script (see Figure 4).

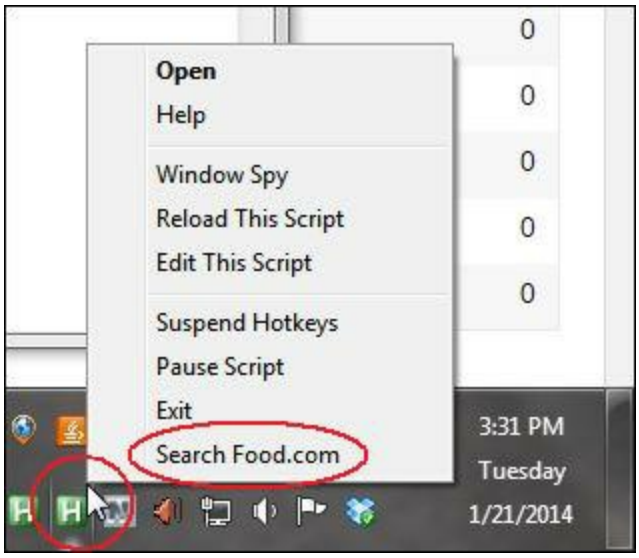


Figure 4. The Menu command is used to place an item at the bottom of the AutoHotkey icon System Tray right-click menu.

The "Search Food.com" parameter is the text displayed in the menu item and *^!r* is the hotkey combination (CTRL+ALT+R discussed above) which calls the routine. Selecting this option from the right-click menu has the same effect as using the hotkeys to start the search. This eliminates the need to memorize the hotkey combination.

Add that one line to the beginning of the last script and you get the following:

```
Menu, Tray, Add, Search Food.com, ^!r

 Gui, Add, Edit, vMySearch
 Gui, Add, Button, Default gSearch, Search
 Return

 ^!r::
  Gui, Show
 Return

 Search:
   Gui, Submit, Nohide
   Run, http://www.food.com/recipe-finder/all/%MySearch%
 Return
```

It is really that simple!

Eliminating the Hotkey Combination

Maybe we don't want to use hotkey combinations at all? We can replace the hotkey with a label name which runs the same subroutine. In this case *^!r* is replaced with the label *RecipeSearch*:

```
Menu, Tray, Add, Search Food.com, RecipeSearch
```

Now the script looks for the label (subroutine) *RecipeSearch:* when the option is selected from the right-click menu. Therefore, the hotkey combination must be replaced with:

```
RecipeSearch:
  Gui, Show
 Return
```

Here is the entire script with the hotkey eliminated:

```
Menu, Tray, Add, Search Food.com, RecipeSearch

 Gui, Add, Edit, vMySearch
 Gui, Add, Button, Default gSearch, Search
 Return

 RecipeSearch:
  Gui, Show
 Return

 Search:
   Gui, Submit,
   Run, http://www.food.com/recipe-finder/all/%MySearch%
 Return
```

You may think that you can do the same thing with the selected text Web site search technique with the same line of code. While you can add the option to the same right-click menu, a little more code is required to make it work. The problem is that when you right-click on the System Tray icon the original window loses focus and is no longer the active window. The copy function (CTRL+C) for saving the selected text to the Windows Clipboard will not work unless the window with the selected text is active. Therefore, before the script can run properly the last window must be reactivated.

While this is a little more complicated, the easiest way to resolve the problem is to use a combination of the two techniques above. A new label (*RecipeSearch* subroutine) is created for the right-click menu which activates the last window by sending the Windows hotkey combination ALT+ESC. This standard Windows hotkey combination cycles to the previous active window. This is done in AutoHotkey with the *Send* command (*Send, !{Esc}*). Then, using the same AutoHotkey *Send* command, the hotkey combination which activates the search is issued (*Send, ^!r*). In this case, we modify the *Food.com* search example above:

```
Menu, Tray, Add, Search Food.com, RecipeSearch
 Return

 RecipeSearch:
   Send, !{Esc}
   Send, ^!r
 Return

 ^!r::
  Send,^c
  Sleep 200
  Run, http://www.food.com/recipe-finder/all/%ClipBoard%
Return
```

Be sure to add the Return after the *Menu, Tray* line to mark the end of the auto-execute section of the script which runs when it's first loaded. Otherwise, the *RecipeSearch:* label will immediately execute on loading.

The problem of adding hotkeys which require a specific active window to right-click menus has confused many AutoHotkey users. While there are other AutoHotkey commands such as *WinActivate* which are designed to open the right window, I've found that simply using the Windows ATL+ESC combination which jumps back to the last active window is far easier than some of the other AutoHotkey gymnastics. For some AutoHotkey enthusiasts it's possible to become so consumed with AutoHotkey that they forget that there are many features built into Windows

which will do the same thing easier and quicker.

Third Edition Update: As I mentioned in a Chapter Two, Chapter Fifteen, "Tracking the Original Edit Window," of *Beginning AutoHotkey Hotstrings* offers a sure-fire technique for saving the current page, then later returning to it using the WinActive() function and the WinActivate command.

I recently noted one person who was simulating the motion of the mouse with AutoHotkey to prevent the computer from going into the Sleep mode. He then wanted to run a program at a specific time. It seemed to me that it would be easier (and more reliable) to disable the Sleep mode and use Windows Task Scheduler to later launch the program. Sometimes we lose sight of the big picture. The goal is to do things the easiest way—whether it uses AutoHotkey or not.

Chapter Six: Hotkeys to Automate Right-click Menus

“Using the same right-click menu repeatedly? Protect from Carpal Tunnel with this AutoHotkey tip.”

> *If you have a right-click menu that you use over and over again, then you can save time and repetitive finger actions by setting up an AutoHotkey hotkey to get the job done. A tip for using the AppsKey to open the context menu—even if you don't have one on your keyboard. Plus, toggling features on and off.*

This chapter contains useful basic information for automating right-click menus, however the AppsKey technique discussed at the end of the chapter contains a simple change that greatly simplifies this problem. While it may not work in all situations, it works much better for the example used in this chapter.

If you want to save yourself some mouse clicks, a possible contributor to carpal tunnel syndrome, in an often used right-click menu sequence, then you should turn your menu selections into AutoHotkey hotkeys. In many Windows programs, there are already hotkeys assigned for right-click menu options. In those situations, it's often easier to use those hotkey combinations rather than right-click, move down the menu, pop up the submenu, slide down the new menu, then left-click on the desired item. If nothing else, doing this mouse procedure over and over again becomes quite tedious. Especially, if you have to do it a lot. The problem is that some programs don't have hotkeys for selected options.

For example, in many e-mail client programs it can be useful to add certain domains to the Blocked Senders list to stop more junk mail from a specific domain. In the Windows 7 Mail program, this is done by selecting the received junk message, opening the right-click menu, sliding down to "Junk email", waiting for the next menu to open, then left-clicking "Add sender's domain to blocked sender list" (see Figure 1). Then any incoming mail from that domain is automatically moved to the Junk Mail folder. But there is no hotkey for adding the domain to the Blocked Senders list. The option and hotkey combination shown as "Mark as junk" merely moves the e-mail to the junk folder without any type of permanent remedy.

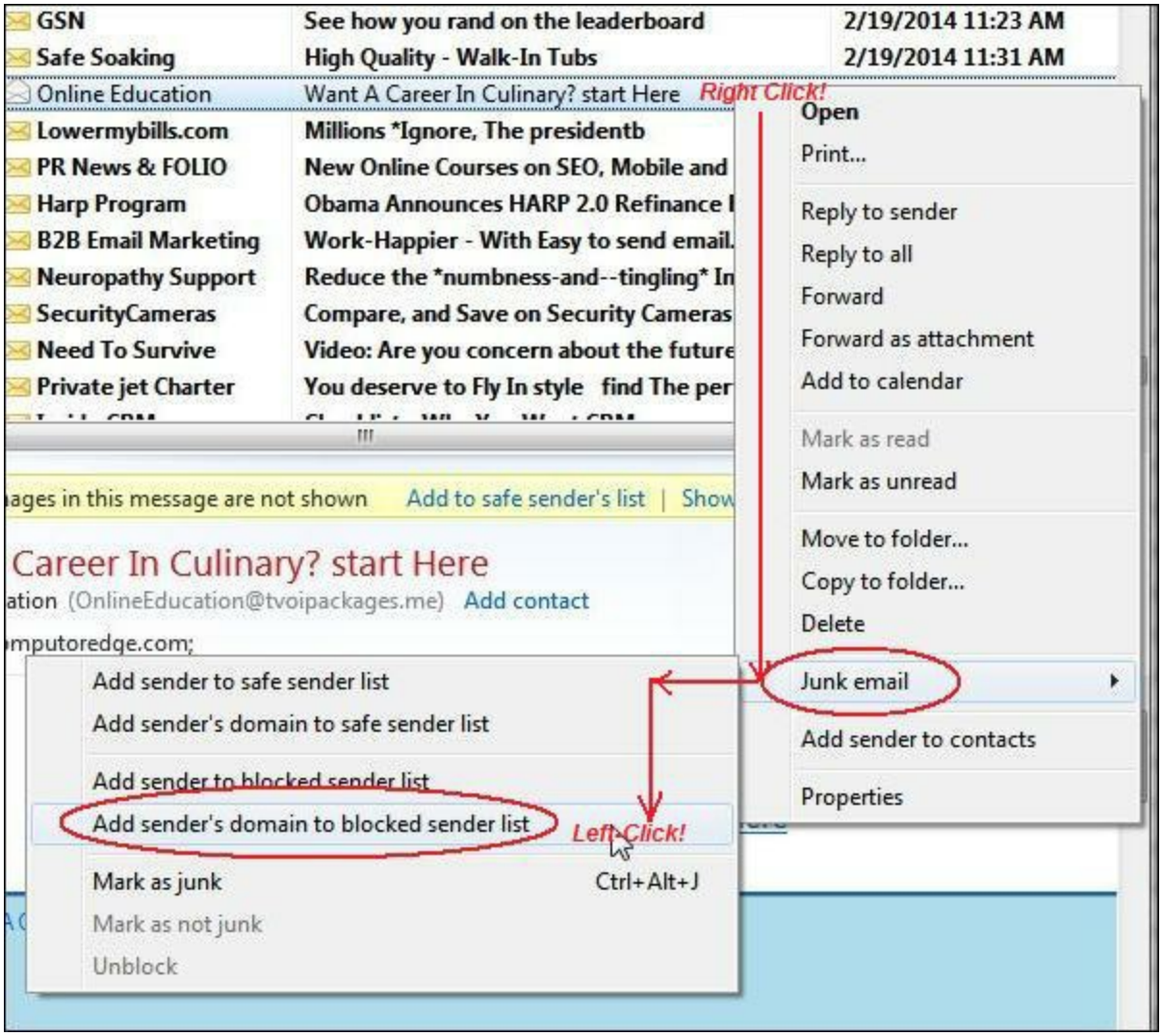


Figure 1. The sender's domain is added to the blocked e-mail list through the right-click menu.

To remedy this situation the following short AutoHotkey script can be used:

```
^!j::
  Click Right
  Sleep 200
  Send, {DOWN 13}
  Sleep 200
  Send, {RIGHT}
  Sleep 200
  Send, {DOWN 3}
  Sleep 200
  Send, {ENTER}
Return
```

When loaded this script will execute the domain blocking for Windows Mail in Windows 7. It is not a universal script because it is tailored to the specific menu in Windows Mail. But that means you can easily change it to conform to any right-click menu in any Windows program.

Hotkeys are set up by adding specific key combinations to an AutoHotkey script followed by a double colon (::), followed by action command lines and terminated with the Return command. This places in memory a routine which will be activated every time the hotkey combination is pressed.

Note: You do need the free AutoHotkey program installed on your computer, but writing a script can be as simple as adding the code to a Notepad file with the .ahk *extension. See "Installing AutoHotkey and Writing Your First Script."*

To activate the action, the target e-mail must be selected in the e-mail list before pressing the CTRL+ALT+J (^+!+J) hotkey combination. (You may note that this is also the hotkey combination assigned to "Mark as junk." You can use any combination you like, but this will override the original combination so the hotkeys will both send the e-mail to the Junk folder and add the domain to the blocked list.)

Use the Click command (*Click Right*) to send a right-click to the mouse cursor location. This opens the right-click menu. But since the *Click* command is used, it's necessary for the cursor to remain within the e-mail list pane when the hotkey combination is executed. Otherwise, the right-click menu will open for wherever the cursor happens to be sitting (the Desktop or other window) with unpredictable consequences. (To prevent random right-click menus from a misplaced hovering mouse, a trick for blocking the routine when the cursor is not over the list of messages appears below.)

The Sleep command (*Sleep 200*) is added merely to pause the script while allowing enough time for the menu to open and become active. Otherwise the script may outrun the menu action and do nothing—or worse the wrong thing. The number of microseconds used can vary based upon what your Windows system needs. Initially, you may want to slow down the script by making the delays longer (*Sleep 1000* for one second) so you can see whether the proper menu items are being selected.

The Send command (*Send, {DOWN 13}*) is used to move the cursor down the menu to "Junk email." The number 13 added to DOWN simulates pushing the down arrow 13 times. It's important to get the number right. Otherwise the wrong option could be activated. That's the reason for slowing it down while testing. Then after the submenu automatically opens, the right arrow key is sent (*Send, {RIGHT}*). Down three more (*Send, {DOWN 3}*), then the ENTER key is sent (*Send, {ENTER}*) to execute the option. Don't use the *Click Left* command because at this point the actual mouse cursor could be sitting almost anywhere.

Preventing Action from a Wandering Mouse

This trick is a little more advanced and you don't need to do it if you're comfortable with keeping the mouse hovering inside the message list pane in Windows Mail. But it does protect against inadvertent right-click menus with wrong selections.

As noted above, the *Click* command will activate at the location of a hovering mouse. As long as the mouse is anywhere over the list of messages, then the *Click Right* command will act on the currently selected message. In order to prevent random right-click menus caused by a mouse wandering over other windows (or panes within Mail), the hotkey routine must be stopped from activating when the mouse is not hovering over the list of messages.

The AutoHotkey download includes a program called AutoIt3 Window Spy. After installation, it can be found in the AutoHotkey programs folder. This program is commonly used to identify specific information about active windows and controls within those windows for use in AutoHotkey scripts. Open Window Spy by selecting it from the Programs list, finding it through the Start Search field (Windows Vista or 7), or right-clicking on the icon in the System Tray for an active AutoHotkey script (*.ahk*) and selecting Window Spy from the menu (see Figure 2).

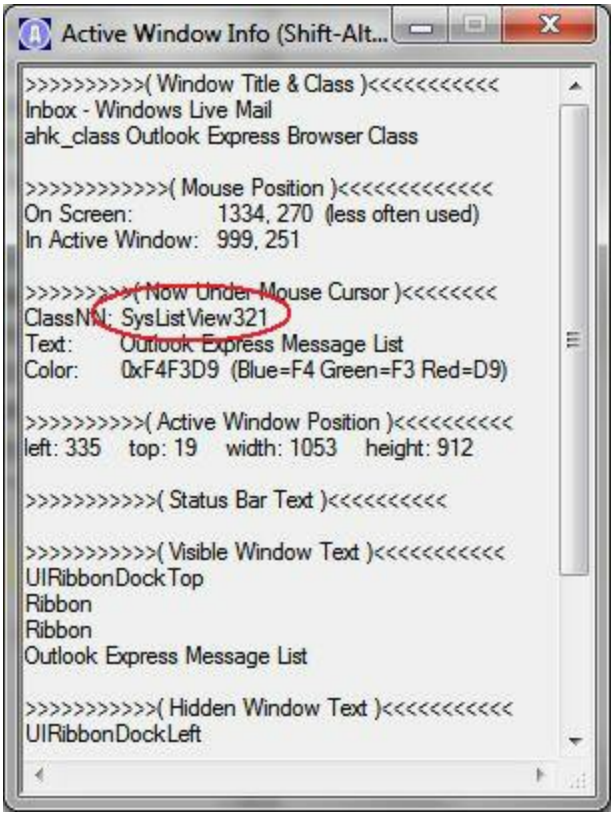


Figure 2. Window Spy shows that the Message List and a control class of SysListView321 when Mail is active and the mouse is hovering over the Inbox list.

Click on a message in Windows Mail (or whatever program you're using) and note the value of ClassNN in the Now Under Mouse Cursor section. This is the control id needed to isolate the routine—in this case *SysListView321*.

To isolate the routine to that control pane the following code is added to the script:

```
^!j::
MouseGetPos,,,, MailWindow
If MailWindow = SysListView321
{
  Click Right
  sleep 200
  Send, {DOWN 13}
  Sleep 200
  Send, {RIGHT}
  Sleep 200
  Send, {DOWN 3}
  Sleep 200
  Send, {ENTER}
}
Return
```

The MouseGetPos command is used to grab the control id for whatever control is under the hovering mouse in the active window and store it in the variable *MailWindow*. Note that there are four commas used before the *MailWindow* variable name. They are all required, otherwise different information will be store in *MailWindow* (see the *MouseGetPos* command linked above).

Next, only if the variable *MailWindow* equals "SysListView321" will the rest of the right-click menu routine run. This prevents accidental firing of the routine when the mouse is in the wrong location—outside the message list

pane.

Warning: If you accidentally add a domain, such as gmail.com, to the blocked domain list, then you won't see any messages from people who use that domain (e.g. Gmail). Periodically check the Blocked Senders list for domains that should not *be blocked.*

You may not need this script for making a hotkey to add domains to the blocked e-mail list, but this same simple technique can be used anywhere in any Windows program where a menu pops up with a right-click. If you find yourself repeating a mouse right-click procedure ad infinitum, then you should set up a hotkey combination with AutoHotkey. It's as simple as that. It's quick and may save you from carpal tunnel syndrome.

A Simpler, More Effective Technique

We've been working on an AutoHotkey script to automate the Windows Mail right-click menu when adding domain names of junk mail to the blocked list. It's part of my grand experiment to determine if I can reduce the volume of Spam that lands (and stays) in my Inbox. I know that I'm probably fighting a losing battle because spammers use a huge number of domain names for their addresses. I know it's not possible to stop them all (spammers can always add more addresses), but maybe I can slow down the flow.

The first step is to make it as easy as possible to add the bad domain names to the blocked list. The normal procedure is to right-click on the Spam listing then navigate to Junk E-mail => Add Sender's Domain to Blocked Senders List (see Figure 1). However, doing this for every piece of junk mail becomes tedious. A hotkey combination set up with an AutoHotkey script is much easier.

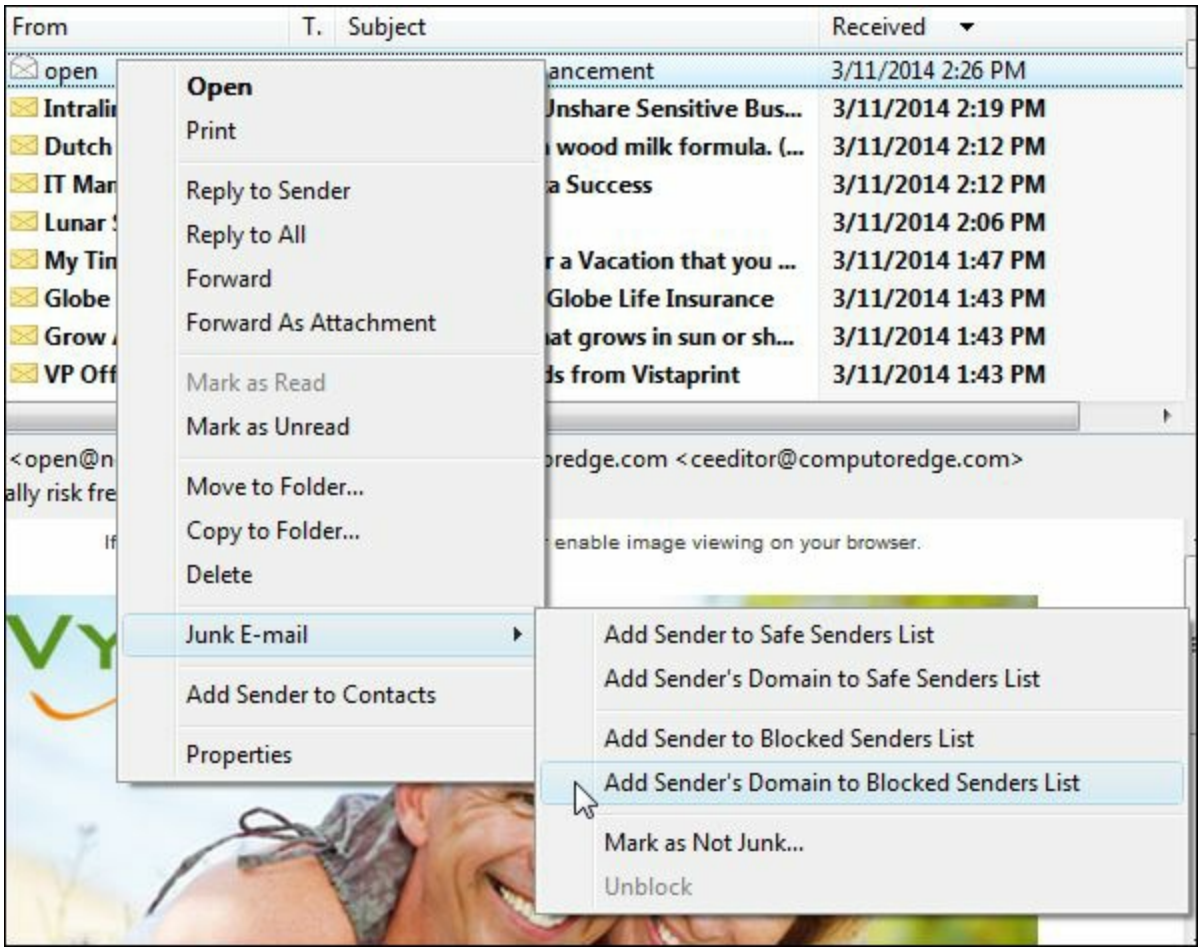


Figure 1. Right-click on the Spam to add the domain to the Blocked Sender's Domain List.

Above I used the Click Right command to open the right-click menu. The problem is that even after the first Spam header was selected, the mouse cursor must continue hovering over the same area of the window to activate the proper right-click menu. I added some code to prevent the wrong menu from opening if the mouse was hovering in the wrong place, but that merely prevented the menu from displaying at all when the mouse was moved away from the right pane. If the mouse wasn't hovering over the list, it still wouldn't work. The slight change to the script discussed next prevents the mouse location from affecting which context menu pops up regardless of the mouse cursor position.

The Windows Context Menu

There is another way to open the right-click menu without using the *Click* command. On many keyboards you'll find a menu key or applications key (pictured at left) which opens the default right-click menu for the active program window or pane within the window—often popping up in the upper-left hand corner of the active pane or over the selected item. This key once served an important purpose when the mouse only had one button. But since a click of the right mouse button on the proper place in the window serves the same purpose, opening the context menu, this key has fallen out of use for most people. Many users don't even know that it exists. On some keyboards it's necessary to push the menu key in combination with another special function key to open the menu—if it exists at all. The advantage to the menu key is that as long as the proper window (and pane inside the window) is active you don't need to use the mouse at all.

It's important to note that since the menu key is not sensitive to the mouse location, it cannot be used to open the context menus which are mouse position sensitive. For example, hovering over a Web link and hitting the menu key will not open the menu that's normally available when right-clicking on the link. Only the standard context menu for the active window pane will appear.

If most people don't use the menu key (and might not even have one on their keyboard), why bring it up at all? Because AutoHotkey has a keyname for the menu key (*AppsKey*) which can be used in conjunction with the *Send* command to achieve the same result. *Send {AppsKey}* can provide all the benefits of the menu key without ever locating or using the key itself.

In the example of blocking junk mail domains Windows Mail, now all that's needed is to click on the target e-mail header of the Spam and use the CTRL+J (^j) hotkey combination:

```
^j::
  SendInput {AppsKey}
  Sleep 100
  Send, {Down 12}
  Sleep 100
  Send, {Right}
  Sleep 100
  Send, {Down 3}
  Sleep 100
  Send, {Enter}
Return
```

Now it doesn't matter where the mouse is located as long as the proper window pane is active.

Note: Don't use the ALT key (!) in your hotkey combination since the ALT key prevents the menu key from activating.

This script can be adapted to other context menus in other programs by adjusting the *Down* and *Right* parameters to the appropriate number for the new menus. It's a good idea to test the script while watching it with longer Sleep periods (say 1000 microseconds or more). This slow-motion observation of the action ensures that the proper selections are enacted by the script. Plus, you might also remove (place a semicolon in front of the line of code) the *Send, {Enter}*command, which actually executes the selection, until after you know you have it right.

If you always left-click in the proper window pane before using the hotkey combination, then you can stop reading right here before I add any further complications to the code. The problem with writing any AutoHotkey scripts is that you can always add more features. But, often after you get the first 90% done, it's pretty much everything that you need. If you're new to AutoHotkey then you might work with the code above for a while before moving on to the following section. While the next part of this chapter does address a real issue, in most situations it's not likely to be a problem.

Caution: If you're using a script similar to the one above to block domains, check each message to make sure that the message isn't coming from common domains such as gmail.com *or* yahoo.com*. This could cause e-mail from many of your friends who use one of those services to be blocked. If you do get Spam from a common domain, block the sender rather than the sender's domain.*

Limit the Hotkey to the Proper Program

If the wrong program window is active, then the hotkey menu results are unpredictable. Another context menu may be activated with disastrous actions following. There are a couple of ways to ensure that the hotkey combination is only available when the right window is active. The first is the #IfWinActive directive. By placing *#IfWinActive* with the title of the window at the end of the script followed by the hotkey definition, the hotkey combination will only be in effect with the right type of active window. However, this does not prevent the wrong context menu from popping up if the wrong window pane (control) is active in that window. For example, Windows Mail has one pane with the various mail boxes, another with the list of e-mails received (this is the one we want), and the message view pane itself. Click in the wrong place and the wrong menu will respond to this script.

This last problem is resolved as it was in the earlier script by isolating the action to the correct pane, the e-mail header list. But this time we use the ControlGetFocus command rather than the MouseGetPos command. This also eliminates dependence on the mouse location:

```
^j::
ControlGetFocus, ActiveControl, A
If ActiveControl = SysListView321
  {
    SendInput {AppsKey}
    Sleep 100
    Send, {Down 12}
    Sleep 100
    Send, {Right}
    Sleep 100
    Send, {Down 3}
    Sleep 100
    Send, {Enter}
  }
Return
```

The *ControlGetFocus* command saves the name of the active control (the e-mail list) to the variable *ActiveControl*. If it doesn't match the Windows Mail heading list (*SysListView321*), then it does nothing.

In truth you should be checking the domain of each e-mail before executing the block. Otherwise you might inadvertently block the wrong person. But, if you adapt this script to the context menu of another program for a

totally different purpose, this may not be an issue.

* * *

The following is more advanced information for the AutoHotkey curious.

Finding Window Titles and Control Names with WindowProbe

Earlier in this chapter, Window Spy was used to find titles and control names. I put together another quick tool I call WindowProbe for capturing some of the same information in a different format. WindowProbe uses a tooltip help pop-up to display vital information as you move the mouse cursor around the screen (see Figure 3).

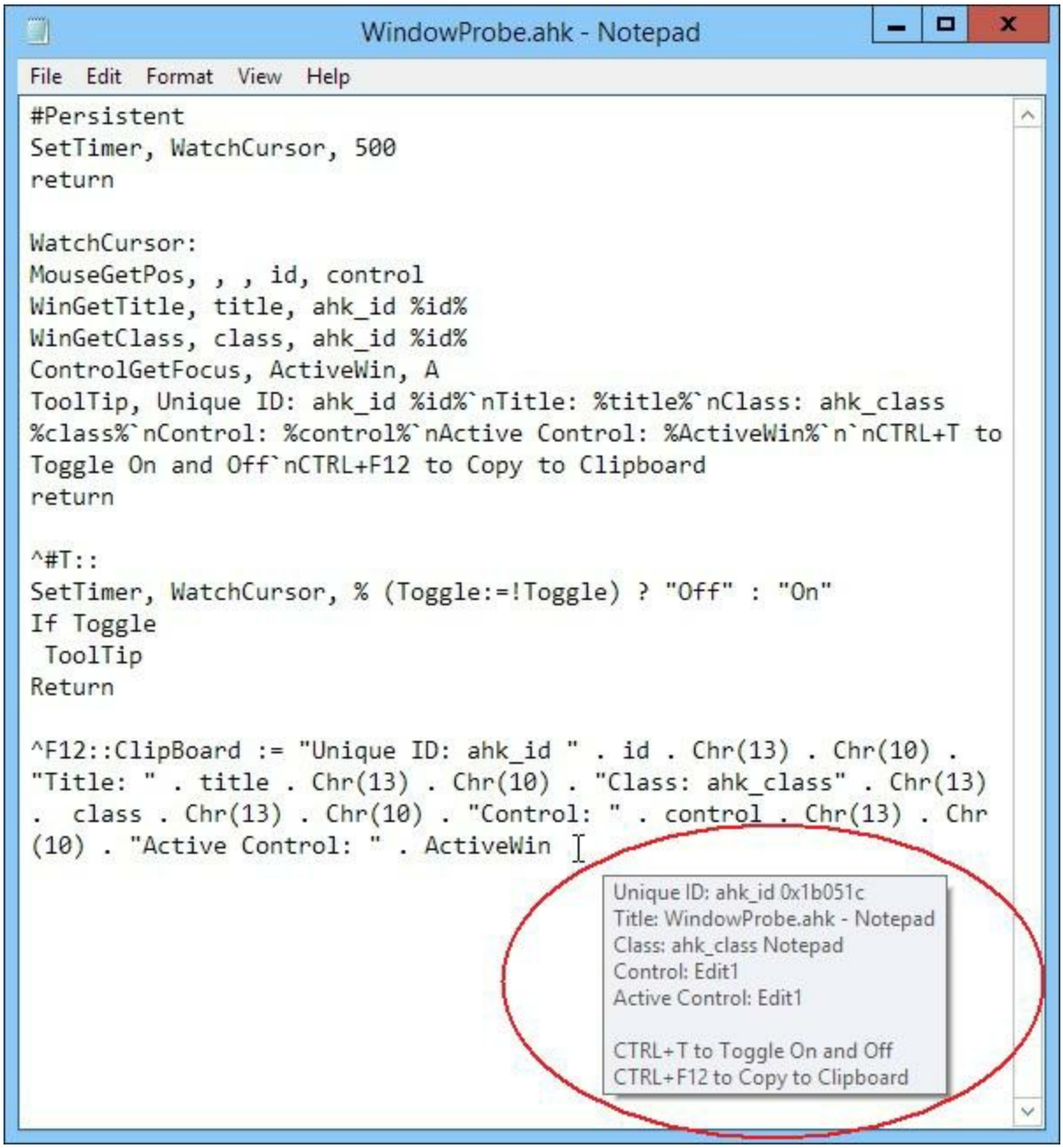


Figure 3. The WindowProbe app is an AutoHotkey script which displays the Unique ID, Title, Class, and Control under the mouse cursor, plus the name of the Active Control in a cursor tooltip message window.

A while back I highlighted a similar script (Chapter Thirty-four of the AutoHotkey Applications e-book) which provided most of the same window information. WindowProbe is the same script with a few modifications. Once loaded, *WindowProbe.ahk* adds a tooltip window to the mouse cursor. As the mouse moves around the screen, the tooltip window displays information from the window and pane just under the cursor: the unique window ID, the window title, the window class, the control name, and the active control name. With the exception of the active control, all the other values change as the cursor moves from window to window. The active control since it is the active field or pane in the active window only changes when a new control is clicked.

The hotkey combination CRL+⊞+T (^#T) toggles the tooltip message on and off. Since you may want to capture this information for use in an AutoHotkey script, the CTRL+F12 key combination saves the codes in the Windows Clipboard. After capturing in the Clipboard, it can be pasted into any text document or file in this format:

```
Unique ID: ahk_id 0x1b051c
Title: WindowProbe.ahk - Notepad
Class: ahk_class Notepad
Control: Edit1
Active Control: Edit1
```

Copy and paste the bits that you need into your script.

The primary additions to the original script are the *GetControlFocus* command to determine the active control, the hotkey combination to toggle the tooltip window on and off (CTRL+⊞+T), and the hotkey combination to save the information to the Windows Clipboard (CTRL+F12). Of particular interest is the toggling technique which can be used to turn virtually anything in an AutoHotkey script on or off. Even though it's not really a beginner's trick, it's worth exploring.

Toggling On and Off

Often there are times when you want to turn a feature on or off without exiting the app. In this case leaving the tooltip in the WindowProbe app turned on can get pretty annoying. The tooltip window (which blinks every ½ second) relentlessly follows the mouse cursor around the screen like a tag-along little brother. It can drive you crazy. This is a job for the toggle.

A toggle is code that switches state every time it runs. It could be from on to off, true to false, or 0 to 1 and vice versa. This is a common programming problem with a variety of solutions. Often, a value is set to 0 for off or false and 1 for on or true, then flipped every time the control state changes. In this case AutoHotkey operators are used to toggle the value of a variable which happens to be called *Toggle*. (It could be any other variable name.):

```
SetTimer, WatchCursor, % (Toggle:=!Toggle) ? "Off" : "On"
```

The exclamation point (*!*) is the logical *NOT*. When *Toggle* (which initially has no value and is therefore false) is set equal (:=) to *NOT* (!) *Toggle*, the value of *Toggle* is flipped from false to true. The next time the code is encountered *Toggle* will flip from true to false—thereafter, switching again every time the snippet runs.

But there are two other aspects of note in this same line of code. The first is the single percent sign (*%*) which forces the evaluation of an expression. Adding the force expression sign (*%*) tells AutoHotkey to evaluate the expression which comes after the *%* sign—in this case is the ternary operator (?:) (the second item of interest). There are numerous AutoHotkey commands where a *forced expression* can return a new setting directly to the command.

The ternary conditional operator is a shorthand form of the *If-Then-Else* statement. It evaluates whatever appears before the question mark (*?*). If true, it uses the first result after the question mark, but before the colon (:). Otherwise, the ternary returns the result after the colon (:). In the line above, if *Toggle* evaluates true, *SetTimer* is

set to *off*. If *Toggle* evaluates false, *SetTimer* is set to *on*. This effectively creates a switch turning the tooltip on or off on each use of the CTRL+  +T hotkey combination. The power of the ternary conditional operator is that it can be placed directly in many of the AutoHotkey commands to change conditional settings without adding multiple lines of code. (If you're running AutoHotkey basic, you may need to place a space before and after the question mark (?) in the ternary operator.)

The next two lines of code:

```
If Toggle
 ToolTip
```

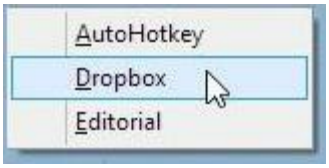
merely turn of the tooltip display when *Toggle* is true (*off*).

Find the *WindowProbe.ahk* script and the app *WindowProbe.exe* (the EXE file runs on any Windows computer without AutoHotkey) at the *ComputorEdge* AutoHotkey Download site in the *WindowProbe.zip* file.

Chapter Seven: Quickly Open Favorite Folders

"If you do nothing else, use AutoHotkey to quickly open your most used folders in Windows Explorer."

This is a cool trick. Do you find that you are opening the same Windows folders innumerable times during the day? Set up a hotkey combination with AutoHotkey to instantly open each of those most used favorites. Plus, you can put those key actions in a quick pop-up menu.



If you have key folders that you're continuously opening in Windows Explorer, then you're going to love this AutoHotkey trick. Rather than open Windows Explorer (+E) and navigate to the right location (or click the folder in your Favorites) every time you need to access an important folder, you can set up a hotkey combination which will instantly open it. All it takes is one line of code. But even better, with only a couple more lines of code, you can add a pop up a menu that does exactly the same thing (see image at left). If you can open Notepad (and have the free AutoHotkey software installed), then in a matter of minutes you can add a cool feature to your Windows computer which will astound your cohorts and friends.

Open Folders Instantly

The first step is to set up a hotkey combination for one of your most used folders. Open Windows Explorer (+E) and navigate to the target folder (see Figure 1). (Windows Explorer is called File Explorer in Windows 8.) Copy the exact path to the folder or use in the AutoHotkey script.

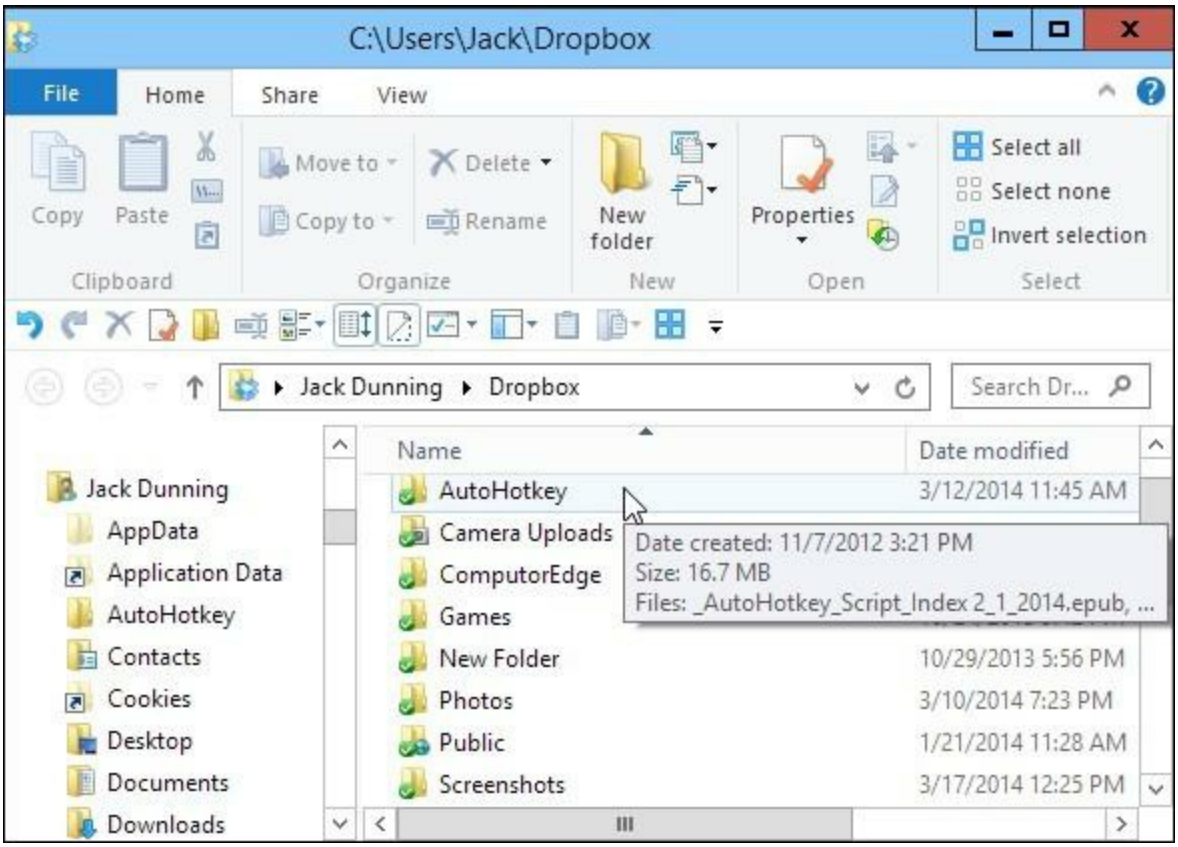


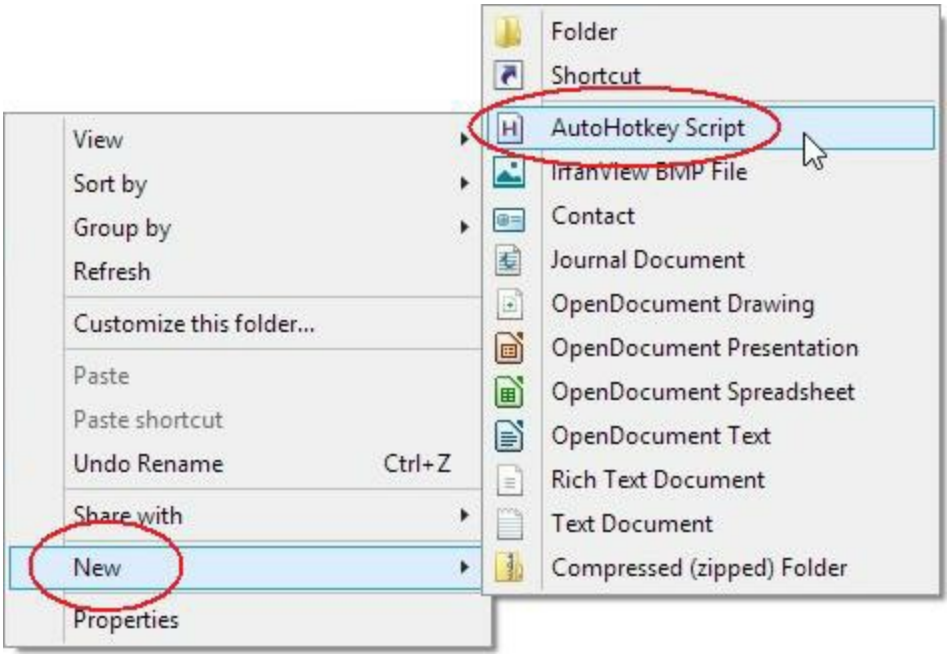
Figure 1. My Dropbox folder is open in File Explorer in Windows 8 (called Windows Explorer in all other versions of Windows).

To copy the folder path, click the little folder icon on the left side of the folder path field (see Figure 2). The field will

convert to the proper Windows format and become highlighted for copying. Use the Copy command (CTRL+C) to save the path to the Windows Clipboard.

Figure 2. Click the folder icon on the left side of the path field to highlight the correct syntax for the folder path. Use CTRL+C to copy.

In Windows XP, the folder path will already be in the proper format. Just select and copy. When used in conjunction with the AutoHotkey Run command, this copied path automatically opens the target folder in Windows Explorer.



Once you have installed AutoHotkey, you can right-click in your script folder (or for that matter any folder) and select New => AutoHotkey Script (see image at left). A new AutoHotkey file will be created with the AHK extension. Give it any name you like—possibly, *OpenFolders.ahk*.

Tip: When the AutoHotkey software is installed on your Windows computer, it adds a number of options to your Windows Explorer right-click context menu. In addition to the New => AutoHotkey Script option mentioned above, whenever you right-click on an

AutoHotkey script file name (AHK extension) options for Run Script, Compile Script, and Edit Script are displayed. These options make it much easier to work with scripts. Editing, testing, and compiling are always just one right-click away.

Right-click on the new AutoHotkey script filename and select Edit Script. Your default text editor will open the file for editing. (In many cases, the default text editor will be Notepad. If another editor or word processor opens, be sure that you always save the file as text with the AHK extension.) Add the following line of code to the file:

```
!2:: Run, C:\Users\Jack\Dropbox
```

where you substitute your copied path for *C:\Users\Jack\Dropbox*. (Type *Run,* then use CTRL+V to paste the new folder path.) Save the file, then right-click and select Run Script which loads the script into memory. That's all there is to it!

Now, every time you use the hotkey combination ALT+2, the target folder will open (or activate if it's already open).

There are certain characters that represent activating hotkeys in AutoHotkey. In this example, the exclamation point (!) represents the ALT key. For CTRL use ^ and for the  key use #. These key symbols can be found in the AutoHotkey online documentation for Hotkeys. When these characters (or a combination of them) are used with another key, it creates a hotkey which activates when they are pressed simultaneously.

The AutoHotkey Run command can be used to open folders, open Web pages with your default Web browser, or load programs.

Add a new line of code with a new hotkey combination for each folder:

```
!1:: Run, C:\Users\Jack\AutoHotkey
!2:: Run, C:\Users\Jack\Dropbox
!3:: Run, \\JACKSLAPTOP\editorial
```

These lines add two more folders with new hotkey combinations (ALT+1 and ALT+3). Note that the third option is accessing a shared folder on my laptop computer through the network.

If all you want are hotkeys that open your most used folders, then, after adding a line of code for each folder, you can stop right here. But if you have trouble remember the hotkeys, or just want to get a little fancier, then you can add all the hotkeys to a pop-up menu.

Open Folders with a Pop-up Menu

There are only a couple of lines of code required to turn your hotkeys into a pop-up menu. First the Menu command is used to add an option to a menu called *Folders*:

```
Menu, Folders, Add, Dropbox,!2
```

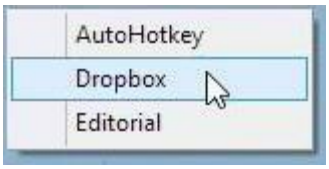
The menu will display *Dropbox* as the item name and the now familiar *!2* (ALT+2) is the hotkey action that clicking the menu item will activate. To display the menu, another hotkey combination is created:

```
!x::Menu, Folders, Show
```

which uses the *Menu* command to *Show* the menu called *Folders*. The combination *!x* (ALT+X) was used because the keys are close together on the keyboard for easy access.

Add another line of code for each menu item:

```
Menu, Folders, Add, AutoHotkey,!1
Menu, Folders, Add, Dropbox,!2
Menu, Folders, Add, Editorial,!3
```



Save the file and Reload the script. (Right-click on the AutoHotkey icon in the System Tray and select Reload This Script.) Now whenever you use ALT+X, the menu will pop up at the mouse cursor's location (see image at left). Select the folder you want to open.

If this script does what you want, stop here! But what if you don't want to use the mouse at all?

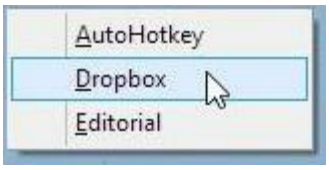
Adding Hotkeys to the Menu

Adding a hotkey to the menu is merely a matter of putting an ampersand (&) in front of the letter you want to use as the activating key. Here is the entire script with the menu hotkeys added:

```
Menu, Folders, Add, &AutoHotkey,!1
Menu, Folders, Add, &Dropbox,!2
Menu, Folders, Add, &Editorial,!3

!1:: Run, C:\Users\Jack\AutoHotkey
!2:: Run, C:\Users\Jack\Dropbox
!3:: Run, \\JACKSLAPTOP\editorial

!x::Menu, Folders, Show
```



When the ALT+X menu is opened, the letter "a" now activates *AutoHotkey*, the letter "d" activates *Dropbox*, and the letter "e" activates *Editorial*. No mouse is needed. The menu will show those letters underlined, as shown at left.

Important note: While the hotkey combination lines of code (*!1*, *!2*, *!3*, and *!x* followed by the double colon *::*) can appear in any order, the three *Menu, Folders, Add* lines must appear first in the script before any of the double colon lines. These *Menu* lines need to load immediately. If any of the hotkey setup lines are encountered before the *Menu, ..., Add* lines, the *Menu, ..., Add* lines will not load.

If this is enough, stop here! But, what if you want to only use the mouse without any hotkeys?

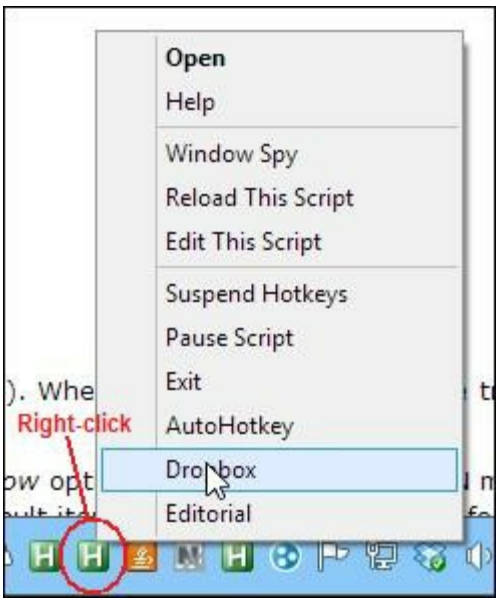
Adding the Menu to the System Tray Right-Click Menu

If you prefer to do everything with clicks of the mouse, then with minor modifications you can add the menu to the right-click menu of the System Tray icon. Then all you need to do is right-click on the running AutoHotkey icon, then left-clicking on the target menu item (see image at left).

The only required change to the script is replacing all occurrences of *Folders* with the word *Tray*. Then the menu will be added to the bottom of the System Tray AutoHotkey icon right-click menu. All of the hotkey combinations will continue to work, but now you will be able to do it all with only the mouse.

Note: If you do activate the menu with ALT+X (menu pops up next to the mouse cursor, not in the System Tray), since it is using the same letter as a menu hotkey, the standard menu item "Edit This Script" will interfere with selecting the Editorial folder by pressing the "e" key. You may want to switch the menu hotkey to a non-interfering letter such as *Edi&torial* (activates the letter "t" as a menu hotkey.)

Here is the new script with all the pertinent changes:



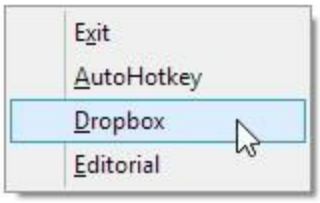

```
Menu, Tray, Add, &AutoHotkey,!1
Menu, Tray, Add, &Dropbox,!2
Menu, Tray, Add, Edi&torial,!3

!1:: Run, G:\Users\Jack\AutoHotkey
!2:: Run, C:\Users\Jack\Dropbox
!3:: Run, \\JACKSLAPTOP\editorial

!x::Menu, Tray, Show
```

If you don't mind all the extra menu items from the standard System Tray right-click menu, stop here! However, the main problem with the script now is that it's too busy with all of the standard items.

Shortening the Menu



There are a couple of ways to approach removing items from the System Tray menu, but you want to be careful. If you add the line:

```
Menu, Tray, NoStandard
```

all the extra items will be removed, but you will also have no way to stop the script (*Exit*) other than using Windows Task Manager to end the process or modifying the script and reloading it from Windows Explorer. You should at least add *Exit* back to the menu (shown at left) using a label (subroutine) to exit the app:

```
Menu, Tray, NoStandard
Menu, Tray, Add, E&xit,Exit
Menu, Tray, Add, &AutoHotkey,!1
Menu, Tray, Add, &Dropbox,!2
Menu, Tray, Add, &Editorial,!3

!1:: Run, G:\Users\Jack\AutoHotkey
!2:: Run, C:\Users\Jack\Dropbox
!3:: Run, \\JACKSLAPTOP\editorial

!x::Menu, Tray, Show

Exit:
 ExitApp
Return
```

The *Exit* menu item calls the label *Exit:* which issues the ExitApp command and closes the program.

This last script is not particularly long. It can be easily modified to open Web pages and run programs by adding a new hotkey combination code lines using the *Run* command and another *Menu, ..., Add* code line to match each new operation hotkey. While there is so much more that AutoHotkey can do, you don't need to go any further than the first step in this column. But finding a quicker way to open your most used folders is something that you really should do.

Third Edition Note: For a more extensive personal menu structure, see the QuickLinks.ahk script for speeding up your usual activities,

Chapter Eight: Using Extra Mouse Buttons and the Wasted Insert Key

"If you do nothing else, use AutoHotkey to power extra mouse buttons and vestigial keys like insert."

Tips on how use AutoHotkey to take advantage of extra mouse buttons, and the annoying Insert and CapsLock keys.

For me a mouse is just a mouse. I use it to move the cursor around the screen, left-click to select, and right-click for the context menu. On most of my computers I had a standard three-button mouse: left button, right button, and the clickable scroll wheel in the center. That's all I really needed…at least until I started working with AutoHotkey.

On my Windows 8 (now Windows 10) computer, I have a five-button mouse. I didn't plan this. The mouse came with the extra buttons. I've always looked upon the two extra mouse buttons (one on the left side and one on the right) as benign extras like your navel or appendix. However, I would occasionally hit the fourth button on the left side with my thumb accidentally causing a Web page to jump Back. What the heck?

If I hadn't realize what just happened, I could have lost whatever I had been working on. (Tip: If you're working on a Blog and you accidentally go Back or Forward, you can most likely reverse the process and get to your unsaved typing by doing the antithetical (opposite) operation. Right-click and select either Forward or Back as appropriate. If you accidentally close the window (F4), I'm afraid you're out of luck.) I decided it was time to disable these annoying extra mouse button features by putting them to good use with AutoHotkey. (I know that I could have used the mouse drivers that came with the keyboard mouse set, but I find the AutoHotkey solution more flexible.)

Make Use of Extra Buttons on Your Mouse with AutoHotkey

One of the great things about AutoHotkey is that you can make better use of extra mouse buttons and normally (or nearly) useless keys such as the Insert and CapsLock keys. I decided to make use of the fourth mouse button on the left side which has easy thumb access. (If you're left-handed, then the fifth button would have easy thumb access.) It's a bit more natural than trying to use the fifth button on the right side with one of my other fingers. My goal was to eliminate the need to use any hotkey combination when activating the pop-up menu for my QuickLinks AutoHotkey app, (see Figure 1). (The QuickLinks script is discussed in the e-book *Digging Deeper Into AutoHotkey*.)

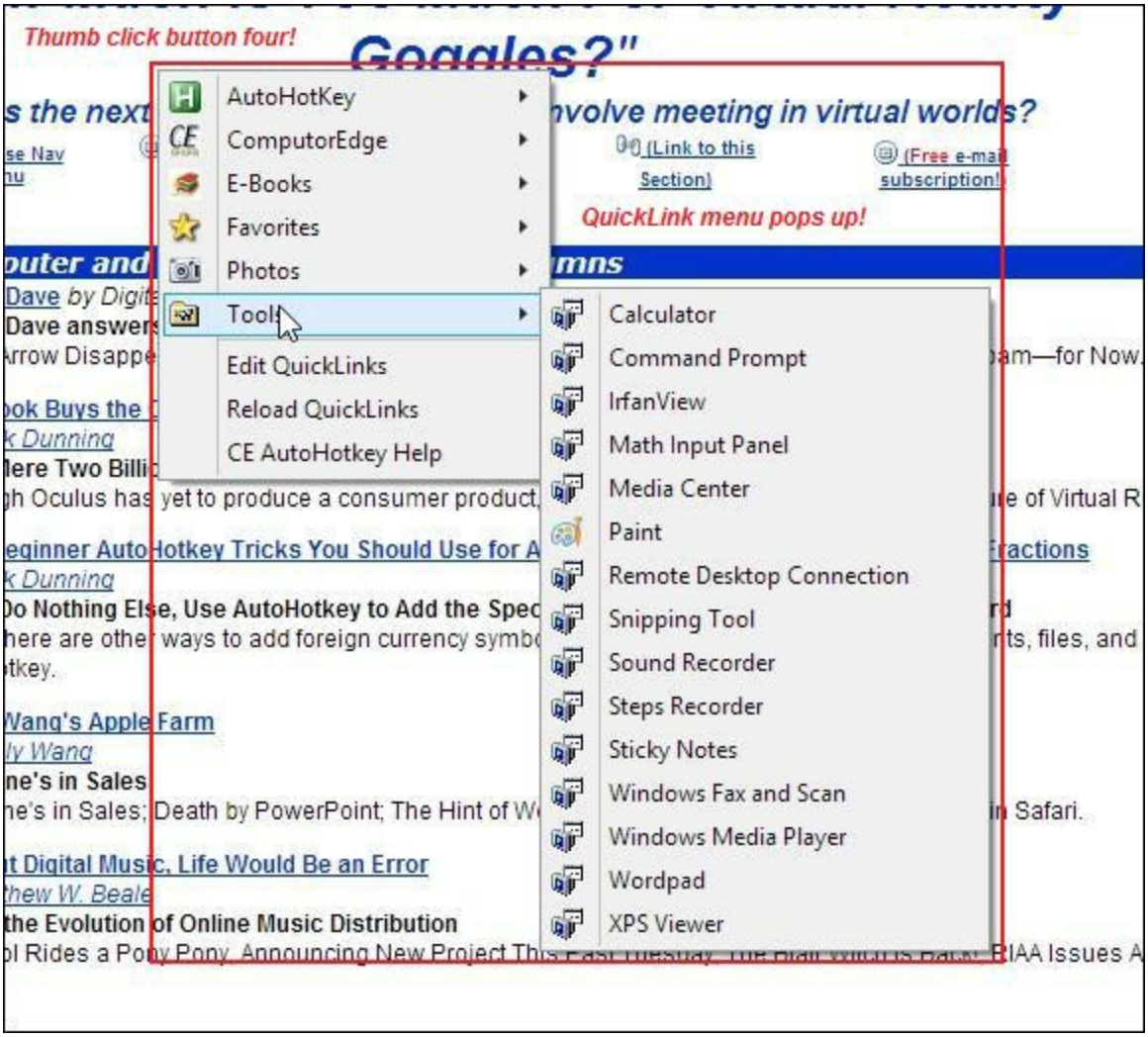


Figure 1. Using the extra mouse button gives me an alternative way to open QuickLinks without using the hotkeys.

In my personal combined AutoHotkey script which loads automatically on bootup, I have two hotkey combinations that activate the QuickLinks menu: +Z on the left side of the keyboard and ALT+, (hold down the ALT key and press the comma) on the right side. I wanted to write a one-line script that would activate the QuickLinks menu with a click of that extra fourth mouse button. I thought that I could use either hotkey, but I ended up using the *ALT,* combination because whenever I tried to use the key, it just wouldn't work properly. I'm not sure why. (There are so many key combinations in Windows 8 and 10 that it's probably best to avoid using in AutoHotkey hotkeys anyway.) All I needed was one line of code to make it work:

```
XButton1::SendInput, !,
```

The term for the fourth button is *XButton1* with *XButton2* designating the fifth mouse button. If you have more than five mouse buttons, then you will need to consult the AutoHotkey documentation's Special Keys section on the KeyList page to uncover how to reach them for reassignment.

If I'm adding the capability directly to the QuickLinks script, then I can put the Menu command for popping up the menu directly into the script:

```
XButton1::Menu, QuickLinks, Show
```

For good measure, I assigned opening the main *ComputorEdge* Web page to the fifth mouse button:

```
XButton2::Run, http://www.computoredge.com
```

Now when I click the extra button on the right side, the *ComputorEdge* Web site opens in the default browser. Note that you can use this technique to activate any AutoHotkey command. If you want to run a longer routine, start the code on the next line after the *XButton2::* and enclose it with a *Return* on the last line.

Making Better Use of the Insert and CapsLock Keys

Most people have no use for the Insert key. The only time it's noticed is when it is accidentally pressed causing volumes of characters to be inadvertently overwritten. Very annoying! This can be prevented by putting the key to good use with AutoHotkey. In the example here, the Insert key is turned into a hotkey which opens the Documents folder in Windows Explorer (File Explorer for Windows 8 users):

```
Insert::Run, %A_MyDocuments%
```

Add that one line to the end of any AutoHotkey script (or run it in a standalone script) and you're done. Now whenever you press the Insert key, Windows Explorer will open located at your Documents folder. If you want it to be a different folder, substitute the full folder path including the drive letter (e.g. C:\) for *%A_MyDocuments%*.

Suppose you want the CapsLock key to open the Command Prompt:

```
CapsLock::Run, cmd
```

or the Windows Calculator::

```
CapsLock::Run, calc
```

The only problem with this approach is that the CapsLock key is unavailable for when you really need it. To resolve this occasional problem add another line of code with a modifier, such as the Shift key, and the CapsLock key:

```
+CapsLock::Capslock
```

Now when you use SHIFT+CAPSLOCK, capitalization will toggle on and off. It's there when you need it, but won't accidentally turn on caps when you inadvertently hit it. However, that slip on the CapsLock key will still activate your AutoHotkey command.

There are a huge variety of things that you can do with extra buttons or unneeded keys, but if all you want to do is disable a button or key, merely add one line similar to this:

```
XButton1::
```

You will never accidentally go Back again by hitting that fourth mouse button with your thumb.

Chapter Nine: A Beginner's Guide to Stealing AutoHotkey Apps, Plus Writing and Running Scripts

“There are hundreds of free AutoHotkey apps available for the taking.”

You don't need to know very much about AutoHotkey to take advantage of the many free scripts available for a wide variety of uses on your Windows computer. Here's how to do it. Plus, four scripts you may want to add to your bag of Windows tricks.

There are literally hundreds (if not thousands) of AutoHotkey scripts available for Windows computers. Their purpose and range are diverse and go far beyond those listed on the *ComputorEdge* AutoHotkey Apps Web page. A plethora of scripts can be found at the AutoHotkey Script Showcase and the AutoHotkey Scripts forum, plus a Web search returns many other pages loaded with AutoHotkey apps. The best thing about all of these scripts is that as long as they offer the source code, you can steal them.

The beauty of using these freely available AutoHotkey scripts is that you don't need to know very much (if anything) about AutoHotkey to use them. You can generally copy them into a new file and run them without alteration. In fact, many people (including myself) use all or part of these listings to build their own apps. You may not understand exactly how the script does what it does, yet you can still do minor tailoring to suit your purposes—especially assigning hotkey combinations which may work better for you.

In the vast majority of cases it's not really stealing. The people who have written these apps are offering them free. Otherwise, they wouldn't post the source code (text files with the *.ahk* extension) which can be compiled by anyone to run on any Windows computer. In fact, I prefer to get the source code since I then know what I'm compiling. Even I'm concerned about running any EXE file without knowing the source—which is why every AutoHotkey executable file on the *ComputorEdge* AutoHotkey Dropbox download site has been compiled by me.

If you want to access the multitude of free AutoHotkey apps available, then there are a few steps you need to take to open up this world of Windows tools and utilities. That's why I've put together this "Beginner's Guide To Stealing AutoHotkey Apps."

Install AutoHotkey

The first step is to install AutoHotkey on one of your Windows computers. This will give you all the tools you need to turn AutoHotkey scripts into running apps. You only need the main AutoHotkey program installed on one computer since once you have an AHK source file, you can compile it into an EXE file which will run on any Windows computer without any additional installation.

For AutoHotkey download and installation instructions see "Installing AutoHotkey and Writing Your First Script."

Stealing an AutoHotkey Script

Once you have AutoHotkey installed you're ready to start adding apps to your Windows computer. Search the pages and forums mentioned above looking for the apps that suit you. Most will give a short description of each app and a code listing either in plain text or as an AutoHotkey script file with the *.ahk* extension (e.g. *GooglePhraseFix.ahk*). An AutoHotkey script file is simply a text file with a listing of code. Since all the code is in plain text and readable, it is safe to copy the code or download the file. Even if someone tried to add something sinister to an AutoHotkey script, it would not be processed until you ran it as an AutoHotkey or compiled file. Merely downloading a script file is safe enough.

Note: While it is possible for someone to add AutoHotkey code to a script that might mess with your computer (e.g. the RegWrite command or the RegDelete command could be used to alter your Windows Registry), it's highly unlikely that any scripts found on one of the AutoHotkey sites will contain any such issues. They have been downloaded and the code has been reviewed by numerous people who are well versed in AutoHotkey. The site moderators would take down malevolent scripts spotted. In any case, make a quick scan of the code. If you see any commands that look a little unusual in the code listing, investigate it on the Web. That's what I do when I modify and compile the scripts I copy from other sources. I wish that I could say that all the AutoHotkey scripts are 100% safe, but the truth is that AutoHotkey is a powerful language which in the wrong hands could do some damage. That's why, unless you get your code from a trusted source, it's best to learn how AutoHotkey works for yourself and validate all your own scripts.

You will want to create a separate folder to hold all of your AutoHotkey scripts. When you have opened that folder in Windows Explorer (+E), you will then be able to create a new AHK script by right-clicking on an empty space and selecting New => AutoHotkey Script—which has been added to the context menu by the AutoHotkey installation. This creates an AHK file with some boilerplate included. (Boilerplate is default text added to every new AHK file when create through the context menu.) Once you have the new file with the *.ahk* extension, you will find additional right-click options for the filename: Run Script, Compile Script, and Edit Script (see Figure 1). These options are all you need to turn your AutoHotkey scripts into running apps.

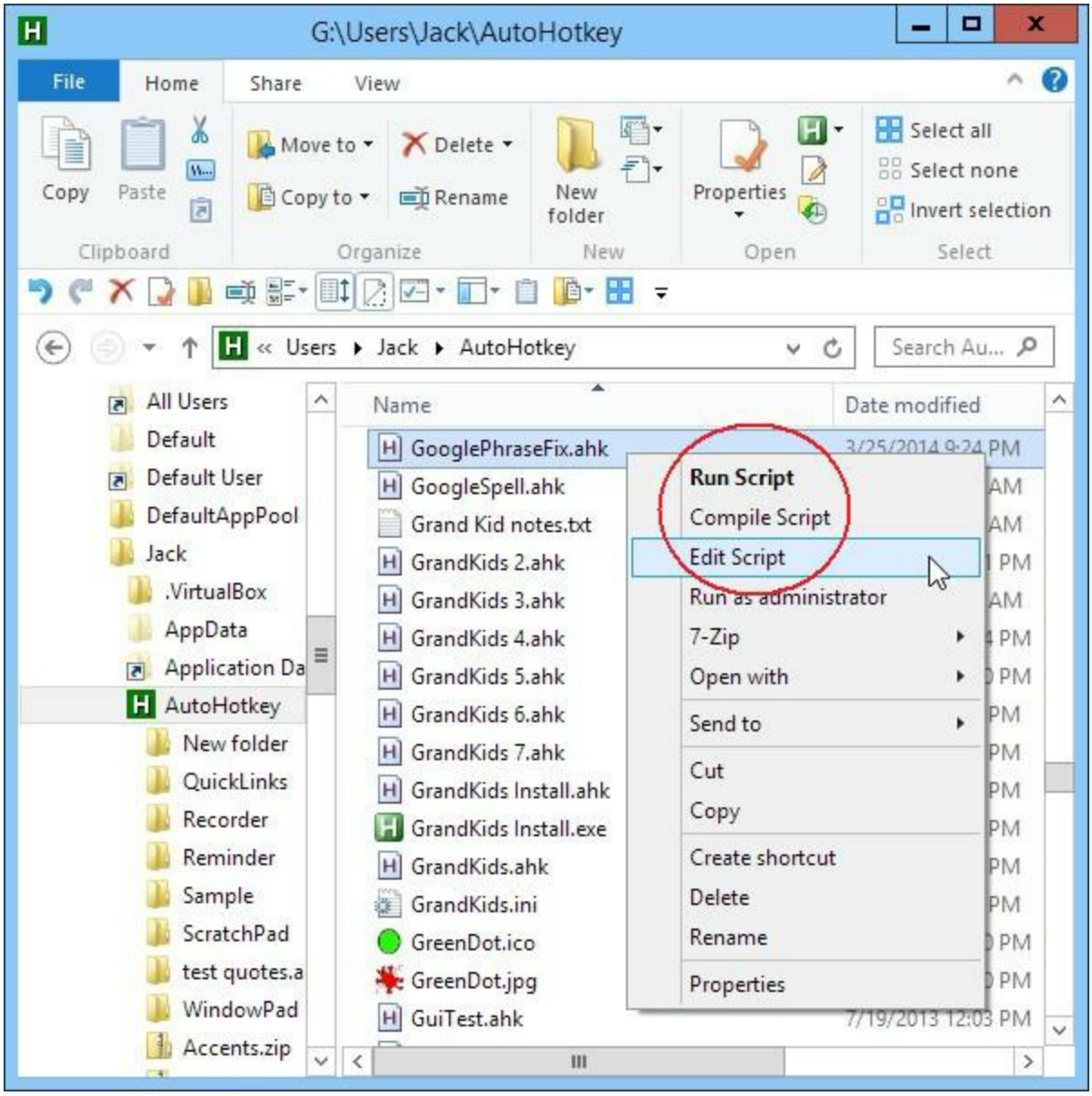


Figure 1. AutoHotkey adds three additional options to the right-click context menu for AHK files: Run Script, Compile Script, and Edit Script.

Edit Script

I recommend that you assign a name to any newly created AHK file which denotes what it does. Then, open the file with a right-click on the file name selecting Edit Script. The file will open in the default text editor—usually Notepad. This is where you paste (place cursor below the boilerplate in the new file, then CTRL+V) the lines of code copied (select all the lines of code and CTRL+C) from the target app at the AutoHotkey script site—CTRL+S to save the file.

Run Script

Once the new script is saved, running it is as simple as right-clicking on the AHK file name in Windows Explorer and selecting Run Script. A little green icon containing the letter "H" will appear in the System Tray and display the file name when the mouse cursor hovers over it. (Some scripts may change the look of the System Tray icon and its Tooltip label.)

You should now be able to test the new AutoHotkey app. After you run it through its paces, you normally will be

able to end it by right-clicking on the System Tray icon and selecting Exit. (Worse case, when the option has been deactivated by the script, open Windows Task Manager (CTRL+SHIFT+ESC) to end the running AutoHotkey process.)

Compile Scripts

Once you know that the app is running the way you want it to run, you can compile it into an executable file (EXE) which will run on any of your (or your friends') Windows computers. The great thing about compiling AHK script files into this type of command file is that it makes them completely portable. You can copy the file to any Windows computer (or keep it on a thumb drive) and run it with a simple double-click—no installation required.

You have stolen the app! Want to automatically load it when you log into your computer?

Run Stolen AutoHotkey Apps at Startup

If you like an app, then you probably want to launch it automatically every time the computer boots up. You do this by adding a shortcut to the EXE file (or AHK file if AutoHotkey is installed on the computer) in the Windows Startup folder. This is a two step process. First, create a shortcut from the target app file (see Figure 2 right-click context menu => Create shortcut). Open the Startup folder (+R, enter *shell:startup*, OK) and drag the new shortcut from the AutoHotkey folder to the Startup folder.

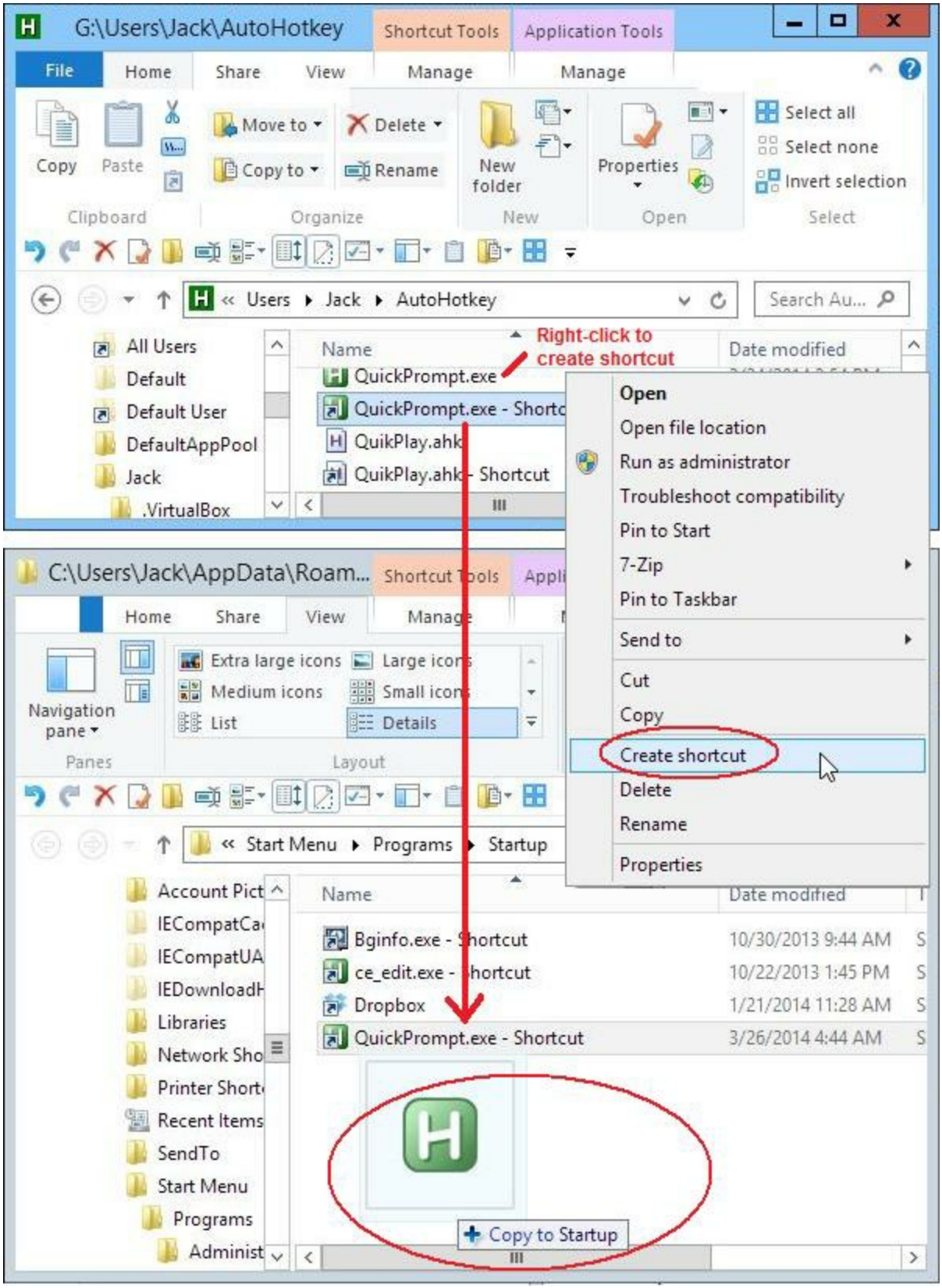


Figure 2. For automatic startup, create a shortcut by right-clicking on the primary file name, open the Startup folder, then drag the shortcut into the Startup folder.

All done! Now every time you log into your computer the new AutoHotkey app will load. If you like the way things are with your new stolen apps, then there is no need to read any further. But, if you want to make a few alterations

that might work better for you, then continue on.

Simple Script Tailoring for Personal Use

Maybe everything about your new app will be exactly the way you like it, but usually there is a little tweaking to do —even if it's just to change the hotkey combination(s). Since you're already staring at the code in Notepad, it's usually pretty simple to make minor adjustments, then Save them (CTRL+S). As examples of how to make these types of adjustments to AutoHotkey scripts, I use these apps GooglePhraseFix, HideWindow, WindowRollup, and QuickPrompt—all of which I stole from an AutoHotkey forum. (These scripts are also available at the *ComputorEdge* Download site.)

Finding and Changing the Hotkey

Hotkeys are distinguished by a double colon (::) appearing after what is usually a combination of at least two characters which often include one or more of ^, !, #, or +. The entire snippets may be all on one line, but usually it consists of a number of lines of code concluded with the *Return* command. When the entire script consists of one hotkey combination (as is the case with the GooglePhraseFix.ahk app in Figure 3), the hotkey combination can be change by editing the code before the double-colon. The CTRL key is "^", the ALT key is "!", the WIN key ( ) is "#", and the SHIFT key is the "+" character. See Hotkeys for more information and other options. I changed the hotkey combination from ^!C to ^!G (for Google) because I use the letter "C" in too many other hotkey combinations.

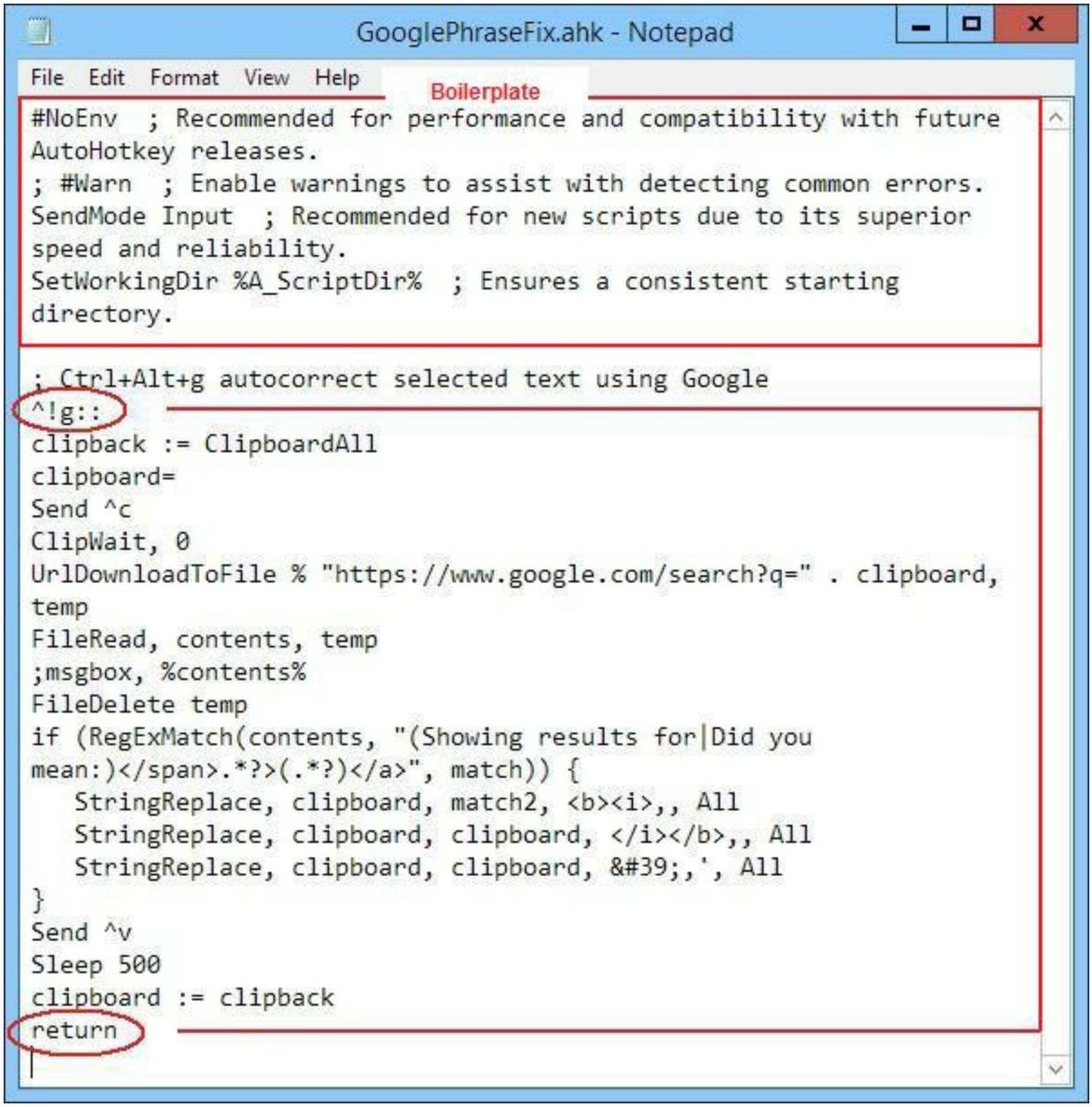


Figure 3. The entire app is one hotkey combination starting with ^!g:: (CTRL+ALT+G) and concluding with the "Return" command. Change the characters before the double-colon to change the hotkeys. This type of enclosed hotkey snippet can be added to the end of any other script without adjustment to combine the two scripts.

One advantage to a script such the one above is that it is completely self-contained in one hotkey block. That means you can add the app to any other script by placing the entire code at the end of the other script's file. Actually, you can place it almost anywhere in the file except in the auto-execute section at the beginning which runs on load—as long as it doesn't break up another block of code (hotkey, hotstring, subroutine, or function).

In the *HideWindow.ahk* script from AutoHotkey Scripts the hotkey combination is saved in a variable:

```
; This is the hotkey used to hide the active window:
mwt_Hotkey = #h  ; Win+H

; This is the hotkey used to unhide the last hidden window:
mwt_UnHotkey = #u  ; Win+U
```

Without the comments it would be more difficult to know where to change the hotkey combinations in this script.

(Any line or partial line preceded with a semicolon (;) is ignored by AutoHotkey—a comment line.) Hotkeys may also be preceded by the Hotkey command—which is the second most likely place to find and make changes to hotkey combinations.

Adding Scripts to Other Scripts

A while back I wrote a column about "How an AutoHotkey Script Is Processed." Knowing how this works is especially important when combining scripts. If a script includes an auto-execute section (even if it only contains one or two commands) it must appear appropriately in the auto-execute section at the beginning of the final combined script.

For example, in the WindowRollup.ahk script from the AutoHotkey Scripts site there is a one line OnExit command in the auto-execute section which calls the subroutine *ExitSub* (see Figure 4). This line must appear in the new combined scripts auto-execute section (without the *return*). The subroutine (ExitSub:) can be placed at the end of the file. However, since many scripts also use *OnExit* subroutines, it may be necessary to check for conflicts and combine the subroutines.

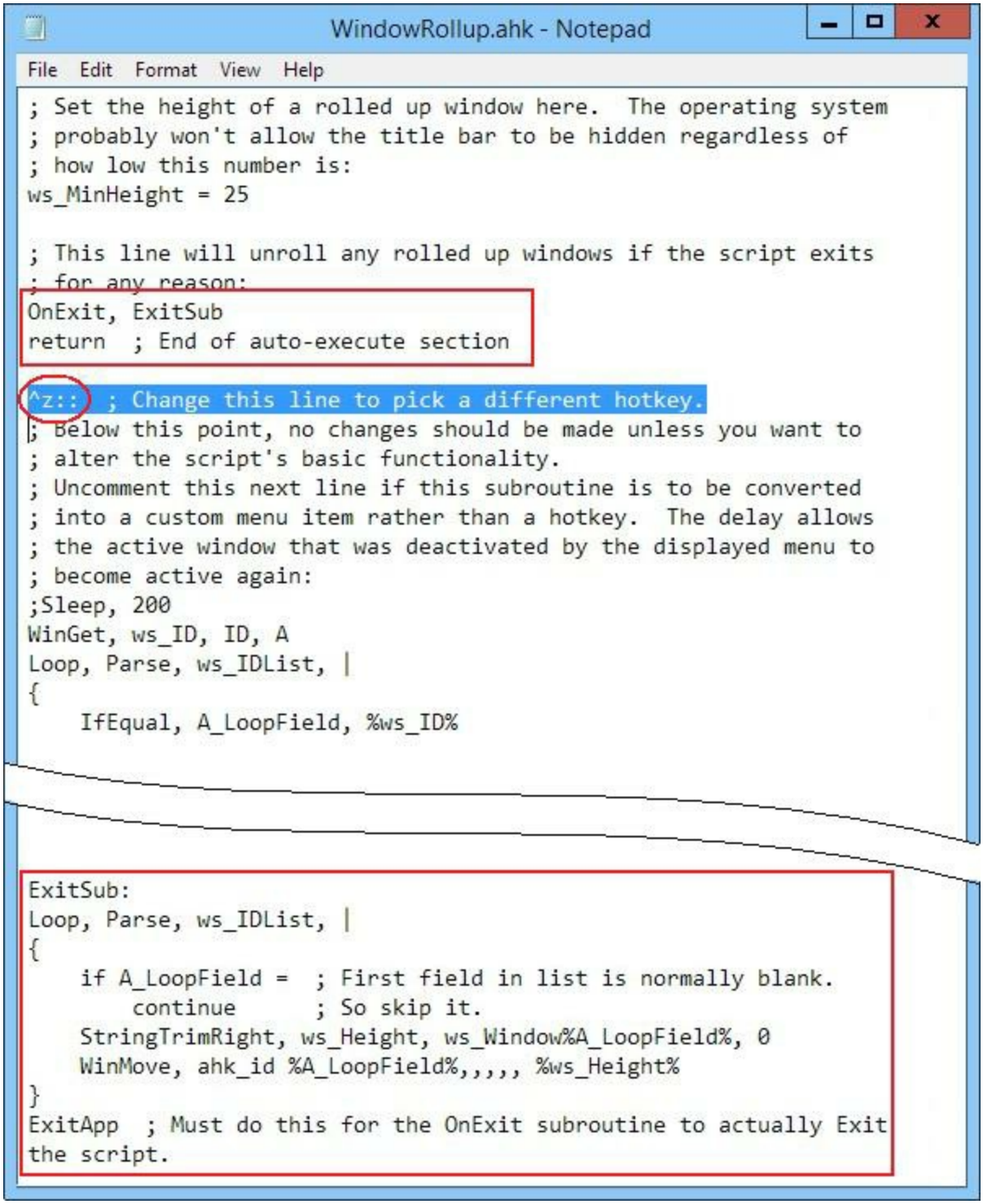


Figure 4. The OnExit command calls the ExitSub routine to unhide all the previously hidden windows before exiting. If there is more than one OnExit command then the subroutines must be combined in some manner—possibly one calling the other before exiting.

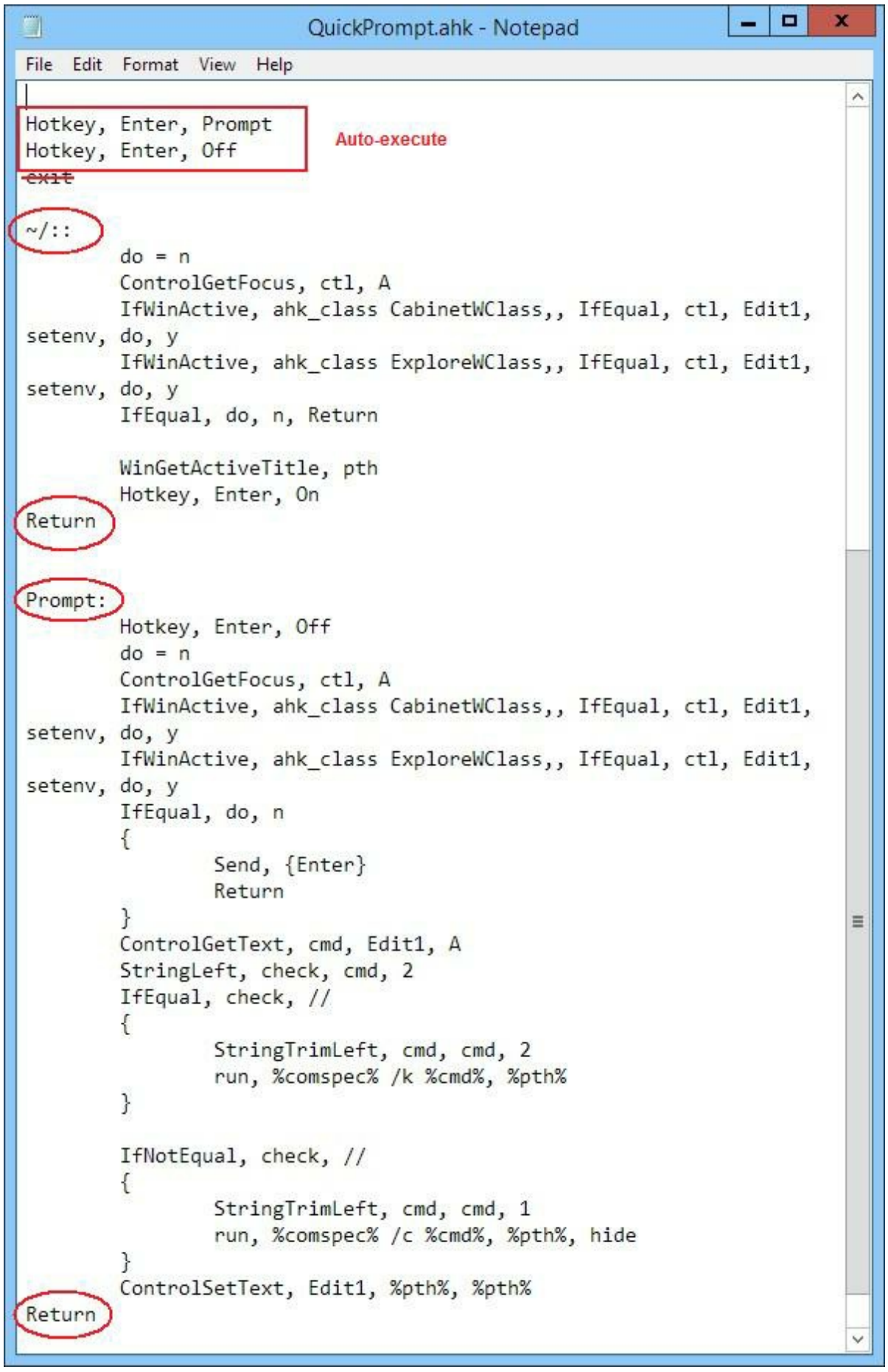
In this app I changed the hotkey combination from +Z to CTRL+Z with the following line:

```
^z::  ; Change this line to pick a different hotkey.
```

However, CTRL+Z is the almost universal Undo command, so I should probably look for something else.

More Auto-Execute

In the QuickPrompt.ahk script found in the AutoHotkey forum there are two Hotkey command lines in the auto-execute section (see Figure 5). These two lines need to be placed in the auto-execute section of the combined script whereas the following hotkey (~/) and the *Prompt:* subroutine can be placed elsewhere in the script as long as they don't break up any other code blocks.




```
Hotkey, Enter, Prompt
Hotkey, Enter, Off
exit

~/::
        do = n
        ControlGetFocus, ctl, A
        IfWinActive, ahk_class CabinetWClass,, IfEqual, ctl, Edit1,
setenv, do, y
        IfWinActive, ahk_class ExploreWClass,, IfEqual, ctl, Edit1,
setenv, do, y
        IfEqual, do, n, Return

        WinGetActiveTitle, pth
        Hotkey, Enter, On
Return


Prompt:
        Hotkey, Enter, Off
        do = n
        ControlGetFocus, ctl, A
        IfWinActive, ahk_class CabinetWClass,, IfEqual, ctl, Edit1,
setenv, do, y
        IfWinActive, ahk_class ExploreWClass,, IfEqual, ctl, Edit1,
setenv, do, y
        IfEqual, do, n
        {
                Send, {Enter}
                Return
        }
        ControlGetText, cmd, Edit1, A
        StringLeft, check, cmd, 2
        IfEqual, check, //
        {
                StringTrimLeft, cmd, cmd, 2
                run, %comspec% /k %cmd%, %pth%
        }

        IfNotEqual, check, //
        {
                StringTrimLeft, cmd, cmd, 1
                run, %comspec% /c %cmd%, %pth%, hide
        }
        ControlSetText, Edit1, %pth%, %pth%
Return
```

Figure 5. The two Hotkey lines in the beginning of this script should appear in the auto-execute section of a combined script. The other blocks (~/:: through to Return and Prompt: through to Return) can appear anywhere in the script after the auto-execute section at the beginning of the file.

The purpose of the two *Hotkey* commands is to set up then turn off the ENTER hotkey which calls the *Prompt:* subroutine. Then *Hotkey, Enter* can be turned on and off as needed—which is only when the cursor sits in the Windows Explorer address field.

The tilde (~) tells AutoHotkey to pass through the slash (/) to whichever program is active. This prevents AutoHotkey from overriding other uses of the slash character outside of the conditions for this app in Windows Explorer. In this case ~/ is more than just a hotkey combination. If you change this hotkey, then you would need to make other adjustments in the script.

These are just a few hints for any beginner who wants to abscond with AutoHotkey scripts and make them truly their own. The following are more detailed explanations of how these favorite scripts work.

* * *

Four AutoHotkey Apps Worth Stealing

From time to time I come across AutoHotkey Windows apps which offer new features which enhance the computing experience. Many of these I add to my own bag of trick. I've found a few AutoHotkey apps that are worth introducing to the 90% to readers. These apps are all free and available for download at the *ComputorEdge* download site. While these are all written in AutoHotkey, you *do not* need AutoHotkey installed (or even know anything about AutoHotkey) to use them. They have all been compiled into executable command files (*.exe* extension) and, once extracted from the ZIP file, run with a simple double-click on the filename.

I didn't write these apps myself. I borrowed (or legally stole) the source code from other people who are smarter than me and as explained above made minor changes (mostly to change the hotkey combinations for my own purposes). The truth is that there are portions of each of these scripts that I didn't completely decipher. Although I could see that the code was completely safe, some of the techniques used were in areas of AutoHotkey I have yet to investigate. (Each of these scripts was found at the AutoHotkey Web site and has been reviewed by people with far more expertise than me.) The most important factor is that these apps work.

Since I compiled each of these scripts myself into EXE files for running on any Windows computer, I can personally guarantee that running the programs found on the ComputorEdge *download site is safe.*

I've already added a couple of these apps to my master tools file for loading on boot up and am considering including the rest.

Google Spelling and Grammar AutoCorrect

When doing a Google search, it's amazing how often it will return the results for "Did you mean?" rather than the typo filled garbage I typed in. It is usually right. An AutoHotkey forum member with the handle *aaston86* decided to use this more-often-than-not correct information as a spell checker and whipped together a short "Autocorrect Anything" app which does just that. This cool little app uses Google search to correct selected phrases by accessing the "Showing results for/Did you mean:" line in a Google search results page.

When I started testing it, I quickly found that it did much more than just fixing spelling errors. It was also a reasonable grammar checker. For example, if I type the following line, highlighted it, then hit the hotkeys (I use CTRL+ALT+G):

```
If your going to you're house
```

it replaced the text with:

```
If you're going to your house
```

Since my dyslexic fingers often confuse "your" and "you're", this is a boon to my writing efforts. I also tested:

```
to much
two soon
to slow
and to fast
```

yielding:

```
too much
too soon
too slow
and to fast
```

Mostly right, but I was going *to fast* today anyway. Obviously, there is no way that everything can be caught. There are just *too* many ambiguities in the English language.

While there are numerous spelling checkers available, this app, which I call GooglePhraseFix, is oriented toward checking groups of words. It seems best at finding typos since if every word is spelled and used correctly it tends to come up empty. Plus, you occasionally may need to run it more than once. For example, the first time:

```
th quck brwn fx jmped over the lzy dg:
```

becomes:

```
the quick brown fox jumped over the lzy dg
```

The second time through, it was fixed:

```
the quick brown fox jumped over the lazy dog
```

Another possible use is for programmers. For example, I noticed that it could correct AutoHotkey syntax:

```
WinActvte
```

is recognized as

```
WinActivate
```

and

```
WnGetTtle
```

becomes

```
WInGetTItle
```

Note that in this case when a missing letter is added after a capital letter it's capitalized. It's a quirk of this search on Google. In fact the accuracy of this app is totally dependent upon how Google sees things. If you highlight more than one paragraph, Google will script out all of the carriage returns and make it one long line. Also, depending on the context of the error, it may be completely missed because of another possible usage.

This Google phrase fix app is certainly not a cure all for grammar and spelling problems, but there are many situations where it will help out—especially if you're cleaning up something someone else wrote. I changed the activation hotkey combination and added a line of code to convert apostrophes properly. You can download the app *GooglePhraseFix.exe* and the AutoHotkey script *GooglePhraseFix.ahk* in the fileGooglePhraseFix.zip from the *ComputorEdge* download site.

Hide a Window So No One Knows It Exists

Most people have seen apps that either put up a fake work screen or quickly minimize your current active window. Nobody likes people looking over their shoulder while they are using a computer—especially if it's their boss and they're playing a game or wasting time on Facebook. The problem with those fake screen apps is that the real running window can still be found when the fake screen is removed. Or, the active task icon continues to show up in the Windows Taskbar. Anyone with even a little knowledge of Windows will know that something's going on. This HideWindow app makes a window disappear and even removes the active Taskbar icon making it almost impossible to find the hidden window—unless someone knows where to look.

Other than not getting caught goofing off, there are more reasons to use this hidden window app. It declutters the Desktop as well as the Taskbar by removing running windows. In essence, HideWindow takes any active process and moves it into the background. It won't interfere with other programs, nor is anyone likely to accidentally stop it. (This could be useful for programs such as BitTorrent engaged in long downloads.)

For an example of how it works, the Desktop in Figure 1 is covered with various open windows. We could minimize each window and use the Taskbar to restore each one when needed.

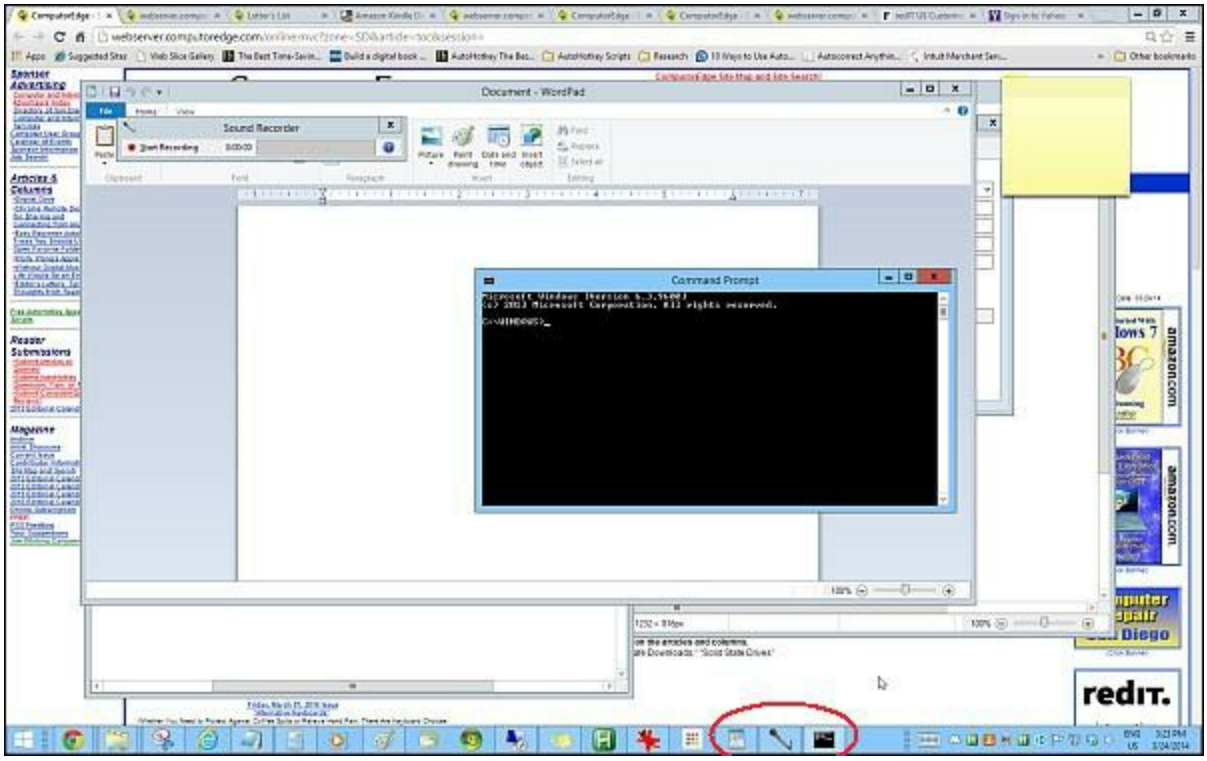


Figure 1. Numerous windows are open on the desktop. Three of them (not pinned) show up on the Taskbar.

However, by hitting WIN+H (+H) each active window disappears from the screen. It is not minimized, but hidden. If not pinned to the Taskbar, it will no longer appear in the Taskbar. If the program is pinned to the Taskbar (Windows 7 and 8 only), it will appear as if there is no instance of the program running—no window thumbnail when hovering over it. If you click the pinned icon, a new instance of the program will be opened in a new window. The

hidden window will not appear, although it is still running in background. (The exact behavior depends upon the version of Windows running.)

The last hidden window can be revealed with the WIN+U (+U) hotkey combination. Use +U again and the next hidden window pops up. (The hotkey combinations can be changed to your preference in the AHK script file as previously explained. To restore specific windows, right-click on the AutoHotkey icon in the System Tray (green icon with white "H" that's labeled HideWindow (*.exe* or *.ahk*) when the cursor hovers over it), then select an individual hidden window from the list or Unhide All Hidden Windows (see Figure 2).

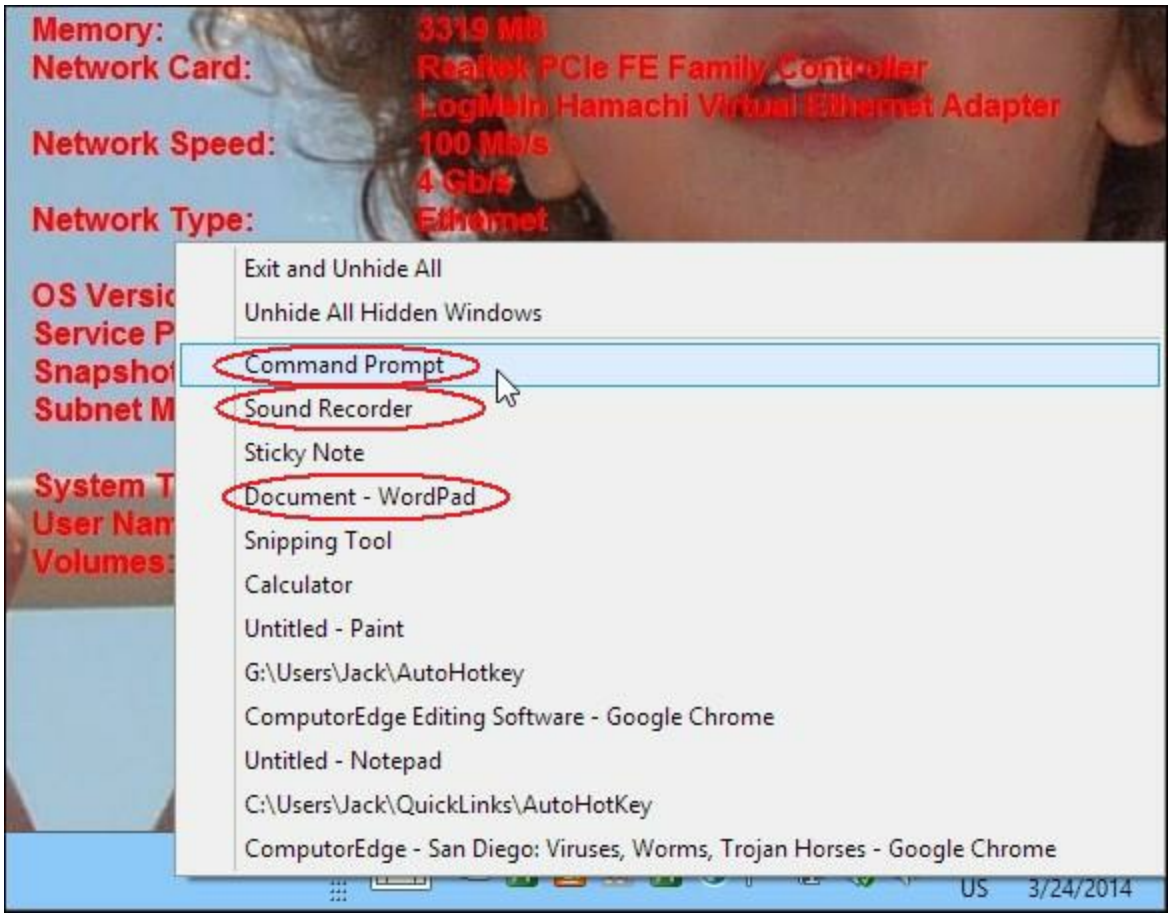


Figure 2. As the windows are hidden (WIN+H) they disappear, are removed from Taskbar (or, if pinned, show as inactive with no thumbnail), and added to the right-click menu for the AutoHotkey icon in the notification area. The three items circled in red (not pinned to Taskbar) no longer appear on the Taskbar. Select a specific window to reactivate it or Unhide All.

This is a pretty old app with the last changes posted on the AutoHotkey site in 2005, but it works well in all versions of Windows. The original source is posted at the AutoHotkey scripts page ("Minimize To Tray Menu"). I've copied it to the *HideWindow.ahk* and compiled it into the executable (for running on any Windows computer with a double-click) *HideWindow.exe*. Both can be downloaded from the *ComputorEdge* download site in the ZIP file HideWindow.zip.

Rollup Windows for More Breathing Space

This next app is designed to semi-minimize active windows when there are numerous windows open making it easier to see what's behind them and quickly access running programs on the Windows Desktop. The difference with this

WindowRollup app from the previous HideWindow is that rather than hiding or minimizing an open window, it reduces the window size to the title bar (or close to it)—making it possible to keep a good number of active windows on the Desktop without them interfering with each other.

The app works as a toggle using the CTRL+Z hotkey combination (see Figure 3). (I changed the hotkey for the original +Z because I was using +Z for my QuickLinks menu app (reviewed in the *Digging Deeper Into AutoHotkey* e-book). I discuss how to make this change in the hotkeys above. I note in that column that you may not want to use CTRL+Z for the hotkey combination, since it is also the Undo hotkey in many programs. This app will block other CTRL+Z functions. It's easy to change the hotkey to something else by following the steps in this week's AutoHotkey column.)

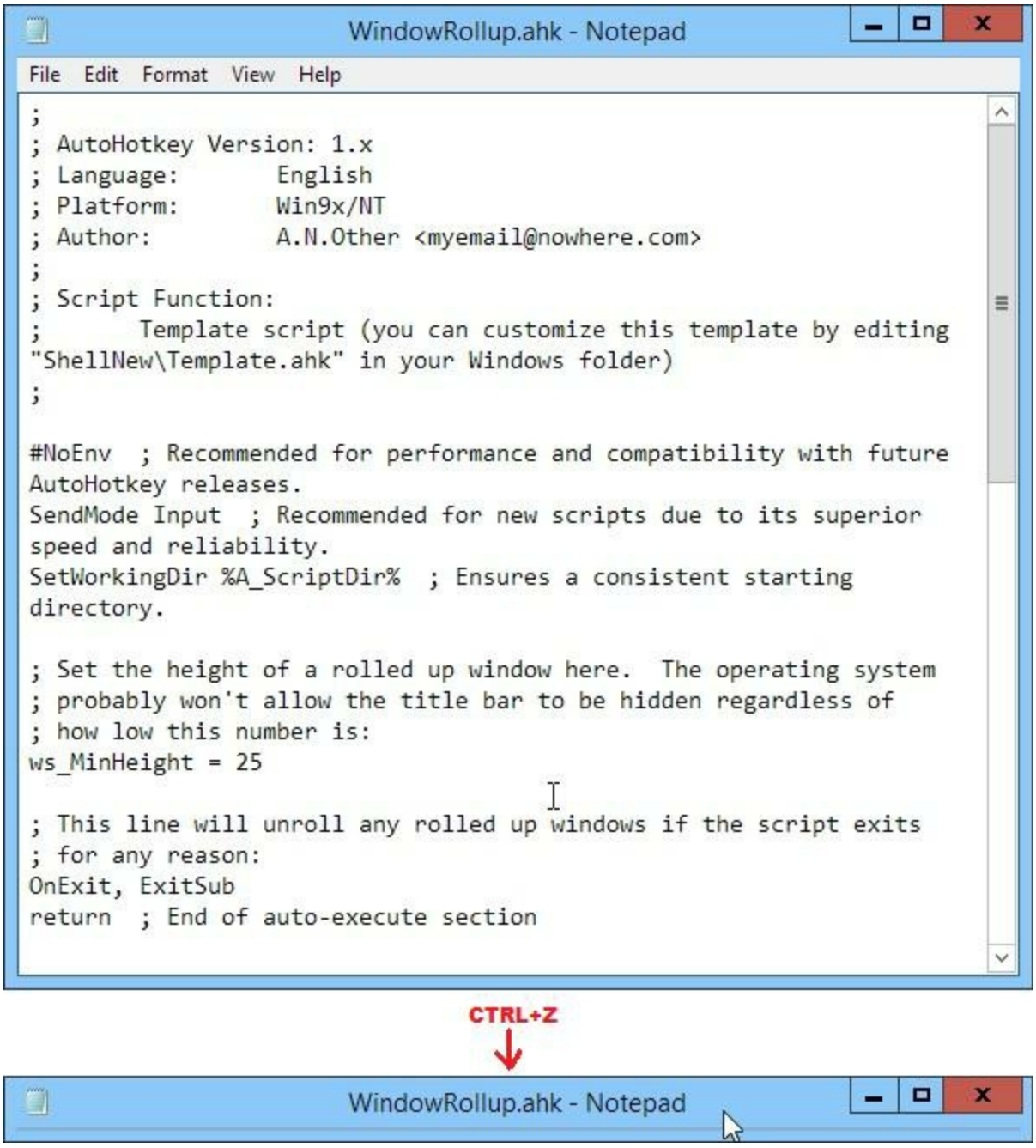


Figure 3. When the window is active CTRL+Z rolls up the window until only the title bar remains. Toggle CTRL+Z to unroll.

The net effect of rolling up multiple windows is quick, unhidden access to semi-minimized title bars on the Desktop

(see Figure 4). This is great for times you need to jump between windows, but don't want to lose view of them behind other windows.

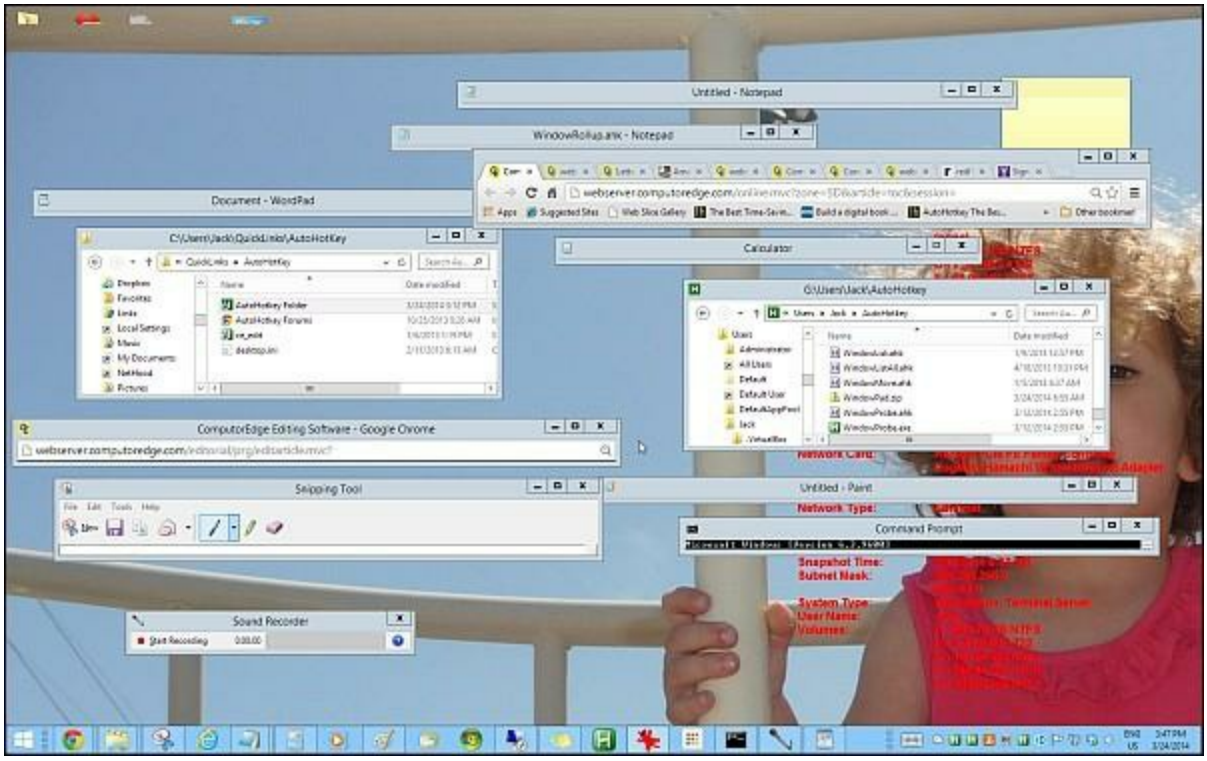


Figure 4. With windows rolled up, many more can be displayed on the same Windows Desktop.

The app called "Window Shading" was written by Rajak and the source is available through the AutoHotkey Scripts Showcase site. I call it WindowRollup and have compiled it into *WindowRollup.exe* for running on any Windows computer. The command file (EXE) and the source (*WindowRollup.ahk*) are contained in the *WindowRollup.zip* file available at the *ComputorEdge* download site.

Quick Command Prompt

If you need to access the command prompt (see Figure 6) on a regular basis, then you might want to add it to Windows Explorer (File Explorer in Windows 8). This next app which I call QuickPromp adds the Command Prompt capability directly to the folder/file path field (address bar) in Windows Explorer (see Figure 5). You no longer need to open the Command Prompt from the Start Menu or pin a quick launch to the Taskbar. Wherever you have Windows Explorer open, you can either run commands as if you were using the Command Prompt or open the window set to the current folder. The activating key is a slash (/) as shown in Figure 5.

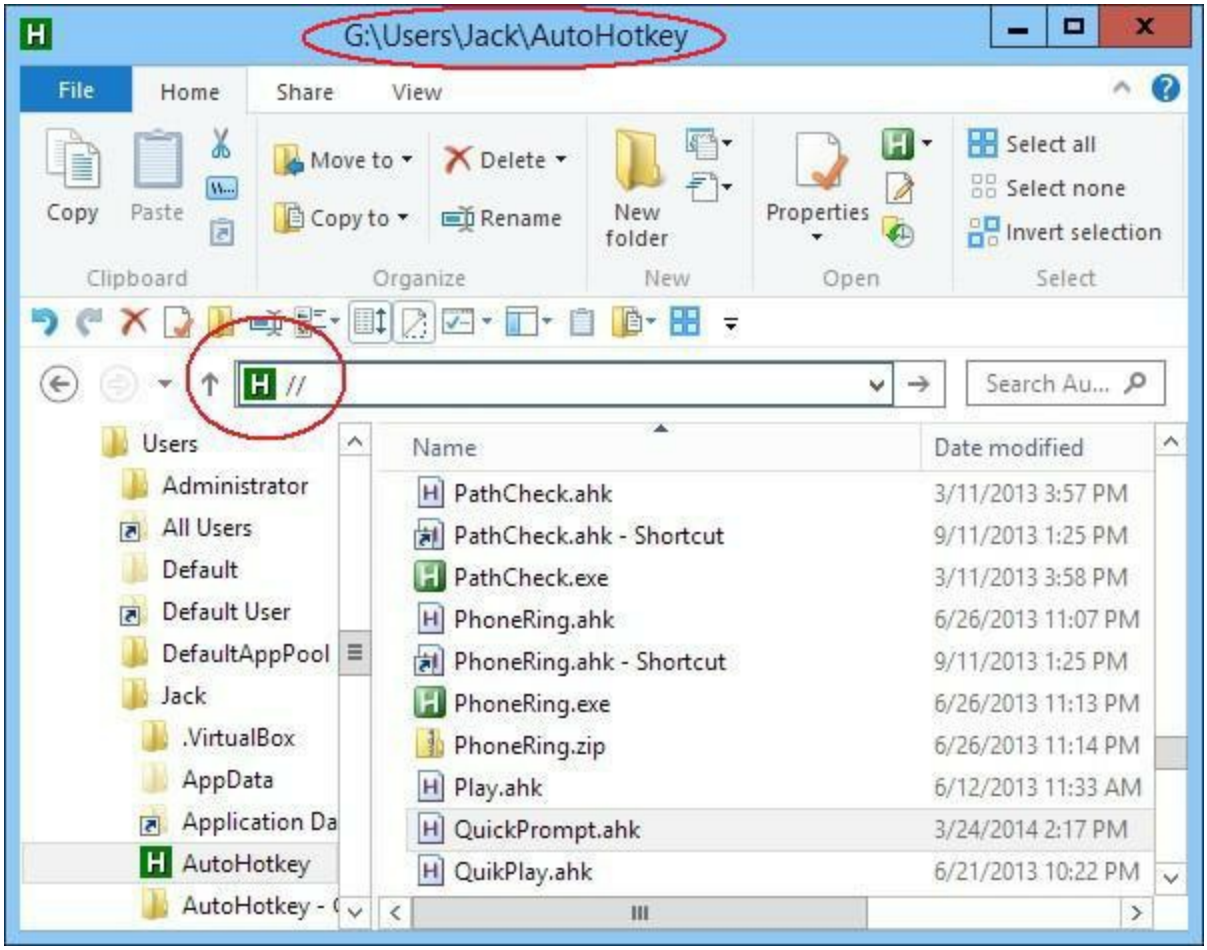


Figure 5. Click the icon on the left side of the path field (address bar) in Windows Vista, 7, or 8 (not required in Windows XP) in Windows (File) Explorer. Enter one slash (/)or enter two slashes (//), then hit RETURN to run commands or open the Command Prompt set to that folder respectively.

Once the app is loaded (double-click on downloaded and extracted *QuickPrompt.exe* file), open the Command Prompt at the current location by entering two slashes (//) and hitting RETURN. (In Windows Vista, 7, 8, and 10, first click on the folder icon on the left side of the path field. This gives input access to the field.) The Command Prompt window will open (see Figure 6). However, you don't need to open the Prompt window to run a command.

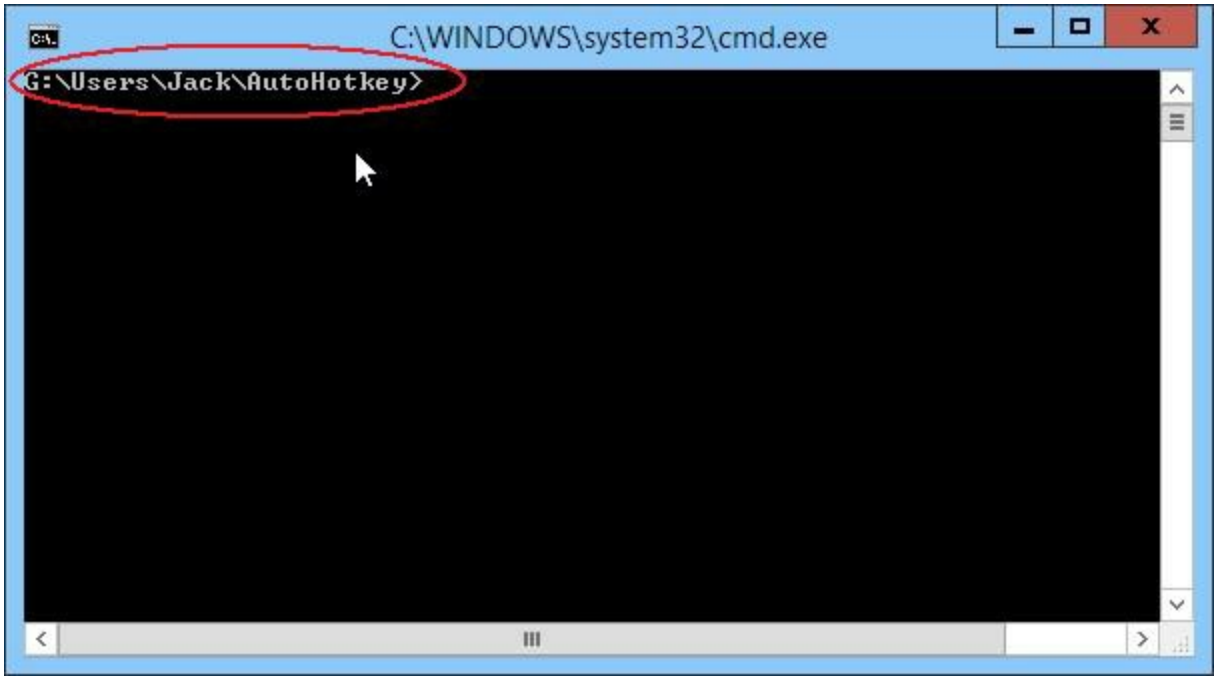


Figure 6. The Command Prompt can be opened directly from Windows Explorer.

To run Prompt commands from Windows Explorer, enter one slash (/) in the path field followed by the command. For example, to open the calculator enter */start calc*. *Start* is the command for running a program and *calc* is the name of the Windows calculator program (see Figure 7). The calculator window will open.

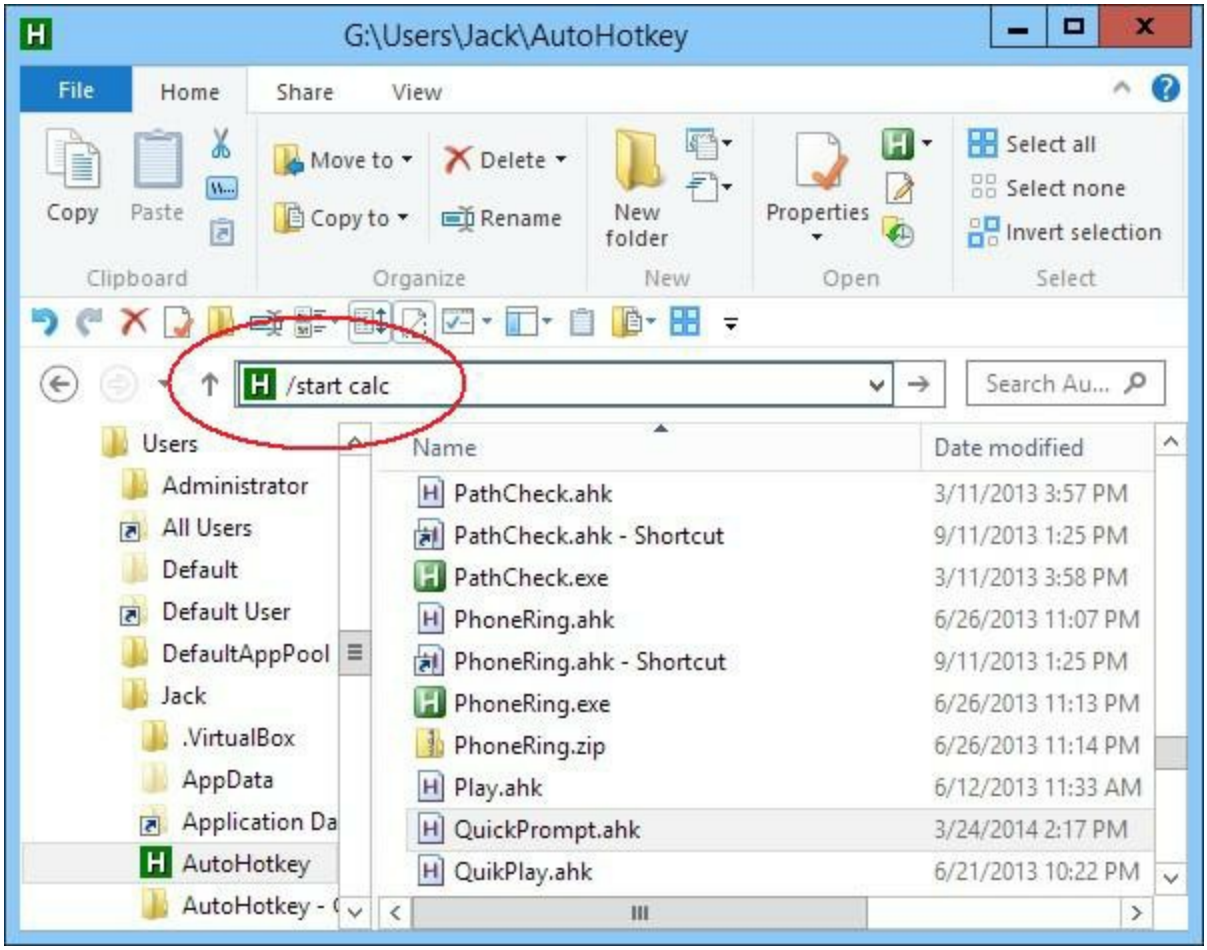


Figure 7. Programs can be launched directly from the Command Prompt added to Windows Explorer address field by first entering a slash (/).

There is a fairly long list of the usual utilities and apps you can run. However, you are not limited to running programs. There is also a list of Windows commands available.

I found the source code for *QuickPromp.ahk* at the AutoHotkey forum. This one was also written by Rajak and I did not make any modifications. I merely compiled it into *QuickPrompt.exe* for running on any windows computer. Both files can be extracted from QuickPrompt.zip after downloading from the *ComputorEdge* AutoHotkey download site.

Anyone Can Use These Apps

I've highlighted these particular apps because of the hundreds I sifted through, they were both simple and the most universally useful. Anyone can add them to any Windows computer without any knowledge of AutoHotkey. The fact that I think "AutoHotkey is the most powerful free Windows utility software ever" is incidental. You don't need to ever look at AutoHotkey to use these tools. Just download and extract the EXE files and you're ready to go—no installation required! The great thing about free software is it's free!

Chapter Ten: AutoHotkey Versus AutoIt

“Choosing a Windows scripting program for automating your computer activities.”

Both scripting languages are from the same roots, but which one is right for you?

When people begin investigating AutoHotkey, they soon discover that its roots lie in the older Windows scripting language AutoIt. Both programs have active support and serve very similar purposes by adding power to your Windows computer. The question is "Which language is right for you?" Before answering that question the two languages must be compared and contrasted. A Google search will uncover many discussions comparing the two Windows scripting tools, but most of the conclusions come down to people liking what they know best. In this chapter I endeavor to offer enough factual information about both free programs for you to make an informed decision—although the only way to know for sure which is best for you is by testing both..

Full Disclosure: While I attempt to put aside my personal bias (I've written five AutoHotkey books with a sixth on its way) and give a fact-based appraisal grounded upon my research and knowledge of programming languages, I do not have the same in depth knowledge of AutoIt that I do of AutoHotkey. Fortunately, there is a good deal of data for review from people who have used both languages.

First, since both AutoHotkey and AutoIt are free to use, there is no reason not to test or use both. I'm a great believer in using the right tool for the job—especially when the tools are free. If you are comfortable with writing scripts, then you may be served well by playing with each language for particular purposes. There may be applications where one is more suitable than the other. The only cost is the time it takes to do it.

It's important to note that the current versions of the AutoHotkey and AutoIt languages are not interchangeable. Each has its own particular syntax (set of commands and functions) which, while they are capable of accomplishing the same thing, are written in a totally different manner. To make an informed decision you will want to see those differences. A couple of simple examples are provided below.

The Names AutoHotkey and AutoIt

The name of each scripting language gives us insight into the differences between the two. While you can do almost anything in AutoHotkey that you can in AutoIt (and vice versa), there are some noticeable differences. The underlying meaning of the terms AutoIt and AutoHotkey express the original impetus for each program. While both are used for Windows program automation, that is the primary thrust of AutoIt—thus the name. As suggested by the name AutoHotkey, hotkey creation and hotstring replacement through keyboard action is integral to the program. While AutoHotkey has implemented simple ways to create hotkeys and hotstrings, the AutoIt community has never consider those features high priority—although there is a specific function for creating hotkeys in AutoIt —*HotKeySet()*.

It seems that the split occurred after programmer "Chris Mallett's proposal to integrate hotkey support into AutoIt v2" received little response from the AutoIt community causing Mallett to start writing his own version of the then open source AutoIt—calling it AutoHotkey. The name AutoHotkey demonstrates the primary functional difference between the two scripting languages. AutoHotkey offers simple direct support for one-line assignment of hotkeys and hotstrings while the current version of AutoIt requires the use of functions to implement the same or similar hotkey and hotstring features. (See the comparisons below.) This has caused the two language to be viewed very differently. You might say that AutoIt is for Windows automation and AutoHotkey for easy implementation of Windows hotkeys and hotstrings plus Windows automation. (This is not a completely fair comparison since Windows automation, app development, and hotkey/hotstring implementation can be done with either scripting language. They

are just done in a different way.)

The result is that many AutoHotkey users who have switched have been extremely pleased with the results after converting most of their scripts to AutoIt, but, even then, they tend to continue using AutoHotkey for their hotkey/hotstring implementations—especially with long list of hotstrings such as the AutoCorrect scripts. Understand that most of these individuals who made the jump to AutoIt from AutoHotkey are more experienced with script writing and programming.

Third Edition Update: *After completing a series of blogs on AutoHotkey Hotstrings and Hotkey techniques, I've come to realize that there are many people who might want to use AutoHotkey just for these easy built-in Hotstring and Hotkey structures. If you're a writer, student, or anyone else who does a lot of word processing, then you'll be amazed at the AutoCorrect and text expansion/replacement capabilities of AutoHotkey Hotstrings. For any Windows user, the simple Hotkey format adds automating power to their computer. There are many cool text tricks which you can implement with Hotstrings. This applies equally to the simple built-in basic Hotkey structure in AutoHotkey. The original split between AutoIt and AutoHotkey occurred because of these two easy features. They make it much easier for the beginner to get started with AutoHotkey.*

For beginning script writers, the syntax for hotkeys and hotstrings (text expansion) is much simpler in AutoHotkey (a couple of colons *::*). This attracts novice users since their first scripts often consists of this type of easy coding. As these beginners become more daring, they naturally venture into the other slightly more complex commands found in AutoHotkey. The function oriented language of AutoIt may be intimidating for the newbie while it might be more natural for the experienced programmer.

The commands in the early versions of AutoIt looked very much like those in the current release of AutoHotkey. After the split between the two languages, AutoIt moved in a new direction (version 3) using more BASIC-like functions while AutoHotkey continued with the original command structure used in the early versions of AutoIt. I don't know that there is a clear advantage of one language structure over the other, but each certainly has its own separate learning curve.

AutoHotkey Versus AutoIt for Hotkeys and Hotstrings

The following is an example of what's required to set up a hotkey in AutoHotkey:

```
Insert::Run, %A_MyDocuments%
```

It's one line of code that turns the INSERT (*Insert*) into a hotkey which opens your Windows Documents folder. The percent signs (%) surrounding the built-in variable *A_MyDocuments* evaluates the path to the Documents folder.

In AutoIt it takes few more lines:

```
HotKeySet("{Ins}", "openDocs")
Func openDocs()
  Run('explorer ' & @MyDocumentsDir)
EndFunc
```

The first line uses the AutoIt function *HotKeySet()* to assign the INSERT key (*Ins*) to run the user defined function *openDocs()*. The *openDocs()* function uses the *Run()* function to open the Windows *Explorer.exe* program at the user's documents folder—designated by the macro (built-in variable) *@MyDocumentsDir*. The ampersand (*&*) appears to be the symbol for concatenation (combining strings). As can be seen, there are a few more lines needed in AutoIt. This is comfortable for experienced script writers, but could be confusing for the average Windows user.

This next example is a text expansion (or hotstring) for the phrase "by the way" in AutoHotkey:

```
::btw::by the way
```

Hotstrings are assigned by placing double colons (*::*) at the beginning of the line followed by the hotstring and another set of double colons *(::btw::*). The replacement text (*by the way*) follows the second set of double colons. Once the script is loaded, anytime "btw" is typed followed by a space or punctuation, it is automatically replaced with "by the way."

I couldn't find a built-in hotstring function in AutoIt, but I did locate this example which includes a special AutoIt script (*HotString.au3*) to create the function *HotStringSet()*:

```
#include <HotString.au3>

HotStringSet("callme{enter}", examplefunction)

While 1
    Sleep(10)
WEnd

Func examplefunction()
    MsgBox(0,"","You typed callme! :)")
EndFunc
```

To replicate the AutoHotkey example would require the use of the AutoIt Send() function in place of the *MsgBox()* function. If you need text expansion, then it looks like AutoHotkey may be your program of choice. This is not a minor point since the same simple hotstring technique found in AutoHotkey (in a slightly different format) can be used to run a set of commands in the same manner as a hotkey subroutine.

Both languages are capable of much more involved programming and Windows app building which have nothing to do with either automation or hotkeys/hotstrings. AutoHotkey syntax uses primarily text commands whereas AutoIt uses functions. For example, to add a button to a Graphic User Interface (GUI) window AutoHotkey uses the following command:

```
Gui, Add, Button [, Options, Text]
```

To create a Gui button in AutoIt the following function is used:

```
GUICtrlCreateButton("text",left,top[,width[,height[,style = -1[,exStyle = -1]]]])
```

This is not a major difference for experienced programmers, but it you've grown accustomed to doing it one way, switching to another method requires developing a different mindset. While there is no particular advantage to the AutoHotkey command structure, it may look simpler to the novice.

The Confusing Side of AutoHotkey

In many ways the current AutoIt version is a much cleaner language than AutoHotkey. It has not fallen victim to the confusion caused by maintaining backward compatibility with older scripts. Since virtually any older AutoHotkey script will run with the current 1.1 version (AutoHotkey_L) of AutoHotkey, there are numerous redundant commands and methods for getting the same thing done. This is a source of confusion for AutoHotkey users because, although there is more than one way to get something done, each requires a different implementation.

The classic example is the use of both the equals sign (=) and a colon plus the equals sign (:=) to assign a value to a variable:

```
MyString = This is a literal string
CopyOfVar = %Var%
```

and

```
MyString := "This is a literal string."
CopyOfVar := Var
```

are identical sets of expressions. But if you use the wrong one in the wrong place with the wrong format, your script won't run properly. There are a number of other examples of this type of duplication caused by legacy AutoHotkey commands included for backward compatibility. Often the different variations may be used in the same script.

There is an attempt underway to fix the problem and clean up the code with version 2.0 of AutoHotkey. (Version 2.0 is still in the alpha stage of development and not recommended for newbies.) It is designed to eliminate the excess baggage. However, that would mean that many of the scripts available today would no longer run properly with the new version of AutoHotkey. It is difficult to know how widely this new, cleaner version would be accepted. The changes which would be required in older scripts are not well documented and many people will not relish the extra work. As far as I know, AutoIt does not have a similar issue.

Interest in AutoHotkey Versus AutoIt

Based upon links found on Google Trends, interest in AutoHotkey appears to have caught up with AutoIt. (See Figure 1.) This is a momentum which is not likely to reverse.

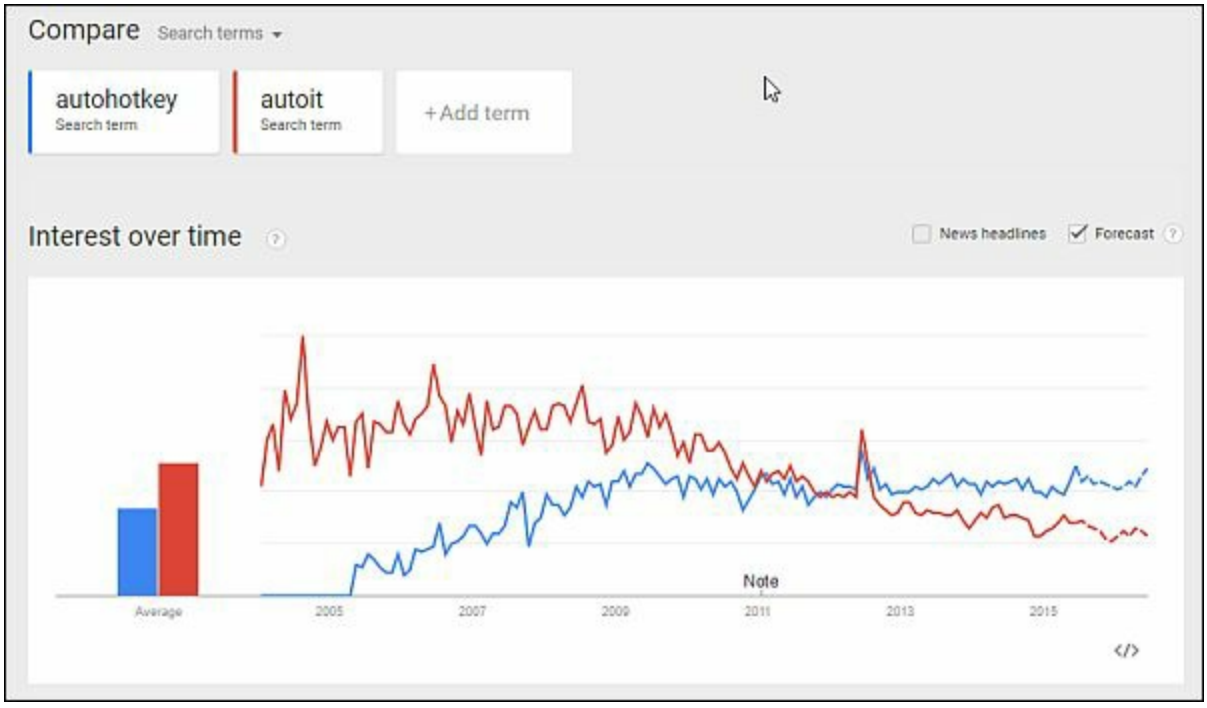


Figure 1. Interest in AutoHotkey in the United States has slowly caught up with AutoIt. The trend favors AutoHotkey.

The primary reason for the growth of AutoHotkey is its initial simplicity. It's easy for the beginning script writer to get a result with just one line of code in a text file (a hotkey assignment or text string expansion). This is what got me started. Over time, and with a little curiosity, newbies start experimenting with other simple commands and scripts. Eventually, they are hooked and become part of the AutoHotkey community ready to help others.

Since many of the users don't consider themselves to be programmers, the AutoHotkey community works to help

each other and is welcoming to beginning users. There are numerous online tutorials and people don't usually face impatiences when they ask newbie questions on a forum. (My experience on the AutoHotkey forums is that at any given time there are usually a number of people online willing and able to answer novice questions.) This makes it easier for people who have never before written a program.

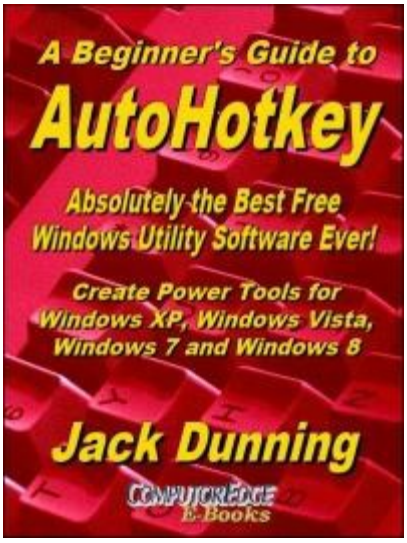
Using AutoIt is a bit more daunting for the newbie script writer. If you're not already a programmer, the concept of using functions can be a little difficult to wrap your brain around. As far as I can see, there is no easy entry point in AutoIt for the complete novice. AutoIt seems to be used more by computing professionals and possibly to a greater degree within corporations. (I have no data to back up this statement, but it was a feeling expressed by AutoIt users.) If I had started with AutoIt first, then, who knows(?), I could be writing AutoIt books. However, I'm not sure the AutoIt users are as likely to need the help. As it is I'm happy with AutoHotkey—even with all of its quirks and idiosyncrasies. It's good stuff.

Some people suggest that AutoHotkey is not as extensive as AutoIt. I would be forced to differ. While there may be some minor technical advantages to AutoIt, I haven't seen much bragged about in AutoIt (arrays, object oriented coding, etc.) which isn't now available in the current version of AutoHotkey 1.1. As far as I can see, both have a great deal of capability for building Windows apps. If more is needed, then possibly the programmers should consider using a programming language such as C++ or Java.

If you're an experienced programmer and the simple hotkey and hotstring assignment found in AutoHotkey is not a requirement in your work, then I would likely recommend AutoIt as your Windows scripting language. If you're a beginner, then AutoHotkey may be just what you need. If you're somewhere in between, then it's a coin toss. Or, maybe you should use both—depending upon what you want to do.

"A Beginner's Guide to AutoHotkey" Contents and Index

“The Table of Contents and Index from the e-book "A Beginner's Guide to AutoHotkey.”"



The second edition with more chapters and an index to the AutoHotkey commands found in the book is available in e-book form from the ComputorEdge E-Books Web site (in EPUB, MOBI and PDF formats). Jack's *A Beginner's Guide to AutoHotkey, Absolutely the Best Free Windows Utility Software Ever!: Create Power Tools for Windows XP, Windows Vista, Windows 7 and Windows 8* offers a gentle approach to learning AutoHotkey. The book is also available at Amazon.

Building Power Tools for Windows XP, Windows Vista, Windows 7 and Windows 8, AutoHotkey is the most powerful, flexible, *free* Windows utility software available. Anyone can instantly add more of the functions that they want in all of their Windows programs, whether installed on their computer or while working on the Web. AutoHotkey has a universality not found in any other Windows utility—free or paid.

Based upon the series of articles in *ComputorEdge*, Jack takes you through his learning experience as he explores writing simple AutoHotkey scripts for adding repetitive text in any program or on the Web, running programs with special hotkeys or gadgets, manipulating the size and screen location of windows, making any window always-on-top, copying and moving files, and much more. Each chapter builds on the previous chapters.

For an EPUB (iPad, NOOK, etc.) version of A Beginner's Guide to AutoHotkey click here!

For a PDF version for printing on letter size paper for inclusion in a standard notebook of A Beginner's Guide to AutoHotkey click here!

Table of Contents to "A Beginner's Guide"

Chapter One: How to Become a Windows Computer AutoHotkey Superhero

"An Introduction to AutoHotkey features. The best way to keep your job (or get a new one) is make yourself more valuable."

Most small offices have one or two people who have made themselves the in-house IT group. Without the job title, they apply their knowledge to keeping the computers running. Now there is another way to add to superhero status by making it easier to use Windows computers with AutoHotkey.

Chapter Two: Programming Is for Everyone

"Don't think you can program? Think again! Writing software scripts is not just for nerds."

Many people avoid programming because it looks too mysterious and complicated. However, it's really not all that hard. Most people could benefit from writing a little bit of code for their unintelligent computer.

Chapter Three: Installing AutoHotkey and Writing Your First Script

"AutoHotkey, an often overlooked utility program, could become your best friend for your PC."

Some people avoid AutoHotkey because it requires scripting. However, once you understand the possibilities there is plenty of incentive to learn a little coding. Here's how to install AutoHotkey and write your first script.

Chapter Four: More Basic AutoHotkey Techniques

"Changing the case of text from upper to lower and back again."

After introducing the free AutoHotkey utility program in the last chapter, Jack adds a few more techniques which can be used immediately by anyone with a Windows computer.

Chapter Five: Sharing AutoHotkey Scripts

"Compile your AutoHotkey scripts into an executable (EXE) file, plus restoring the original clipboard contents."

There is no need to install the complete AutoHotkey program on every computer. All you need to do is compile your scripts for use on any Windows computer. Plus, how to restore the original Clipboard contents after an AutoHotkey operation.

Chapter Six: Instant Search and Replace

"StringReplace to search and replace any text, anywhere, anytime using Loop and If."

So far we've demonstrated straightforward text substitution and manipulation. Now it's time to do search and replace for multiple items using a Loop and If.

Chapter Seven: Cool Date Tricks with AutoHotkey

"Enter the current date into any Windows program automatically."

AutoHotkey includes tools that make it easy to enter the time and/or date into your documents and files. There is even a cool pop-up calendar for picking alternative dates.

Chapter Eight: Powerful Screen Object Controls in AutoHotkey

"Using the Graphic User Interface (GUI) controls in AutoHotkey to build simple gadgets."

The addition of screen objects to AutoHotkey scripts increases the number of ways that you can use what is "absolutely the best free Windows utility ever." To see the possibilities, peruse the chapter or just look at the pictures.

Chapter Nine: Automatically Resizing Windows with AutoHotkey and User-Defined Functions

"Write a user-defined function for resizing windows to exact dimensions on your desktop."

AutoHotkey includes commands for manipulating the position and size of windows. Including these commands in a user-defined function will add more flexibility to your Windows computing.

Chapter Ten: Make Any Window Always-on-Top Anywhere, Anytime, Plus More "If" Statements

"Using AutoHotkey to make a window always-on-top."

Using AutoHotkey for always-on-top and using "If" conditionals to solve some window size and location problems.

Chapter Eleven: Opening Useful Hidden Windows Folders with AutoHotkey and Making a Help Pop-up

"An easier way to open hidden Windows folders, such as Startup...plus more AutoHotkey and an AutoHotkey help pop-up!"

It can be difficult to find some of the important Windows system folders. Here is the secret to opening them quickly. Then, add this technique to an AutoHotkey script to make it even easier. Also, how to make a pop-up Help window in AutoHotkey.

Chapter Twelve: Cleaning up the Desktop

"A tip for organizing Windows Desktop clutter and more power with AutoHotkey Replace."

Using program icons to quickly recognize new folders used for common file types, plus use AutoHotkey for e-mail addresses and adding "boilerplate."

Chapter Thirteen: Disabling Annoying Windows Hotkeys

"It's easy to Delete All when the Control key is next to the Shift key."

While most Windows hotkey combinations are very useful, there are times when you may want to disable one or two of them. There is a quick and easy way to do that in AutoHotkey.

Chapter Fourteen: AutoHotkey for Copying and Moving Files

"If copying files become tedious with a mouse, it may be time for AutoHotkey."

For copying and moving files, Windows is a point-and-click operating system. Sometimes it's just easier to write a short AutoHotkey script to manipulate file locations.

Chapter Fifteen: How to Cheat at Computer Games and Restarting with AutoHotkey Action Recorders

"AutoHotkey is great for empowering your avatar, plus automatic script creators for beginners."

One of the most popular uses for AutoHotkey is automating computer games. Plus, did you get bogged down with learning AutoHotkey? Actions recorders such as AutoScriptWriter can give you a fresh start.

Chapter Sixteen: Common AutoHotkey Messages and Errors Encountered by the Novice

"Here are a few issues every beginner should understand."

While all AutoHotkey users encounter these warnings and errors at some time, they can cause a great deal of frustration for the beginner. Here is how to deal with them.

Index to "A Beginner's Guide"

A—Index to "A Beginner's Guide"

B—Index to "A Beginner's Guide"

C—Index to "A Beginner's Guide"

D—Index to "A Beginner's Guide"

E—Index to "A Beginner's Guide"

F—Index to "A Beginner's Guide"

G—Index to "A Beginner's Guide"

H—Index to "A Beginner's Guide"

I—Index to "A Beginner's Guide"

K—Index to "A Beginner's Guide"

L—Index to "A Beginner's Guide"

M—Index to "A Beginner's Guide"

N—Index to "A Beginner's Guide"

O—Index to "A Beginner's Guide"

P—Index to "A Beginner's Guide"

R—Index to "A Beginner's Guide"

S—Index to "A Beginner's Guide"

T—Index to "A Beginner's Guide"

U—Index to "A Beginner's Guide"

V—Index to "A Beginner's Guide"

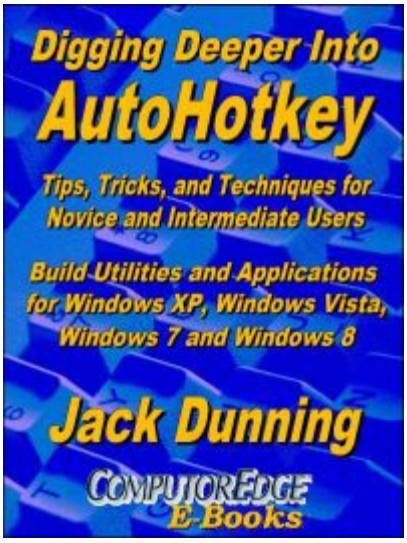
W—Index to "A Beginner's Guide"

X—Index to "A Beginner's Guide"

XButton; Chapter Fifteen

"Digging Deeper into AutoHotkey" Contents and Index

“The Table of Contents and Index from the e-book "Digging Deeper into AutoHotkey.""



Jack's second AutoHotkey book which is comprised of updated, reorganized and indexed columns from *ComputorEdge* is now available at ComputorEdge E-Books in EPUB, MOBI and PDF formats. Since the columns were not all written in a linear fashion, the book has been reorganized and broken up into parts by topic. The book is not for the complete beginner since it builds on the information in *A Beginner's Guide to AutoHotkey*. However, if a person is reasonably computer literate, they could go directly to this book for ideas and techniques without the first book. Digging Deeper is also available at Amazon.

For an EPUB (iPad, NOOK, etc.) version of Digging Deeper into AutoHotkey click here!

For a PDF version for printing on letter size paper for inclusion in a standard notebook of Digging Deeper into AutoHotkey click here!

The Table of Contents "Digging Deeper into AutoHotkey"

Whether quoting from a Web page or adding parenthetical remarks, AutoHotkey makes it easier to surround the text in a document or edit window.

Chapter Three: Tip for Long E-Mail Addresses

"There is no need to constantly type those long e-mail addresses."

If you have long e-mail addresses then this AutoHotkey tip will make your life easier.

Chapter Four: Always-on-Top Window Tip

"A tip for putting any window always-on-top...and off again."

The one-line AutoHotkey always-on-top script is a must-have for Windows Calculator and Sticky Notes.

Chapter Five: A Tip for Mild Dyslexia, Swapping Letters

"Here's a trick for swapping two mistyped letters."

A quick app for swapping two letters could help those of us who make typos.

Chapter Six: Do a Google Search from Any Program!

"Do a quick Web search!"

This is cool! Here is a short, simple AutoHotkey script that allows you to search the Web from any program or window. Anyone can do it!

Chapter Seven: The Disappearing Taskbar Trick

"Even the Windows Taskbar can be put away with the stroke of a key."

While this trick may not be the most useful, it does give insight into how AutoHotkey works.

Chapter Eight: Make the Insert and Caps Lock Keys Useful and the Missing Windows Key Solution

"Not all keyboards are alike. Here's how to deal with it."

Whether you're an accountant who wants a tab key closer to the number pad or there's no Windows key on your keyboard, AutoHotkey has a solution.

Chapter Nine: A Simple Way to Automate Any Windows Program with (or without) Mouse Movement

"An AutoHotkey script for speeding up any Windows program menu action, plus use a hotkey to simulate mouse movement."

Tired of navigating menus just to do something simple? Automate it in any Windows program with a single line AutoHotkey script. Not only can AutoHotkey automate keystrokes, but it can also simulate mouse movement and clicks.

Chapter Ten: AutoIt Window Spy for Identify Window Names and Coordinates

"Window Spy is possibly the most important utility to use when writing AutoHotkey scripts."

AutoIt Window Spy will give you everything you need to know about the window's inner workings and hidden

mechanisms you need to know when writing AutoHotkey scripts.

Part II: The Zen of Writing AutoHotkey Scripts

"AutoHotkey script writing is a process which does not necessarily go in a straight line."

This section of Digging Deeper into AutoHotkey has a two-part purpose. The first is to show new script writers how the process really works. If there are programmers who get it right the first time, I don't know any of them. The second part acquaints you with how AutoHotkey moves, positions and resizes windows, plus a script for finding hidden (off the screen) windows.

Chapter Eleven: The Zen of the Script Writing Process (Part 1): Snapping Windows

"Writing AutoHotkey scripts that work...eventually! Moving, positioning and resizing windows in Windows."

Programming is a process of trial and error. By doing the work one piece at a time, anyone can successfully write a script. In this first part a script for window resizing and moving is studied.

Chapter Twelve: The Zen of the Script Writing Process (Part 2): Snapping Windows

"Writing AutoHotkey Scripts that Work...Eventually!"

Jack continues writing an AutoHotkey script for moving and resizing windows by learning the Windows screen coordinate system and adding the action.

Chapter Thirteen: Finding Lost Windows with AutoHotkey

"The Answer to a Question about Bringing Back a Lost Window with AutoHotkey"

While it may be rare that an open window can't be found, there are ways in AutoHotkey to locate and move all open windows.

Part III: AutoHotkey for Quick Backup of Your Work, Anywhere, Anytime!

"Learn how to create your own backup program while creating and writing text files."

One of the most important Windows tools is one that backs up your current work, even when editing on the Web. In Part III Jack writes an AutoHotkey script that not only saves data but shows you how to create and write text files.

Chapter Fourteen: Protect Against the Pain of Losing Your Computer Work

"A simple AutoHotkey script for temporary backup of new text whether on the Web or working locally."

"Suddenly your computer locks up or the power goes out for a split second. Your heart sinks as you realize that you've lost everything you were working on."

Chapter Fifteen: Incremental Backup to Protect Your Computer Work

"An AutoHotkey System for Saving Changing Versions of Your Files"

Many programs include incremental backup so you can recover earlier versions of your work. Now you can have that protection everywhere instantly with AutoHotkey.

Part IV: Collect Data in One Pot

"This handy AutoHotkey trick for copying from one window to another is a must have for anyone who collects Web information."

Not only will you learn how to quickly collect data, but a free AutoHotkey Scratchpad app by another script writer is introduced as an example of what AutoHotkey can do. It uses an INI file to save setup parameters.

Chapter Sixteen: How to Collect Web Page Text, Web Addresses, and Other Info for Research in One Pot

"With AutoHotkey you can build a tool for collecting data."

This tool for collecting text from Web pages, e-books, documents or any other source is helpful to reporters, students, business people, researchers, or anyone else who needs to work from multiple sources.

Chapter Seventeen: A Free Scratchpad App

"Scratchpad is a simple, quick, free app that makes it easy to take notes."

While you could save notes to an open Notepad window, Scratchpad by Desi Quintans has the advantage of popping up and disappearing quickly at the stroke of a key combination.

Chapter Eighteen: Instant Paste in Scratchpad and Using INI Files

"How to use an AutoHotkey instant paste technique and saving AutoHotkey app values in an INI file."

Copy quotes into Scratchpad without changing windows, plus how to use an INI file to save key parameters and variables in AutoHotkey scripts.

Part V: Quick Launch Links and Menus

"Two different AutoHotkey techniques for adding Start windows and menus to your Windows computer, plus compiling AutoHotkey scripts to EXE files."

See how to use both AutoHotkey GUI windows and the MENU command to create pop-up menus for launching your most used programs and opening favorite Web sites. Plus, adding specialized menus to the System Tray and compiling new icons.

Chapter Nineteen: Make Your Own Start Pop-up for Windows

"Add a new Start window to any version of Windows including Windows 8!"

Missing the Start Menu in Windows 8 or want to do more with any version of Windows? Now you can create your own tailored Start window for any of your Windows computers with AutoHotkey.

Chapter Twenty: A Free QuickLinks Windows App from Jack

"All Your favorite programs, Web sites and documents are only a click away."

Jack offers a simple, easy to use, free app which can replace both the Windows Taskbar and Start Menu links (even in Windows 8). Guaranteed safe. Jack should know. He wrote it.

Chapter Twenty-one: How to Build a Pop-up Menu for Programs, Web Sites and Files

"Reading files from folders to build a menu structure with AutoHotkey."

Jack discusses how his new app QuickLinks was written with a short AutoHotkey script.

Chapter Twenty-two: Tweaking the QuickLinks AutoHotkey Script

"A reader's question prompts a look at improving AutoHotkey scripts by adding QuickLinks to the System Tray icon right-click menu."

What if your keyboard doesn't have a Windows key? Here are some simple modifications to the AutoHotkey scripts. Plus compiling AHK scripts into EXE files that run on any Windows computer.

Part VI: AutoCorrect for All Windows Apps and AutoUpdating AutoHotkey on Multiple Computers

"Add the automatic correction of commonly misspelled words to all of your Windows computers."

AutoCorrect is one of the most valuable AutoHotkey apps. It works across all Windows programs and Web browsers. If you add Dropbox, then you can keep it updated on all of your Windows computers.

Chapter Twenty-three: Add AutoCorrect to Your Windows PC

"For people who fall victim to typos and common misspellings."

Microsoft Word offers AutoCorrect, but now you can add it to all of your other Windows software and Web browsing. You can even add your own personal pet peeves to the list.

Chapter Twenty-four: Adding More to AutoCorrect

"Anyone can add needed keys and other special characters to AutoCorrect."

An AutoHotkey tip for novice script writers to add that needed special character to their techniques for improving AutoCorrect.

Chapter Twenty-five: Updating Programs on Multiple Computers

"A trick for replacing program files on computers with Dropbox."

You are writing and compiling your AutoHotkey scripts on one computer, but using the compiled EXE programs on a number of other Windows computers. Here's how to use Dropbox for a no-hassle way to keep all the connected computers updated.

Part VII: Building the Reminder App Step by Step

"Starting out as a quick same-day reminder, this app slowly gets more robust."

The Reminder app is a quick way to set up a text pop-up cue with an audio reading component. Follow the evolution of this AutoHotkey script.

Chapter Twenty-six: A Cool Little Appointment Reminder for Windows

"Don't forget those scheduled meetings you can't afford to miss!"

AutoHotkey's flexibility is demonstrated with a script which sets up a personal reminder in Windows that breaks through the haze.

Chapter Twenty-seven: Lonely? Make Your Computer Talk to You!

"Add many cool features to your Windows computer with the free NirCmd utility."

NirCmd is a free command line utility which can open and close your DVD trays, hide the clock on the Taskbar, make your computer read to you, and much more.

Chapter Twenty-eight: A Talking Reminder for Windows

"Add the NirCmd Speak command to the AutoHotkey Reminder app."

Now you can make your AutoHotkey Reminder app read your appointment out loud. Plus, there's a better way to calculate time/date differences in AutoHotkey (EnvAdd).

Chapter Twenty-nine: Extending the Reminder AutoHotkey Script Beyond 24 Hours

"Add a calendar and default the date/time GUI to the current time, plus limit the reminder to future dates and time."

Why not a reminder program which schedules for more than the next 24 hours? Plus, default the GUI to the current date and limit your reminders to the future.

Chapter Thirty: Solving More AutoHotkey Reminder Script Issues

"Correcting a time calculation error, check for voice support, disabling the Escape key, and RETURN key problem."

There is another error in the time calculation, plus what happens if there is no voice support. We need to disable the Escape key, and an accidental pressing of the RETURN key dismisses the reminder.

Chapter Thirty-one: Adding Reminder Setup to System Tray Saving the Reminder to a File

"Adding setup to System Tray and how to make the Reminder app last for days with an INI file."

Corrections and enhancement for the Reminder app which teach more valuable AutoHotkey techniques, including adding Reminder setup to the System Tray icon and saving your reminder to disk.

Chapter Thirty-two: Check for Voice, Turn It On and Off, and Change the Cursor

"Is NirCmd installed?; turning voice on and off; and change the cursor to a pointing press finger."

In this wrap up of the current state of the Reminder script, features are added to make the app a little more useable.

Part VIII: Introduction to Databases, Calculating Ages, and Formatting Problems

"A quick app which calculates the age of grandkids uses an INI database and special time calculation function."

While introducing an age calculation function, an INI file is used as a database of grandchildren. The formatting and placement of controls in a GUI object is explained.

Chapter Thirty-three: A GrandKid's Age Calculating App

"Remembering your grandchildren's ages with an INI database."

Here is how to write an app with a simple AutoHotkey database to remember how old your grandkids are.

Chapter Thirty-four: The Age Calculating Function

"A function for dealing with the complications of calculating years, months and days since birth."

For the AutoHotkey obsessed user, the function for age calculation is explained.

Chapter Thirty-five: Text Formatting in GUIs

"Text positioning options in AutoHotkey GUI windows explained."

AutoHotkey has tools for positioning and sizing controls in AutoHotkey windows, but you need to understand how they work.

Index to the E-Book "Digging Deeper into AutoHotkey"

A—Index to "Digging Deeper"

B—Index to "Digging Deeper"

C—Index to "Digging Deeper"

D—Index to "Digging Deeper"

E—Index to "Digging Deeper"

F—Index to "Digging Deeper"

G—Index to "Digging Deeper"

H—Index to "Digging Deeper"

I—Index to "Digging Deeper"

K—Index to "Digging Deeper"

L—Index to "Digging Deeper"

M—Index to "Digging Deeper"

N—Index to "Digging Deeper"

O—Index to "Digging Deeper"

P—Index to "Digging Deeper"

Q—Index to "Digging Deeper"

R—Index to "Digging Deeper"

S—Index to "Digging Deeper"

T—Index to "Digging Deeper"

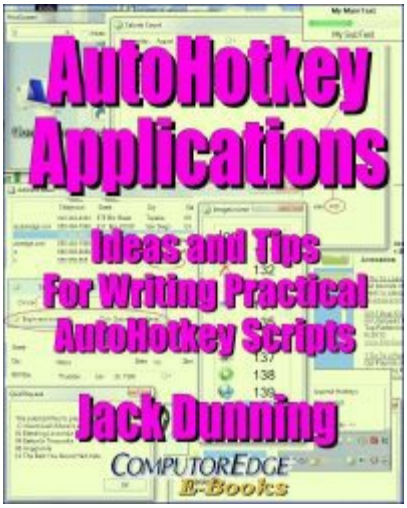
U—Index to "Digging Deeper"

V—Index to "Digging Deeper"

W—Index to "Digging Deeper"

"AutoHotkey Applications" Contents and Index

“The Table of Contents and Index from the e-book "AutoHotkey Applications."”



Jack's third AutoHotkey book is an intermediate level book of ideas and applications based primarily on the AutoHotkey GUI command. It is available at ComputorEdge E-Books in EPUB, MOBI and PDF formats. The book emphasizes practical applications. The book is not for the complete beginner since it builds on the information in the other two books. However, if a person is reasonably computer literate, they could go directly to this book for ideas and techniques without the other books. There is an extension index to the ideas and techniques covered in the back of the book. It can also be found at Amazon.

For an EPUB (iPad, NOOK, etc.) version of AutoHotkey Applications click here!

For a PDF version for printing on letter size paper for inclusion in a standard notebook of AutoHotkey Applications click here!

The Table of Contents "AutoHotkey Applications"

"ClipJump clipboard manager has a number of advantages."

Not everyone needs a Clipboard manager, but if you find that you are constantly using copy-and-paste, then ClipJump may be just the ticket for you.

Chapter Four: An AutoHotkey App for Temporary Hotkeys (HotKey GUI Control)

"Set up hotkey text for inserting into documents and Web pages."

Sometimes you just need a quick way to temporarily enter repetitious and/or long text (e-mail addresses, account numbers, etc.) into documents or forms. Here is a free, quick and dirty app that will do the job with no hassle. This script uses the Gui, Add, HotKey control and the HotKey command to create new hotkey combinations.

Chapter Five: A Quick AutoHotkey App for Playing Music (SoundPlay and FileSelectFile)

"Sometimes all you need is a barebones media player."

If you just want to quickly play a few songs in the background on your computer, here is a free, easy way to do it.

Chapter Six: A Multimedia Greeting Card (Progress/Splash Image, ComObjCreate())

"Rearrange AutoHotkey commands to create novel apps."

Ellen combines a graphic file, an audio file, and the computer voice to create this fun multimedia app.

Chapter Seven: The Perfect Soft Boiled Egg (UpDown and Progress Bar GUI Control, SoundBeep)

"The Immersible Egg Timer or a Recipe for a Barebones Windows Countdown Egg Timer"

Jack offers an AutoHotkey app that counts down to zero before setting off three alarms.

Chapter Eight: Slider App for Dimming the Computer Screen (Slider and StatusBar GUI Control)

"A Simple App That Anyone Can Write to Change the Brightness of Their Windows Computer Screen"

There are AutoHotkey functions available to control most aspects of your Windows computer. You don't need to know how they work to use them. Here is a simple AutoHotkey script called ScreenDimmer which uses one of those mysterious AutoHotkey functions.

Chapter Nine: A Daily To-Do List App (ListView GUI Control and Saving a Data File)

"Using the ListView graphic user interface control a quick and dirty to-do list app is built."

One of the most powerful AutoHotkey controls, ListView, is used to make a simple, easy to use To-Do List app. Plus, the data is saved to a simple text file.

Chapter Ten: More To-Do List App (Resizing and Positioning the GUI)

"Tips and tricks for making AutoHotkey windows easier to resize, position and use and save - and ListView editing."

Prompted by a question, Jack makes improvements to the AutoHotkey To-Do List app. The ListView columns read easier, the window is resizable, and the app saves the window's last size and position for later use. Plus, how to edit directly in the first ListView field.

Chapter Eleven: The Address Book App (ListView, Right-Click Menus, CSV Data File, and E-mail)

"A barebones address book app adds columns and inserts formatted addresses in any document and sends e-mails."

Based upon the To-Do List app built with the AutoHotkey ListView, the Address Book app takes us a few steps further with the addition of columns, use of a CSV data file format, text insertion and e-mail organization.

Chapter Twelve: Address Book App with a Formatted Input Screen, Deletion Protection, and More!

"Built on AutoHotkey ListView, the address book app gets a formatted input screen, protection against accidental deletes, and age calculation."

The Address Book app is changed for better editing, backup, and protection against accidental deletions. Some of these tricks apply to any AutoHotkey app. Plus, age calculation is added as a new feature.

Chapter Thirteen: Calorie Counting App (ListView and GroupBox GUI Controls)

"This AutoHotkey Calorie Counting app imports diet information from the Web, shows the use of GroupBox, and filters ListView by a hidden date column."

AutoHotkey is used to write a script which logs daily food intake while calculating and totaling calories. It builds on the previous To Do List and Address Book apps which use the ListView graphic user interface control.

Chapter Fourteen: Fixing the Calorie Count App (Removing Blank Lines)

"Making the Calorie Count import routine compatible with more Web browsers."

The original version of the script only worked with Google Chrome. Here's how to make the Calorie Count app compatible with more Web browsers.

Chapter Fifteen: Build Your Own Special Purpose Calculator for CalorieCount (ListView)

"These simple calorie count techniques show how to do spreadsheet-like calculations with AutoHotkey."

You could use a spreadsheet for repetitive calculations, but wouldn't it be great to have a specialized AutoHotkey pop-up calculator? The Calorie Count app shows you the basics of how to do it.

Chapter Sixteen: Manipulating Data Files in a Variable for CalorieCount (ListView, FileRead)

"Reading a data file into a variable can save time and disk access."

While it is often easy to work directly with a saved data file, reading the same file into a variable may speed up your AutoHotkey apps.

Chapter Seventeen: Building a Recipe Book with TreeView, Part I (ItemID for Tracking Content)

"AutoHotkey script development with TreeView control, plus using a variable to save a variable."

This time Jack starts the process of writing a recipe book app using the AutoHotkey TreeView control. But, rather than just giving code and explanations, Jack reveals his thought process during the script design and code writing. A data tracking technique that uses the value of a new ItemID as a variable is introduced.

Chapter Eighteen: Building a Recipe Book with TreeView, Part II (CSV File Format for Data)

"Loading the RecipeTree from a data file."

Choosing data file structure is one of the most important decisions when designing scripts. Get it wrong and the

headaches will be endless.

Chapter Nineteen: Building a Recipe Book with TreeView, Part III (Editing and Saving Data)

"Time to edit and write the RecipeTree data to a CSV file."

The next step in writing the RecipeTree app is editing data in the window and saving it to a CSV file.

Chapter Twenty: Building a Recipe Book with TreeView, Part IV (Moving Branches Up and Down)

"Moving the recipe ingredients up and down in the TreeView list."

Since the recipe ingredients should be in preparation order, we need a way to swap the branches around. Here is a trick for moving ingredients up and down the list.

Chapter Twenty-one: Building a Recipe Book with TreeView, Part V (Adding Right-click Menus)

"Adding menus to the RecipeTree script for inserting and deleting recipes and ingredients."

While there is always more that can be done, including the features for adding and deleting recipes and ingredients finally makes the app fully functional.

Chapter Twenty-two: Building a Recipe Book with TreeView, Part VI: Finishing Up

"Here are a few things to make any AutoHotkey app a little better."

While it's true that a program is never really completed, here are a few things, such as automatic backup, resizing fields with the window, saving the GUI windows size and position, showing the window with a hotkey or tray menu, checking for changes in the data, and saving on exit, which will make your app a little safer and more usable.

Chapter Twenty-three: Build Your Own AutoHotkey App Control Center (ListView with Menus)

"A basic framework for making it easier to work with various AutoHotkey (and other) programs."

If you're seeing System Tray clutter from too many AutoHotkey icons and getting confused, then here is a way to get better control of your apps.

Chapter Twenty-four: AutoHotkey App Control Center: Adding System Icons and More!

"Adding more to the AutoHotkeyControl app, plus there are 306 icons available in one Windows file."

It's not necessary to provide all of the icons for your AutoHotkey apps yourself. Here's where to look and how to use them. Plus, the simplicity of adding more menu options to the AutoHotkeyControl script.

Part II: AutoHotkey Tips and Tricks

"More AutoHotkey tips and tricks which just may answer that niggling question."

Many more miscellaneous beginning to intermediate AutoHotkey tips are offered in no particular order.

Chapter Twenty-Five: Protect Against Windows Hotkeys

"How to block dangerous unwanted Windows hotkeys, plus stripping double returns."

Windows hotkeys that you didn't know existed can cause you to lose data—and you may not even know why!

Here's how to eliminate the problem. Plus, a question about how to remove extra carriage returns from text.

Chapter Twenty-six: Working with Hotstrings as Hotkey Commands

"Hotstrings, although similar to hotkeys, give added flexibility to the auto-replacement feature."

It's easy to forget that hotstrings can do more than just add text to documents through auto-replacement. They can also run snippets of AutoHotkey code.

Chapter Twenty-seven: How to Automate Your New E-mail Messages

"Automatically add the name, e-mail address, subject, message and attachments to your new e-mails."

It's easy enough to send an e-mail attachment in Windows, but now you can add all the other information with AutoHotkey. Here are some alternative methods including how to add multiple attachments.

Chapter Twenty-eight: Fixing Broken Word Wraps

"A short script for removing embedded misplaced new line characters."

Carriage returns and line feeds are a problem when they appear in the wrong place causing formatting issues and broken word wrap. Here is an easy AutoHotkey script that fixes the problem.

Chapter Twenty-nine: A Trick to Avoid Memorizing Hotkeys with a Right-Click Menu

"It's often easier to use a right-click menu."

If you use AutoHotkey extensively then it might be difficult to remember all the hotkey combinations. Here is a quick and dirty trick for adding hotkeys to the System Tray AutoHotkey icon right-click menu.

Chapter Thirty: Adding Icons to AutoHotkey Menus (IfInString, Menu,...,Icon)

"Add Icons to any AutoHotkey Menu making it easier to use."

A long list of items in a menu can be enhanced by adding icons to help distinguish each option.

Chapter Thirty-one: Ellen Gets Help Centering Windows (Functions and Curly Brackets)

"Getting help with a useful AutoHotkey technique by understanding functions and code structure."

Even when already on the right track, sometimes all a person needs is a little push in the right direction. It helps to understand the structure of functions and the use of curly brackets.

Chapter Thirty-two: An Easier Way to Get Your Computer to Talk to You (ComObjectCreate)

"The Reminder app talks to you. Here is a less complicated method."

The development of AutoHotkey for the last few years has continued under new guidance. There are many new powerful features in the latest version AutoHotkey_L. Here is one of them.

Chapter Thirty-three: Adding a Help Window to the AutoHotkey Reminder Script (OnMessage)

"The Reminder App gets an improvement by activating the help button."

As work on the Reminder script continues, the nonfunctional Help button is made functional.

Chapter Thirty-four: Pop-up Labels for All Your Programs (ToolTip Command)

"A question about CapsLock and how to add ToolTip labels that magically appear when hovering."

When hovering a mouse over various controls in programs often help windows pop-up. Learn how to make them happen with AutoHotkey. Plus, controlling CapsLock.

Chapter Thirty-five: A Script to Change the Windows Registry (RegRead and RegWrite Command)

"Many people don't realize that AutoHotkey can modify the Windows Registry."

Some changes to software can only be made in the Windows Registry. Here's how to do it with AutoHotkey.

Chapter Thirty-six: Adding Color to ListView Rows (A_GuiEvent)

"While a little more advanced, this AutoHotkey technique controls the font and background colors of ListView rows."

Using AutoHotkey commands and functions such as DllCall(), SendMessage, and OnMessage, an advanced technique for adding color to AutoHotkey ListView rows is used to enhance the To-Do List script.

Part III: Inner Workings and Hidden Mechanisms

"The more you know about how AutoHotkey processes the AHK files, the easier it is to write scripts."

To resolve many common AutoHotkey problems it's necessary to understand how it works.

Chapter Thirty-seven: Combining Apps into One Script

"Too many AutoHotkey apps running? How to combine them into one! Learn how AutoHotkey thinks."

This may be the most important AutoHotkey chapter yet for understanding how to write and debug scripts. How AutoHotkey processes files.

Chapter Thirty-eight: Packaging Files When Compiling (FileInstall Command)

"Creating EXE files that contain all the needed files."

All the files (graphic and audio) are included in the compiled EXE file and extracted on a double-click.

Chapter Thirty-nine: Hiding the System Tray Icon and Running in the Background (NoTrayIcon)

"How to hide the System Tray icon and running a program in the background."

Too many System Tray icons? Hide a few! Plus, AutoHotkey scripts naturally run in the background, but AutoHotkey can also be used to run other programs invisibly.

Chapter Forty: Common AutoHotkey Messages and Errors

"Here are a few issues everyone should understand."

While all AutoHotkey users encounter these warnings and errors at some time, they can cause a great deal of frustration. Here is how to deal with them.

Part IV: AutoHotkey References

"Places to go for more AutoHotkey help."

There is no book that covers everything that you need. Here are many free resource which will help you on you AutoHotkey journey.

Chapter Forty-one: Free Resources for New AutoHotkey Users

"Where to get answers to Your AutoHotkey questions."

Whether new to script writing or a long-time programmer, there are free resources at the AutoHotkey Web sites.

Chapter Forty-two: Restart Learning AutoHotkey with Action Recorders

"Automatic script builders for generating AutoHotkey code."

Did you get bogged down with learning AutoHotkey? Action recorders such as AutoScriptWriter can give you a fresh start.

Index to "AutoHotkey Applications"

A—Index to "AutoHotkey Applications"

B—Index to "AutoHotkey Applications"

C—Index to "AutoHotkey Applications"

D—Index to "AutoHotkey Applications"

E—Index to "AutoHotkey Applications"

F—Index to "AutoHotkey Applications"

G—Index to "AutoHotkey Applications"

H—Index to "AutoHotkey Applications"

I—Index to "AutoHotkey Applications"

J—Index to "AutoHotkey Applications"

L—Index to "AutoHotkey Applications"

M—Index to "AutoHotkey Applications"

N—Index to "AutoHotkey Applications"

O—Index to "AutoHotkey Applications"

P—Index to "AutoHotkey Applications"

Q—Index to "AutoHotkey Applications"

R—Index to "AutoHotkey Applications"

S—Index to "AutoHotkey Applications"

T—Index to "AutoHotkey Applications"

U—Index to "AutoHotkey Applications"

V—Index to "AutoHotkey Applications"

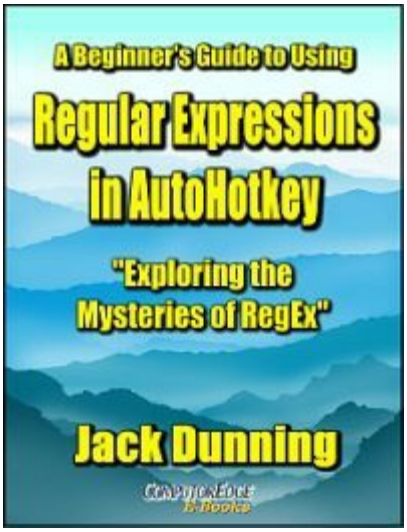
W—Index to "AutoHotkey Applications"

X—Index to "AutoHotkey Applications"

Y—Index to "AutoHotkey Applications"

"A Beginner's Guide to Using Regular Expressions in AutoHotkey" Contents and Index

"The Table of Contents and Index from the e-book "A Beginner's Guide to Using Regular Expressions in AutoHotkey."



This Beginner's Guide to Using Regular Expressions in AutoHotkey is not a beginning level AutoHotkey book, but an introduction to using Regular Expressions in AutoHotkey (or most other programming languages). To get the most from this book you should already have a basic understanding of AutoHotkey (or another programming language). Regular Expressions (RegEx) are a powerful way to search and alter documents without the limitations of most of the standard matching functions. At first, the use of RegEx can be confusing and mysterious. This book clears up the confusion with easy analogies for understanding how RegEx works and examples of practical AutoHotkey applications. "Regular Expressions in AutoHotkey" will take you to the next level in AutoHotkey scripting while adding more flexibility and power to your Windows apps. (This book is also available at Amazon.com)

Table of Contents to "A Beginner's Guide to Using Regular Expressions in AutoHotkey"

"More Regular Expression Tricks with Numbers for AutoHotkey Validation"

This time there are more simple examples of how to use RegEx functions to manipulate data in AutoHotkey.

Chapter Five: Eliminating Double Words with RegEx

"How to Use AutoHotkey RegEx to Eliminate Duplicate Words—RegExReplace()"

Digging deeper into AutoHotkey RegEx with an expression that will find and remove double words in any text, anywhere.

Chapter Six: Fixing Contractions with RegEx

"RegEx can fix multiple errors in contractions such as isn't and won't—RegExReplace()"

Another practical example of a Regular Expression in AutoHotkey with word contractions.

Chapter Seven:A Simple Beginner's Trick for Swapping Letters and Words

"An AutoHotkey Technique for Swapping the Order of Words—RegExReplace()"

Jack shows some easy AutoHotkey techniques for swapping errant letters or words, then step-by-step builds a Regular Expression (RegEx) for doing the same thing and more—with only one line of code.

Chapter Eight: A Simple Way to Find Out Where in the World That IP Address Is Located

"Find IP Addresses in E-mail, Documents and Web Pages, Then Automatically Locate Them!—RegExMatch()"

Have you ever wanted to know where that Spam is coming from or the geographic location of an IP address? This short AutoHotkey script extracts IP addresses from any selected text and downloads its world location from the Web.

Chapter Nine: Stripping Out HTML Tags

"How to extract and save the text from a web page—RegExReplace()"

Learn how to strip HTML tags from Web pages with AutoHotkey RegEx.

Chapter Ten: An App for Extracting Web Links from Web Pages

"Web Link Extractor AutoHotkey Scripts—RegExMatch()"

Need to save Web links from Web pages? Here are two AutoHotkey scripts which do the job.

Chapter Eleven: Verifying E-mail Addresses with AutoHotkey

"How e-mail address checking works with AutoHotkey RegEx."

There are plenty free Regular Expression (RegEx) examples on the Web. The problem is that they do not all work as advertised.

Chapter Twelve: Look-Ahead and Look-Behind RegEx Mysteries

"Look in front of and behind the Haystack for RegEx signposts to create a match."

A look at the confusing world of look-ahead and look-behind assertions in AutoHotkey RegEx. See how they can extend the power of Regular Expressions.

Chapter Thirteen: Using RegEx Property Symbols

"RegEx Properties \p{xx} extend the flexibility of Regular Expressions."

Use properties (\p{xx}) in AutoHotkey RegEx to correct punctuation, change currency symbols, and remove sets of brackets and parentheses.

Index to "A Beginner's Guide to Using Regular Expressions in AutoHotkey"

Regular Expressions (RegEx) A-C

Regular Expressions (RegEx) D-F

Regular Expressions (RegEx) G-L

Regular Expressions (RegEx) M-O

Regular Expressions (RegEx) P-R

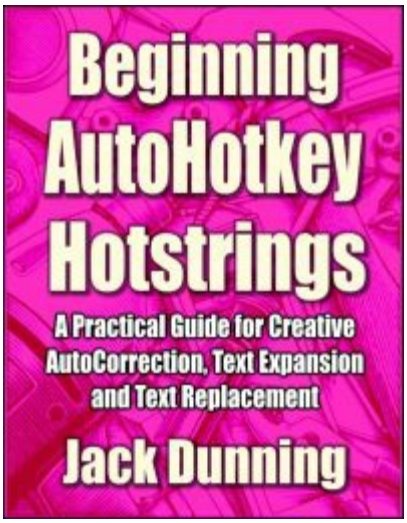
Regular Expressions (RegEx) S-T

Regular Expressions (RegEx) U-W

"Beginning AutoHotkey Hotstrings" Contents and Index

A Practical Guide for Creative Autocorrection, Text Expansion, and Text Replacement

“The Table of Contents and Index from the e-book "Beginning AutoHotkey Hotstrings.”



Hotstrings are only one piece of the extensive AutoHotkey Windows scripting language, but if you're a writer, editor, programmer, or word processing professional of any type, then you have a reason to use AutoHotkey Hotstrings. Whether you need to AutoCorrect your writing in any Windows application or on any Web site; want to quickly expand abbreviations; create menus of options for text replacement; or build lists of business or professional jargon for quick insertion, AutoHotkey is the ticket for you. The strength of AutoHotkey Hotstrings is that they are simple to implement, yet powerful tools. Plus, the software is free!

The primary focus of Beginning AutoHotkey Hotstrings is text replacement and text expansion, although almost anything that can be done with Hotkeys can be done with "action" Hotstrings. The beauty of AutoHotkey Hotstrings is that virtually anyone can learn to use them with little or no programming background. The basic Hotstring script may contain only one line of code standing on its own. The most popular use of Hotstrings is a series of these one-liners making up AutoHotkey text AutoCorrect apps—working in any Windows text editing or input field—whether a word processing program or Web page.

One of the great things about starting your AutoHotkey journey with Hotstrings is that it is some of the easiest scripting that you can do. All you need is a Windows text editor (Notepad). Then, as you add Hotstrings one line at a time, you become much more comfortable with the process—making it easier to include more advanced features. That's how this book is organized. It begins with the basics of AutoHotkey Hotstrings building on each chapter, concluding with techniques which venture into the intermediate (or even advanced) level in the last chapter.

Beginning AutoHotkey Hotstrings offers practical examples and techniques for getting the most out of this unique scripting language. If you're an experienced programmer, then you won't need this book—unless you're just looking for more ideas on how you might use AutoHotkey. The techniques are simple and straightforward while uncovering many of the more subtle possibilities.

The topics include adding menus and pop-up windows to your AutoHotkey Hotstrings, The last two chapters even explore the little known, yet powerful Input command.As a convenience for people who don’t want to dig through the Web for individual tips, but would like to learn some cool AutoHotkey Hotstring tricks, the e-book *Beginning AutoHotkey Hotstrings* is now available on the ComputorEdge E-Books site and through Amazon.

Beginning Hotstrings explores the potential of the basic AutoHotkey Hotstring option and how they can aid anyone who uses word processors, text editors, or Web input fields on Windows computers. It's surprising how this one small area of AutoHotkey can add power to your computer through Hotstring menus and the enigmatic Input commmand. For more details about *Beginning AutoHotkey Hotstrings* (Table of Contents and the entire book index), read on!

Table of Contents

Index to Beginning AutoHotkey Hotstrings

#

*

?

`

{

A — Beginning AutoHotkey Hotstrings Index

B — Beginning AutoHotkey Hotstrings Index

C — Beginning AutoHotkey Hotstrings Index

D — Beginning AutoHotkey Hotstrings Index

E — Beginning AutoHotkey Hotstrings Index

F — Beginning AutoHotkey Hotstrings Index

G — Beginning AutoHotkey Hotstrings Index

H — Beginning AutoHotkey Hotstrings Index

I — Beginning AutoHotkey Hotstrings Index

K — Beginning AutoHotkey Hotstrings Index

L — Beginning AutoHotkey Hotstrings Index

M — Beginning AutoHotkey Hotstrings Index

O — Beginning AutoHotkey Hotstrings Index

P — Beginning AutoHotkey Hotstrings Index

R — Beginning AutoHotkey Hotstrings Index

S — Beginning AutoHotkey Hotstrings Index

T — Beginning AutoHotkey Hotstrings Index

U — Beginning AutoHotkey Hotstrings Index

V — Beginning AutoHotkey Hotstrings Index

W — Beginning AutoHotkey Hotstrings Index

About the Author

“Jack Dunning”

Jack Dunning was the publisher of *ComputorEdge* Magazine. He was with the magazine since the first issue on May 16, 1983. Back then, it was called *The Byte Buyer*. His Web site is *www.computoredge.com*. He can be reached at *computoredge@gmail.com*. More information on Jack can be found at Amazon's Author Central. In March of 2015, *ComputorEdge* published its last online issue.

Jack has written hundreds of articles and columns about computers and the Internet. He now specializes in Blogs and e-books about AutoHotkey. His current list of e-book titles can be found at ComputorEdge E-Books.

Author of *A Beginner's Guide to AutoHotkey, Absolutely the Best Free Windows Utility Software Ever!: Create Power Tools for Windows XP, Windows Vista, Windows 7 and Windows 8*, *Digging Deeper into AutoHotkey*, *AutoHotkey Applications* and *A Beginner's Guide to Using Regular Expressions in AutoHotkey*. Jack helps people make their Windows computer run the way they want it to run. These books are also available in EPUB and PDF format at ComputorEdge E-Books.

More Windows Tips and Tricks e-books by Jack Dunning:

• *Hidden Windows Tools for Protecting, Problem Solving and Troubleshooting Windows 8, Windows 7, Windows Vista, and Windows XP Computers*
• *Misunderstanding Windows 8: An Introduction, Orientation, and How-to for Windows 8*
• *Getting Started with Windows 7: An Introduction, Orientation, and How-to for Using Windows 7*
• *Sticking with Windows XP...or Not? Why You Should or Why You Should Not Upgrade to Windows 7*
• *Jack's Favorite Free Windows Programs: What They Are, What They Do, and How to Get Started!*

For some really stupid computer and Internet cartoons see:
• *That Does Not Compute, Too! ComputorEdge Cartoons, Volume II: "Do You Like Windows 8 or Would You Prefer an Apple?"*
• *That Does Not Compute! Computer and Internet Cartoons from ComputorEdge Magazine*

To see all of Jack's books on Amazon, see his Author's page.

Now available *Windows 7 Secrets Four Book Bundle*

Or individually:

• Windows 7 Desktop Secrets
• Windows 7 Start Menu Secrets
• Windows 7 Explorer Secrets
• Windows 7 Taskbar Secrets

Table of Contents