Þ

A HACKER'S GUIDE TO THE UNIVERS:

WinWord TECH REF EDITION

VERSION 1.10a

~~DON'T PANIC !~~

Compiled by

Woody Leonhard
Pinecliffe International
Post Office Drawer 7337
Coal Creek Canyon
Golden, Colorado USA 80403-0100

Distribute Freely – **~~But Please Don't Modify~~**

Send Updates, Corrections, Suggestions to

Woody Leonhard
CompuServe 74730,1734 – Internet 74730.1734@COMPUSERVE.COM

Our father who art in Redmond
Hallowed be thy DOS. Thy OS/2 come
Thy version 3 be done
on RISC as it was on CISC.
Give us this day our daily update
As we give to them who report all the bugs.
Lead us not into closed systems,
But deliver us from UNIX
As we deliver those who buy our products.
For thine is the platform,
The language and the application
For ever and ever.
Amen.

Þ

Dedicated to the original WinWord Gadfly Team: Guy Gallo, Barry Simon, James "Chaos" Gleick, and Robert Enns ... without whom all of this would not have been ... er ... necessary.

This compendium is a distillation of thousands of messages on the CompuServe MSAPP forum 12. It is eclectic in that I have picked tips, hints, and warnings that make sense to me – there is no great claim for accuracy or thoroughness, and I certainly am ***not*** the author of most of the good stuff you'll find here. (I ***am***, however, most likely the author of anything you find that is wrong!) Think of the **Hacker's Guide** as a starting point, where you get to eavesdrop on a zillion conversations. Simultaneously.

> This is not a Microsoft© publication. No way, Jose. I have a lot of respect for most of the 'Softies, but the ground rules for this little handbook are simple: we're gonna tell it like it is, the best we can, even if some people get offended. No punches have been pulled. You're gonna get it pure and uncensored. You have my word on that.

If you want to contribute anything, please post a message on the CompuServe MSAPP forum, or send me a message over Internet. When you submit an item, please specify a page number in the Tech Ref so we can cross-reference. And post the item out in the open, so people can take a look at it, making comments where appropriate. I'll try to update the **Hacker's Guide** as time permits.

There just isn't enough room to credit individual contributors, and for that I apologize. But rather than clutter up the works, the info here is just presented for you to consider. Primary contributors: Guy Gallo, Barry Simon, James Gleick, lots of MS folk (some of whom may wish to remain anonymous), Ellen Nagler, and many other WinWord Gadflies and Gadflits (i.e., Gadflies in Training). I have also unabashedly stolen postings in the MSAPP forum libraries, when they appear to shed light on tough topics.

The **Hacker's Guide** starts with several pages of "The Most Common Questions" regarding WinWord itself, WordBASIC, and Adobe Type Manager, plus a few more pages of general WordBASIC macro stuff. Take a look. Might save you some time. And a few grey hairs.

Oh. Trademarks are the property of their respective Trademark holders. HarumphTM.

Woody Leonhard
Coal Creek Canyon

LOGISTICS

The **Hacker's Guide to the Univers: WinWord Tech Ref Edition, Version 1.10a** (hereafter referred to as the **Hacker's Guide**) is designed to be printed and interlaced with the pages in your Microsoft© Word for WindowsTM Technical Reference – the one with the blue cover, not the puke-green one.

If you have a "normal" printer and print driver (i.e., just about anything other than a Canon LBP or SuperPrint), you can get the **Hacker's Guide** pages printed automatically by double-clicking on the box below: (If you see something like {macrobutton ..... }, you need to turn OFF View Field Codes [click View, then Field Codes to turn off the check mark] or you need to turn OFF Show All [click the asterisk in the upper right corner].)

~~Double-click~~

The pages that print (in landscape mode) are about a half-inch shorter than the pages in the Tech Ref. They are marked with an Þ arrow for the correct point to slice 'em if you are going to tear apart your Tech Ref. You might want to make allowances if you don't intend to hack at your book, i.e., you could cut the pages just a wee bit narrower, so they don't stick out when stapled, taped, or simply layed inside the Tech Ref.

To ... uh .... literally "hack" your Tech Ref: there are a couple of tricks to slicing up a perfect-bound book. Start with a very dull, heavy knife (I used a Tillamook® cheese knife, but a Swiss® Army® Knife® would probably work as well). You need a good, heavy-duty paper slicer. And an industrial strength three-hole punch. Most of all, you'll want a heavy-duty 2-inch or 3-inch 3-ring binder, preferably one with D-rings – or a catalog binder – so the sheets will flip without binding.

Slice off the front cover of the Tech Ref. Slice off the ragged edge, down to about 7.5 inches. Three-hole punch the cover (slide it up to the top of the punch!), and put it in your binder. There. That's a start. Now separate about 15 sheets (30 pages). Bend 'em back and forth a few times. Then close 'em back up, insert your knife, and slice from the top to the middle of the page. Next, slice from the bottom to the middle. Lift out the pages, slice 'em at about 7.5 inches, snug 'em into the top of the three-hole punch, and have at it. Be sure you insert them into the binder in the right order.

Once the Tech Ref is eviscerated, it's easy to go back and insert **Hacker's Guide** pages. You might want to punch odd-numbered pages on the right, even-numbered on the left. Works pretty well. Information at your knife-tip!

Table of Contents

General WinWord Stuff

Some commonly asked WinWord questions (and a few answers)

~~CHARACTERS AND FONTS~~

Where's the bullet? It's Symbol font character #183. Pick Symbol from the ribbon. Turn NumLock ON, hold down Alt, type 0183 on the number pad, release Alt. If you use Adobe Type Manager, and you're working with an ATM soft font, you can get a very respectable bullet with Shift-Escape. And, again if you are using ATM, the Alt-0149 bullet, which is so anemic in most fonts, gets re-captured by ATM and printed as a real bullet, in spite of the fact that it displays on the screen as a lower case letter "oh".

Where's the em-dash? It's character #150 in most fonts. Turn NumLock ON, hold down Alt, type 0150 on the number pad, release Alt.

I just installed a bunch of new fonts. They show up everywhere except WinWord. What did I do wrong? Nothing. You have to do the "Printer Setup Shuffle": Crank up WinWord. Click on "File" then "Printer Setup" then "Setup". Click OK all the way back out. Look at your ribbon. There's your fonts....

How do I change my default font? When you're in WinWord, do a File Open on NORMAL.DOT. Click on "Format" then "Define Styles". The NORMAL style should be highlighted. Click "Character". Pick whatever formatting you like: we're partial to Palatino 11-point. (You can change Paragraph Formatting at the same time, by clicking on "Paragraph".) Click OK all the way back out. Do a File Close, and when asked if you want to save Global Command and Glossary Changes, click YES. Consider changing your default line spacing while you're at it – so superscripts and subscripts don't force more white space between lines. You do that by clicking on Format Define Styles again, this time click "Paragraph", and in the box that says "Spacing", where it says "Line", type in "-1" (that's a minus one).

Next time you exit WinWord, be sure to click YES when asked if you want to save global command and glossary changes. And see the reference above to NORMAL.DOT save times.

Note that changes to styles are not retroactive. If you use this tip to change your NORMAL style, it won't change any documents you've already typed. It will only change any new documents you create and any that are open at the time you change NORMAL.

I just installed a new print driver and my Symbol fonts are gone. What should I do? Here's a copy of SYMBOL.TXT, from the CompuServe MSAPP library 12. These instructions almost always solve the problem.

#: 33409 S12/Word for Windows

28-May-90 00:20:05
Sb: #33374-Windows 3.0 & W for W
Fm: Doug Timpe, Microsoft 75140,427
To: Charles Edwards 73300,3044

1. Get your CONVERSIONS disk if you're using 5 1/4" disks, your SETUP disk if you're using 720K disks.
2. Make a C:\SYMBOL (or whatever drive you want) directory. Copy the files from the \SYMBOL directory off of the floppy into this directory.
3. Copy the SFINSTAL.DIR file from the *root* of your floppy (not the \SYMBOL directory) to the root of drive C: (or whatever).
4. Copy the contents of SYMBOL.W3 into the C:\SYMBOL directory (it will overwrite all of the .PFM files -- the incoming ones are intended for Win 3.0, the overwritten ones were for Win 2.xx.)
5. Run the Control Panel. Double click Printers. A dialog box will appear. Click Configure. Another dialog box will appear. Click Setup. Another dialog box will appear. Click Fonts. Another dialog box will appear. Click Add Fonts. A message box will appear. Type in C:\ <Return>. The fonts should show up as Symbol in various point sizes on the right. Select them all and copy them over. Make sure a PCLFONTS directory is specified, OK, and then you should be all set.

Doug Timpe, Winword Development Team

(Historical notes: this is a year old – and the problem isn't solved yet; Doug posted this just after midnight <a 'Softie's work is never done!>; and especially note the "Winword Development Team", which gave rise to the "WinWord Gadfly Team".)

My Zapf Dingbats just mysteriously disappeared. WinWord says it's using Zapf Dingbats – and they flash on my screen for just a fraction of a second – but then they turn into characters in some other font. Wuh? Try this: delete WINWORD.INI (~~**NOT**~~ WIN.INI). Next time you crank up WinWord, you'll have to enter your name again, and reset your View Preferences, but that's about it. Usually your fonts will mysteriously reappear.

The em-dash and en-dash in Zapf Chancery looks lousy – almost like WinWord put together two or three hyphens to make dashes. What's happening? You've hit a couple of co-opted characters. For some unknown reason, Windows and/or WinWord take over a handful of characters and "manufacture" a new character on the fly. The en-dash is, indeed, two hyphens. The em-dash is three. Curly open double quotes are two curly open single quotes. And so on. There's a detailed discussion of these "co-opted" characters in the **Dinger** documentation (it's pretty involved, so I won't duplicate it here). Look for **Dinger** wherever you found the **Hacker's Guide**. File name is DNGR20.ZIP; it's currently in the CompuServe MSAPP forum library 12.

~~PAGE LAYOUT – FORMATTING – VIEWING~~

Why do Superscripts and Subscripts force line spacing to expand? Bad design decision. Try Format Paragraph with a negative line spacing. Using -1li (that's a "negative one line") seems to force a 12-point line spacing – even with larger point sizes.

I can't see my pictures. All I get is {IMPORT ....} gibberish. Or: all I see are these {macrobutton ....} things. Click on View, then Field Codes, to turn OFF the check mark. And double-check View Preferences, to make sure Show All is turned OFF. Finally, double-check View Preferences to make sure "Pictures" is turned ON. (Field codes are imbedded in a document. If you don't understand what they do, don't worry about it. Just keep View Field Codes OFF.)

WinWord gripes when I do an "Insert Picture", but the picture isn't available (yet) on my machine. How do I tell WinWord that the picture will be along some day now? You "unlink" the graphic. Put your cursor in the field, press Ctrl-F11, then press Ctrl-Shift-F9. Sounds like WordPerfect to you, too, huh?

How do I get words to flow around my pictures? It's easy if you follow these instructions blindly: Turn on "View Preferences Paragraph Marks". Type a Paragraph Mark (carriage return) somewhere on the page where you want the picture. Put the cursor/insertion point in front of the paragraph mark. Click on Insert Picture and insert it. Click on Format Position. Click on both of the "Page" buttons. Click Preview and **~~DON'T PANIC!~~** Click Boundaries. You'll see a gray box around your picture. Click on the edge of the box and drag the picture to wherever you want it. Double-check the "flow" by clicking on Boundaries again. Click on Utilities then Repaginate to see how the text flows. When you're done, click Cancel. Do the rest of your work with View Page ON (click View, and see if there's a check mark in front of Page; if not, click Page). You'll see the text flow around your picture.
Once you get it working, leave the paragraph mark that's next to the picture alone. Don't type any text next to it. The reason: WinWord positions paragraphs, not pictures – so all your positioning information is attached to the paragraph mark, not the picture.

Oh, and if the text flows around oddly, try changing the "Distance to Text" setting in Format Position. Also, try playing with the formatting of the paragraph mark – Point Size, Single/Double Spacing, etc.

How do I draw a box around the whole page? There was a fancy way of doing this, in the User's Reference, that involved formatting the header and footer – but I've lost it! Can anybody help? (I remember it distinctly because we went around and around on this for months, and then a brand-spanking-new WinWord user chimed in with the solution: RTFM! A humbling experience, that!) If you have a PostScript printer, follow the instructions in the User's Reference on page 21. And if you have a LaserJet III and don't want to play with it, get Guy Gallo's BorderPage macro, on the CompuServe MSAPP forum.

How do I get my Autocad drawings into WinWord? One of the options under plotter

configuration in autocad is an ADI plotter and then there are several formats, one of which is binary (from the main menu, choose 5 – Configure autocad – then 5 again – Configure Plotter – then pick the "ADI Plotter"; choose option 1 – Binary file – and you're on your own from there!). Plot the image into a file and then InsertPicture into W4W. Here, W4W shows a slightly stupid streak. It doesn't know to use the ADI filter on the file unless the file has an "ADI" extension on it, but that's minor. One monster Autocad file came up 1.9 megabytes, its DXF file is 4.9 megabytes, ADI plot file is 62.5k and W4W doc with nothing but the image is 128k so ADI is a pretty attractive way to go vs DXF. Designer just rolls over and sticks its legs in the air when it tries to read in the big DXF file. However, it was possible to paste this monster image from W4W into Designer, the first time this image made it into Designer (or anywhere in the Windows enviroment). Of course, WordPerfect had no trouble reading in the DXF file all along!

How do I get screen shots to show up as figures in WinWord? Both Hotshot and Hijaak are supposed to do an excellent job.

~~PRINTING~~

Some printing tips for Winword (from Guy Gallo):

1) Size the application Window before starting to print (that is, make WfW **not** full screen).

2) Set Printing priority to Low or Medium (not high -- counter intuitive).

3) if you have the memory (getting more of which is cheaper than getting the 25Mhz upgrade) set up a ramdrive of 1-1.5 Meg, and use it as your temp directory by issuing a SET TEMP=D:\ in Autoexec bat.

4) if you are using ATM, then generate bitmap fonts for the most frequently used fonts.

5) if you do a FileAbout in WfW and see 1.1 (and not 1.1a) call MS (in the USA, 1-800-426-9400) and order the maintenance upgrade ($10.00) to 1.1a.

Reasons:

1)If WfW takes up the entire screen then the cursor *never* stops being an hourglass while WfW is "spooling" to the temp directory. When it is sized you will notice that moving the hourglass off of WfW returns the cursor to a pointer. You can then select another application and let WfW spool in the background.

2)If you set Printing Priority to High, the Print Manager has more of a time slice than WfW, and until the documents is spooled, it doesn't matter how much priority the printer has, since there's nothing to send... That is, setting printer priority to low in print manager will give WfW more of slice in which to write the spool files, the more of a slice the faster it goes, the faster it spools the faster it finished (and the hourglass vanishes and you can go on writing the next document)...

3)if the spool files are being created on a ramdrive instead of the hard disk there is less real disk access, and things proceed faster.

4)Smaller spool disk size, faster completion.

5)1.1a speeds printing in WfW generally. If you're using an HP LJ III, I've also read that their driver version 3.7 is faster than version 3.6

The difference between WP (or any DOS word processor) and WinWord (or any Windows word processor) is that the dosWP can address the ports directly. WfW must go through the services provided by Windows.

Another important tip (this one from Woody): if you're printing multiple copies of a large document, change the "Copies" setting in File Printer Setup – and leave the File Print dialog box's number of copies at 1. Why? WinWord sends multiple copies of the document to the printer, if you use the number in the File Print dialog box. If you use the File Printer Setup value, the printer is smart enough to generate multiple copies itself.

I can't get envelopes to print worth a damn. ENVELOPE.DOT is a joke. You aren't alone! Check out Enveloper, file ENVR32.ZIP or ENVR32.EXE, wherever you found the **Hacker's Guide** (just to confuse things, it's file ENVELOP.ZIP on CompuServe's PC Magazine ZNT:UTILFORU forum, library 14); Guy Gallo's PrintEnvelope in GTOOLS.ZIP, CompuServe MSAPP forum library 12, or Paul Mayer's WinGRAB, file WINGRA.ZIP, also in MSAPP library 12.

I print a WinWord HELP screen on my LaserJet III, and I can hardly read it. Why? HELP and the LJ III don't seem to like each other. MS is working on it. In the meantime, try copying the HELP text to a document and printing the document.

When I print my document – which includes pictures – everything is fine. But when I "print" it to a file, and then use DOS to COPY that file to the printer, everything goes wrong. Why? You have to use the /B option of the copy command, e.g.,

The /B switch tells DOS COPY to copy all of the file.

How do I print a landscape page in the middle of a portrait document? Not very easily! It can be done with {print } statements and a lot of patience, but it's a bear. Easiest method: compose the landscape page outside WinWord (in, e.g., CorelDRAW) and import it as a full-page picture. Another option: get the INFOWW.ZIP file from CompuServe MSAPP forum, library 12, and look at the orientation-changing macro there. It ain't much, but it's the best we've got.....

How do I do a print/merge, skipping over blank input fields? Say you are not sure if the field "Address2" will be empty or not. This will do it:

"}{city}, {state} {zip}

The trick is to include a paragraph mark inside the quote, after {Address2}. WinWord takes you quite literally, and will only "print" the paragraph mark if {Address2} is not null.

There is a very good introduction to print/merge – CompuServe MSAPP Library 12 file PRINTME.ZIP – that is *so* good it's reprinted in a **Hacker's Guide** section all to itself. (See next section, nine pages, well worth reading.)

~~MISCELLANEOUS~~

How do YOU speed up WinWord? First, I got a 486/33 (heh, heh). Seriously, there are lots of things you can do, and most documents will run just fine on a 286/12. Whenever possible, run with View Draft checked; failing that, just don't run in View Page. If you use SmartDRV and have the memory, try:

Sometimes WINWORD.INI gets screwed up and slows people down. Just delete it. Next time you crank up WinWord, you'll have to re-enter your name and change your View Preferences settings, but that's about it. If you use Adobe Type Manager, go into ATM.INI and make sure

Finally, set up a RAMdrive in your CONFIG.SYS, and point your temp file to it with the following in AUTOEXEC.BAT (assuming your RAMdrive is D:):

Finally, remember that patience is a virtue!

Why does NORMAL.DOT take forever to save? At some point (128K?) it can no longer be FastSaved. And, man, when it's slow, it's slow! I've personally clocked it at four minutes on a slow 286.

I want to start WinWord in a directory other than C:\WINWORD. How? Easiest way: get a copy of Guy Gallo's ChooseDirectory macro, in the CompuServe MSAPP forum. It does a lot more than start you in the right directory, it lets you switch around subdirectories (and give them legible names) much more easily. If you want to go it alone: get out of WinWord, back to your desktop. Make sure that the Winword directory is part of your AUTOEXEC.BAT's PATH statement (use, say, Sysedit). Click once on the WinWord icon. Click File, then Properties. If

you want to start WinWord in the subdirectory D:\FOOBAR, type this in for the Command Line:

Windows will have a fit – tell you it's an invalid path – but ignore it. Double-click on the WinWord icon. There ya go.

How do I permanently change all document extensions from .DOC to .WFW? There is a WIN.INI setting that changes the default file extension for WinWord documents:

For a complete listing of .INI settings for WinWord, look in CompuServe MSAPP forum Library 1 or 12, file INISET.DOC.

I copy a piece of text from one window to another, and the formatting goes all haywire. Tell me about it! The **Hacker's Guide** is pieced together with a lot of cut-and-paste from MSAPP forum traffic. I often cut stuff, insert it, and SURPRISE! the formatting changes to Tms Rmn, when there's no Tms Rmn in sight. It's a bug, pure and simple: one of the oldest known WinWord bugs, and it persists in verrsion 1.10a.

Why doesn't the spellchecker recognize apostrophes? It can, if you insert items into the dictionary with a | where the apostrophe should be.

Why can't I use the "B" on the ribbon to turn stuff in a big table bold? Because the "B" will not work when your selection extends over more than 20 rows.

When my tables get to be more than one page long, the printing gets all messed up. Tables are supposed to be limited to one page in length.

Why won't the Text + Formatting ASCII export filter get my hanging indents right? It just doesn't. It should be converting hanging indents to the text file so the second line goes out to 0.5 inches. It just puts 'em in at 0 inches – i.e., no hanging indent at all. Tabs have a problem, too. If all you want to do is get your file out as ASCII, try this WordBASIC macro:

```
EditReplace .Search = "^p", .Replace= "^p^p", .WholeWord = 0, \
MatchCase = 0, .Confirm = 0, .Format = 0
```

Why do my Glossaries get all screwed up? It isn't your fault. Saving any formatting (fonts, sizes, bold, etc.) in a glossary entry is risky business: some times the formatting is applied to all Glossary entries, sometimes it's just screwed up in general. Get a copy of Guy Gallo's Glossary macros. They strip off formatting before storing things in the glossary, so you won't clobber what's already there. WinWord 1.1a may be doing a better job of leaving your Glossary entries alone: some people report that if you just save text, without paragraph marks, the problems don't seem so bad. Saving bookmarks in a glossary is an exercise in futility.

FileFind only finds some of my files! Yep. FileFind is really screwed up. WinWord, if so instructed, will scan your entire disk for .DOC files. As it finds .DOC files, it appends the directory info for those "found" files to a string that appears at the top of the FileFind Search dialog box. When that string – of all stupid things – hits 255 characters, WinWord stops searching, and doesn't tell you! Bad bug. It's in DOS Word, too, if that's any consolation.

Print/Merge in WinWord

Following is a just-ever-so-barely edited version of the file PRINTME.ZIP, in the CompuServe MSAPP forum library 12. It's from Microsoft, and it's so well done, it should be required reading for anybody who's trying print/merge. (By the way, any direct reference to bugs is entirely mine: Microsoft doesn't talk that way!)

Half the problem with print/merge is the terminology; conceptually, once you get the hang of it, it's pretty easy. If you have a choice, always turn your data into a WinWord table. It's much, much simpler than any of the other options. We'll build a working print/merge setup right here, to show you how it's done. Here's the (edited) Microsoft intro:

You must have at least two documents to do a print merge in Word for Windows. One of the documents will contain the data (e.g., names and addresses) that you want to merge. This document will be referred to as the "data document". This document, if you have a choice, should be one big WinWord table.

The other document will be your letter or address label form. This document will contain special "fields" that tell Word where to find your data and where to place the data in your letter or labels. This document will be referred to as the "main document" or "merge document".

~~CREATING THE DATA DOCUMENT:~~

The information in your data document can be separated by tabs or commas or it may be placed in a WinWord table. You may also use files that were created in Excel as well as other database applications.

Using a Word Table for the Data Document:

1. First, you need to determine how many categories of data you have. For example, do you want a separate category for first name and last name or just one category for both? Consider whether you will later want to sort the data by last name or whether you will have a "Dear" line that will use only a first or last name.

We'll build our example with seven data categories: Honorific ("Mr.", "Ms.", "Herr Dr. Prof.", whatever), Firstname, Lastname, Address, City, State, Zip.

2. Once you have settled on the number of categories, start a new document with File New, then click on Insert, then Table. In the box after "Number of Columns:" type the number of categories you will have. In our example, that's 7. Click OK.

3. You should see the empty table on your screen. If you do not see it, click View, then

Preferences, and turn ON "Table Gridlines".

4. Next you will need names for your categories of data. (They're called "bookmark names" in Word for Windows – bookmarks do lots of weird things in WinWord – but to reduce confusion we'll just call them "data names".) The names can be at most 20 characters long, may only contain letters, numbers, and the _ underline character, and must start with a letter.

Note: Certain words are reserved, so just avoid them: Author, Date, Time, Title, Subject, and Filename are the most common.

5. Put your cursor in the first "cell" of the table, type your first data name. Then hit your Tab key to move to the second cell and type your second data name. Repeat this process until you have typed last name in the last cell.

At this point, your Data Document might look like this:

Honorific
Firstname
Lastname
Address
City
State
Zip

6. When you have finished typing your last data name ("Zip" in this example), you can hit Tab again and Word will add a new row to your table. You may begin typing your data in this row. Type your data and move through your table with the Tab key just as you did in the first row of names. When you get to the end of row 2 hit Tab to start row 3, etc.

Each row of data is a "record". Do not be concerned if some of your data "wraps" within a table cell; Word will ignore this when it merges the data. You may leave blank cells in a data record. For example, you may have to skip a cell because there is no "Company_Name" entry for a particular record.

Your final Data Document might look like this:

Honorific
Firstname
Lastname
Address
City
State
Zip

Mr.
Joe

Tech
One MS Way, # 25
Redmond
WA
98052

Ms.
Jane
Tech
TechStuf
12 "A" St.
Coal Crk Canyon
CO
80403

Using Excel Files or Other Database Files as Data Documents:

Excel: You may merge data from .xls or .csv files that were created in Excel version 2.1x. WinWord is not able to merge data from Excel version 3.0 .xls files. If you have version 3.0 of Excel you will have to save your file in .csv, .txt or 2.1x format (when saving to the 2.1x format you must choose File Close in Excel rather than File Save As).

The category names in your Excel file must conform to the same conventions as WinWord category (or "bookmark") names. The row with the names must be the first row in your spreadsheet. And, the Excel document must be closed when you perform the merge.

Data documents from other database applications must be saved as text files or "comma separated value" files. As with Excel, names must conform, they must be in the first row, and the file must be closed.

Additional Information About Comma or Tab Delimited Data Documents:

When you have a comma delimited data document and your record contains a comma that is intended to be part of the data text you need to place a back slash in front of the comma or put quote marks around the data entry. (Of course, if you build your Data Document as a WinWord table, you don't run into these problems!) For example, the file:

```
Honorific,Firstname,Lastname,Address,City,State,Zip
Mr,Joe,Tech,One Microsoft Way, Bldg 25,Redmond,WA,98052
```

would produce an error because WinWord does not know that the comma between "Way" and "Bldg" was intended to be part of the data text. To correct the record, place a back slash in front of the comma or put quote marks around the whole data entry.

```
Mr.,Joe,Tech,One Microsoft Way\, Bldg 25,Redmond,WA,98052
```

Mr.,Joe,Tech,"One Microsoft Way, Bldg 25",Redmond,WA,98052

If you do not have an entry for a category in a particular record you type two commas in a row to indicate that there is no data in the category. In a tab separated data document you would hit tab twice when there is no entry. For example, if you didn't know Mr. Tech's first name, you would have a record like this:

Mr.,,Tech,"One Microsoft Way, Bldg 25",Redmond,WA,98052

Note the double commas after "Mr.".

CREATING THE MAIN DOCUMENT FOR A FORM LETTER

The main document (also called a "merge" document) will contain special fields that tell WinWord where to find your data as well as which data to merge. This document will also contain the text of your letter.

1. First you will need to tell WinWord where your data document is stored. You do this by placing a field called "Data" at the beginning of your document.

To create the Data field, press CTRL(your control key)+F9; you will see a special set of curly brackets on your screen. Type the word "data" between the brackets and then press the spacebar. After the space, put the filename of your data document. You will need to put a path name to your data document unless your data document and your main document are stored in the same directory.

If you are not sure what to type for a path name, open your data document and choose Edit Summary Info. Write down exactly what it says after "Filename:" and after "Directory:". In your Data field, type the full path name to your data document separating all directories and the filename with two back slashes.

For example, if Edit Summary Info says:

"Filename: Datadoc.doc"
"Directory: C:\Winword\Data"

your data field would say:

{data C:\\Winword\\Data\\datadoc.doc}.

2. After you create your data field you can begin constructing your letter. To put the current date in your letter, for example, move your insertion point to the place where you want the date to appear. Press CTRL+F9 to get the field brackets and then create this field: {date \@ "MMMM d, yyyy"}. (For information on setting a default date format see pages 396-97 of the Word for Windows User's Reference).

3. Now move your cursor to the line where you want the recipient's address to begin. Put each of the data names from your data document between its own set of field brackets and arrange the fields the way you want your address to appear.

For example:

{data C:\\Winword\\Data\\datadoc.doc}.

{date \@ "MMMM d, yyyy"}

{honorific} {firstname} {lastname}
{address}
{city}, {state} {zip}

You do not need to use all of the data names ("bookmarks") that are in your data document. For example, you may have included "phone_number" as one of your categories of data; you do not need to use {phone_number} in your letter.

4. You may use your data names again as often as you like, wherever you like. So now our example main (or "merge") document might look like this:

{data C:\\Winword\\Data\\datadoc.doc}.

{date \@ "MMMM d, yyyy"}

{honorific} {firstname} {lastname}
{address}
{city}, {state} {zip}

Dear {firstname},

Be the first person in {city} to take advantage of our Catatonic offer!

We have selected only you, {firstname}, to participate in the remarkable new Martian Symbiosis/Catatonia Partner Program. Send $395 in small, unmarked bills to me, and if you pass our test, we will consider you for participation in the MS/CP Program. Act now! This is a limited time offer.....

Sincerely,

Woody Leonhard, President

GadMaggot Woodmites, Inc.

~~PERFORMING THE MERGE~~

Once your data document and your main ("merge") document are complete, you are ready to merge. To merge your data with your form letter, make sure your main ("merge") document is active (just do a File Open, or click on its window if you have several open documents), click on File, then Print Merge. The Print Merge dialog box offers several options:

"All" Merges all your records
"From:/To:" Merges a selection of records, e.g., Record 1 to Record 10.
"New Document" Will merge your records to a document that will appear on your screen. The document will be called "FormletterX". Nothing will come out of your printer when you choose this option.
"Print" Will merge the records and send your form letters directly to your printer.

When the merge is done, your main ("merge") document may look like it got screwed up. DON'T PANIC! In most cases, it will look like all of your {field}s (like {firstname} and {lastname}) got overwritten by the merge. They didn't. The fields are still there. To see them, click on the asterisk up in the upper right-hand corner of your screen. See? All safe and sound. Once you have successfully performed a merge, the field codes in your main document acquire "results". The "results" are from the last record that you merged. Your field codes still exist and they will acquire new "results" when you perform another merge.

If problems occur during your merge, you may need to "toggle" the field codes in order to pinpoint problems in the main document. "Toggling" switches you from a view where you see the field codes to a view where you see the result and vice versa. You can toggle field codes by clicking on the Asterisk Button on your Ribbon.

~~TROUBLESHOOTING PRINT MERGE PROBLEMS~~

The following error messages may appear when you perform your merge. You can determine where to look for a problem by the text of the message you receive. (See information above on "toggling" field codes).

"NOT A VALID BOOKMARK NAME": This message means that WinWord didn't recognize one or more of the data ("bookmark") names that you have used to define the categories of data in your data document. Open your data document and review the words you put in the first row. Look for spaces or other illegal characters (See "Creating a Data Document", above, item #4). A blank cell in the first row of your table is an "invalid bookmark name". In a comma separated data document, a comma after your last data name will cause this error message. Word will review all the data names in your data document whether or not you use them in your main document.

"CANNOT OPEN DATA FILE": In most cases this error message means that WinWord cannot find your data document. Review the instructions under step one of "Creating the Main Document" above, to ensure that your Data field is correct – and pay particular attention to those bloody double-backslashes. Try typing in a path to your data document *even if it is stored in the same directory as your main document*.

You will also receive this error if you are using a data document from another application (like EXCEL) that is open in the other application when you perform the merge.

"BOOKMARK NOT DEFINED": You may see this error on your printed page after you perform the merge. Generally this means that the {name} you put in the merge document does not match any of the data names in your data document. Check your main ("merge") document for spelling errors and typos.

"NUMBER OF FIELDS DOES NOT MATCH NUMBER OF NAMES IN RECORD X": This error means that the number of delimiters (usually tabs or commas) in the designated record does not match the number of delimiters in the first row of the data document. This most often occurs when you have a comma delimited data document and one of your records contains a comma that is intended to be part of the data text. For example, with:

```
Honorific,Firstname,Lastname,Address,City,State,Zip
Mr.,Joe,Tech,One Microsoft Way, Bldg 25,Redmond,WA,98052
```

the comma-delimited data record should be one of these:

```
Mr.,Joe,Tech,One Microsoft Way\, Bldg 25,Redmond,WA,98052
Mr.,Joe,Tech,"One Microsoft Way, Bldg 25",Redmond,WA,98052
```

You may also get this error in a tab delimited document when you type a data entry that ends so close to a default tab that you accidentally hit tab twice to put some space between it and the next data entry. Choose View Preferences Show All to see tab arrows on your screen.

You may also get this error when you have a stray quote. For example,

```
Mr.,Joe,Tech,One Microsoft Way Apt. "C",Redmond,WA,98052
```

will drive the merge nuts. But this will work just fine:

```
Mr.,Joe,Tech,One Microsoft Way Apt. \"C\",Redmond,WA,98052
```

All of these errors can be easily avoided if you set up your data as a WinWord table. If you are exporting comma- or tab-delimited data from a database program, make sure your program tests for commas, tabs or quotes imbedded in the data, and puts a backslash in front of all of them.

"REQUESTED FIRST RECORD IS BEYOND END OF DATA FILE": In some cases, you will get this message erroneously, because of a bug in WinWord. When you get the message,

you may notice that your status bar says "Merging Record 131" (the number can also be 132 or 133). This situation has been known to occur due to erroneous file information. Possible causes include document corruption, and use of the FastSave feature used in Saving a file after several complex edits have been done. In either case, select all of your main document EXCEPT THE LAST PARAGRAPH MARK and then choose Edit Copy. (To see the paragraph marks in your document, select Show All under View Preferences). Open a new file and choose Edit Paste. Save this document and perform the merge. Your merge should work properly from the new document.

~~SUPPRESSING BLANK LINES IN A PRINT MERGE~~

Often in a merge, some of your records will not have an entry for a particular category. For example, not all of your records will have an entry for the categories "company_name" or "Address2". Using an "If" field will prevent Word from leaving a blank line when there is no entry for the category. For example:

{Firstname} {Lastname}
{If {Company_Name}<>"" "{Company_Name}
"}{Address}
{If {Address2}<>"" "{Address2}
"}{City}, {State} {Zip}

The statement means: If the entry for "Company_Name", is not equal to (<>) blank ("") then insert whatever is listed under "Company_Name" plus a line feed. If the entry for "Company_Name" is equal to blank, WinWord goes on to the Address field and does NOT print a blank line for "Company_Name".

~~SPECIAL EFFECTS~~

Character Formatting: To ensure that your merged text will appear with the character formatting that you desire use the "*charformat" field switch. Format the **name of the field in your merge document** with the desired character formatting and then place the character format switch in the field. For example: {***Company_Name*** *charformat} will give you a bold, italic company name on merge. See page 114 of the Word for Windows User's Reference for information on this switch.

Merging More Than One Record To The Same Page: The *Next* field tells Word to merge the next record to the same document rather than to a new form letter. For example:

{Name} {Ref Title} {Company_Name}
{Next}{Name} {Ref Title} {Company_Name} {Next}{Name} {Ref Title} {Company_Name}

The {Next} field is most frequently used when creating address labels.

Using a Header Document: You may keep your row of data names in a separate document from the document that contains your data records. If you do this, put both document names in your main ("merge") document's Data field with the document containing the data records listed first.

{data c:\\winword\\data.doc c:\\winword\\header.doc}

~~ADDRESS LABELS~~

WinWord provides several templates for address labels. Refer to the template descriptions on page 1 of the Word for Windows User's Reference to determine which of the templates best fits the type of labels you will use. You may need to adjust the templates to work with your labels, data document, and/or printer.

1. Once you have decided on a template, choose File New, select New Document and select the template you want to use from the list under Use Template.

2. Toggle field codes on (See "Performing the Merge"). Put the path name and file name of your data document in the Data field. Change any of the fields that do not match the data category names you set up in your data document (do not change the {next} field if there is one). Add any additional merge fields that you want to use. (Refer to the section, "Creating the Main Document", for information on the data and merge fields).

3. You may also need to adjust the layout or size of the labels. All of the label templates use tables to form the individual labels. Adjust the size of the labels by changing the Column Width and Minimum Row Height measurements under Format Table. Be sure that the whole table is selected before you make these changes. If you get a page of blank labels after one page prints, the table may be too long for the page. Try decreasing the Top and Bottom Margins.

4. If the template has a Nextif field, and you get results you do not expect, remove the field and replace it with a Next field. Eg. Replace {Nextif Name <> ""} with {Next}. Also remove the If field between the city and state fields and replace it with a comma and space. Eg. Remove {If city "" "," ""} and change to {city}, {state} {zip}.

References:

Word for Windows User's Reference, OB14328-0990, pp 1-4, 19, 114, 130-132, 397, 408, 412, 419, 421, 422.

Word for Windows Companion, Mark W. Crane. pp 580-614. Microsoft Press, 1990.

Working With Word For Windows, Russell Borland. pp 467-97. Microsoft Press, 1990.

(Woody again) – now, I don't know about you, but this little blurb restores my faith in Microsoft's ability to write documentation. It's infinitely superior to anything in the docs that come with WinWord. Kudos to whoever wrote it!

Print/merge is tough the first few times. But hang in there. It's worth the effort.

What's New in WinWord 1.10a

Faster printing of Table borders on LJ prinnters. Table borders print up to 10-15 times faster, but only on LJ printers.

Faster scrolling of Tables, by 15-25%

Faster printing of TIFF files, 50-100% faster on PostScript and LJ printers.

Faster display of pictures and bitmaps. TIFF images, in particular, display 2-5 times faster, after the first time (the first time, when WinWord retrieves the image, is the same as always).

WordPerfect 5.1 conversion. WP converters support user definable font mapping. Margins and line spacing are handled better, as are extended characters, revision marking, headers, footers, footnotes and Tables.

Support for: TIFF level 5.0 (which includes color images and various compression techniques); color PCX; EPS through the new EPS fileter (replaces a macro that was used previously); color DIBs and color DDBs, from both the clipboard and disk.

The MacWord converter supports user definable font mapping; the PC Word converter has a number of fixes and new switches.

Two converters were added to support conversion of documents to plain ASCII text, while preserving some formatting. Text + Layout files can be read into WinWord, too.

EXAMPLES.DOC has some new macros, particularly print/merge macros.

MEMO.DOT and LETTER.DOT are easier to use.

MSD.EXE comes with 1.10a; it's a diagnostic that generates reports of machine configuration and other things to help troubleshoot problems you may encounter.

General WordBASIC (Macro) Stuff

NOTE: Many of the macro examples given do NOT include Sub MAIN at the beginning and End Sub at the end. It's to save space. If there's any potential for confusion, I've usually included Sub MAIN and End Sub.

Some commonly asked WordBASIC questions (and a few answers):

I tried using the /m switch to crank up a macro when WinWord starts – but it doesn't work. Why? Not your fault. The manual is very hard to read. The trick: there's no space between the "/m" switch and the macro name. Say you want to start WinWord and immediately run the macro "Foobar". Your command line should look like this:

Note that there is no space between the "m" and the "Foobar".

I am writing a new macro, and I keep getting an Error message about a command not being available. Why? Certain commands have to operate on an underlying document: Insert is but one common example. In order for these commands to perform properly while you're editing, you have to make a document active. To do that, just click anywhere inside a document prior to STARTing or STEPping through your macro.

How can I get a .DOC – like HAKR.DOC, right here – to have macros attached to it? Good question. To see the macros attached to the Hacker's Guide, just do a File Open on the Guide, click "Macro" then "Edit" then click the button that says "Template". There ya go. Macros attached to a .DOC! The trick is simplicity itself: Do a File New and specify a Template. Save the template as, say, FOOBAR.DOT. Then go outside WinWord and rename the file FOOBAR.DOC. That's all there is to it. If you're superstitious like yours truly, you'll turn OFF FastSave, because of infrequently reported problems (see the **Hacker's Guide**, page 177). Other than that, your .DOC will work like a .DOT.

Where is the INKEY$ function? There isn't any. But you can simulate Inkey by judicious use of MacroAssignToKey. If you think of INKEY as being terminated by a Ctrl-Z, for example, you can assign a macro to pick up at the end of input, when the user hits Ctrl-Z.

How do you turn "ECHO OFF"? I.e., how do you turn off screen updates while a macro is running? You don't. You can minimize things sometimes, but that won't necessarily make things run any faster. See the **Hacker's Guide** on Page 152.

I keep getting "If without EndIf" errors. But I'm following the manual exactly. What's wrong? WordBASIC is not smart enough to tell the difference between "EndIf" (one word) and "End If" (two words). You must spell "End If" as two separate words. Of course, "ElseIf" is one word – "Else If" (two words) won't work. Go figger.

How do I access the "At:" and "Ln:" information displayed on the status bar? You don't.

How do I get at built-in dialog box values when there's no WordBASIC command to help? In general, you can use SendKeys in a very kludged-up manner. Example: here's how to find all the point sizes that appear in the ribbon for a specified font:

```
SendKeys("%P{UP 50}" + Downer$ + "{Enter}")

Dialog dlg

AvailablePoints$(i)=dlg.Points

CurrPoint$=dlg.Points

Downer$=Downer$ + "{Down}"

i=i+1
```

Why won't the Macro Editor's STEP function work with SendKeys? I dunno. But it doesn't.

I wrote a replacement for the FileOpen macro. But when I run it, the "Find" button is greyed out. What did I do wrong? Nothing. When you write a macro that calls a built-in macro, the only buttons that will work are the OK and CANCEL buttons. WordBASIC only allows two buttons per dialog box: you have to go to the Windows SDK and write a DLL (on the PDQ?) to get more than two. QED.

How do I get a list of all currently valid subdirectories into my dialog box? You don't. At least, the best way anybody has found is to shell to DOS, with "command.com /c", then redirect a DIR listing to a file, then open the file and read it. Yecch.

The Tech Ref says I can use a function name inside a function, but whenever I put it on the "right side of the equal sign", the function dies. How 'bout that!

will bomb out because X cannot appear to the right of the equals sign – and it can't appear as an argument to another function, either. Simple rule of thumb: only use the function name once, immediately before exiting the function, to assign a value to it. Anything else is risky.

I can't pass an array to a Function or a Subroutine. Well, there is a tricky way. See the discussion on Redim, **Hacker's Guide** page 242.

Why can't I run FormatParagraph or FormatCharacter while a macro is in the active window? I dunno. Doesn't make any sense. But you can't. Make a document active – even open a new file, if necessary – before running those commands.

When I have a {print ...} field at the beginning of a document, and a mixture of hard and soft-font characters, the {print ....} stuff is duplicated. Welcome to the {print "A"}<tab>B bug. It's a hard bug with no apparent easy workaround. Either avoid the {print ....} field as the first item in your document, or re-format so only hard or soft fonts appear on any single page. There have been reports that this bug may even appear when {print } is NOT the first thing in your document. Beware!

I can't find InsertPara in the Macro Edit pull-down menu, even with "Show All" checked ON. For some reason, not all WinWord commands appear in Macro Edit. My guess is that some of the ones that are "in-line", i.e., coded directly into WinWord, are not available. Why there's no dummy "Super" macro, I dunno.

WordBASIC Copy Library

Some things you can copy straight into your WordBASIC macros:

```
Para$ = Chr$(13) + Chr$(10)
LineFeed$ = Chr$(11)
Esc$ = Chr$(27)
Quote$ = Chr$(34)
SingleQuote$ = Chr$(39)
Picture$=Chr$(1)
Footnote$=Chr$(2)
FootnoteSeparator$=Chr$(3)
AnnotationMark$=Chr$(5)
Tab$=Chr$(9)
PageBreak$=Chr$(12)
SectionBreak$=Chr$(12)
ColumnBreak$=Chr$(14)
LeftFieldBracket$=Chr$(19)
RightFieldBracket$=Chr$(21)
True = - 1
False = 0

Function Swap$(InputString$, OldString$, NewString$)
'Copyright © 1991 Pinecliffe International
'Contact Woody Leonhard, CompuServe 74730,1734 before distributing
'Calling Example:  VarWithoutBs$=Swap$(VarWithBs$,"B","")
Temp$ = InputString$
index = InStr(1, Temp$, OldString$)
While index <> 0

Len(OldString$) - index + 1)

Wend
Swap$ = Temp$
End Function

Function IsNumber(InputString$)
'Copyright © 1991 Pinecliffe International
'Contact Woody Leonhard, CompuServe 74730,1734 before distributing
'Calling Example:  If IsNumber(SomeString$) Then ....
IsNumber = - 1
If Len(InputString$)=0 Then IsNumber=0
For i = 1 To Len(InputString$)
```

```
Next i
End Function


Function IsEven(n)
'Copyright © 1991 Pinecliffe International
'Contact Woody Leonhard, CompuServe 74730,1734 before distributing
'Calling Example:  If IsEven(Number) Then ....
If(Int(n / 2) * 2) = n Then IsEven = - 1 Else IsEven = 0
End Function
```

What you Can Search For

This may be the most important list in the **Hacker's Guide**. Here is what you can use for arguments in EditSearch, EditReplace, and the Search and Replace pulldown menus:

^1
^2
^3
^5
^7
^9
^10
^11
^12
^13
^14
^19
^21

If Asc(Selection$())=21 works, but While Selection$()<>"21", for some reason, doesn't.

?

Note: This will work as expected for, e.g., ?X or X?, but if you use a ? all by itself, it is the same as using ^?

^?

^w
^t
^p
^n
^d
^s
^-
^~
^^
^m
^c
^nnn
^0nnn

General ATM Stuff

Some commonly asked Adobe Type Manager questions (and a few answers):

The flicker with ATM in WinWord is driving me nuts. What can I do about it? You can use the ATM Foundry to generate screen bitmap fonts for your most commonly used fonts. Or you can increase the ScanBufSize in ATM.INI to 16. You might also try Aliasing both Tms Rmn and Roman to the ATM font Times. See the ATM manual, page 13, for info on Aliasing. Make sure you do the "Printer Setup Shuffle" (see page 5) after you change any ATM fonts.

Can I change the WinWord system font? Yes, if you use ATM. WinWord will use the first sans-serif proportional font listed in ATM.INI as its system font (i.e., the font that appears in the ribbon, ruler, status bar, etc.). You can also create an Alias for Helvetica (which is the font normally acquired as the system font), using the instructions on page 13 of the ATM manual.

Can I change the font WinWord uses for macros? Yes again. Create an alias for Tms Rmn, following the instructions on page 13 of the ATM manual. Remember the "Printer Setup Shuffle"!

Page 65

GoTo Label

A label may be preceded by at most one space, and followed by at most one space, then a colon. If you put a tab in front of a label it will not be recognized.

Page 69

While....Wend

A typo. It should say:

```
Count = Count + 1

EditSearch "macro", 0, 0, 2
```

User-Defined Functions

The name of a function may not be used as an argument inside the function. For example:

will crash.

Dialog Box Definition Statements

Dialog boxes are limited to 512 characters (including variable data – so a dialog box can die unexpectedly). They are also limited to 32 "items" where most statements are one "item", but ListBoxes count as two, and ComboBoxes count as three.

Begin Dialog UserDialog x, y, dx, dy

x seems to be limited <= 510

ComboBox

That should read ComboBox x, y, dx, dy, ArrayVariable$(), .Result$

It isn't a .Field -- it's just the .Result$, i.e., the thing the user clicks on, returned as a text string. Ignore the mumbo-jumbo.

The () parenthesis on ArrayVariable$() are required. Dumb.

A multi-dimensional array used as the ArrayVarriable$ argument will only display the first dimension. For example,

will only show the first-dimensioned ten items of MyArray$, i.e., MyArray$(0 thru 10, 0, 0) – I think.

GroupBox

That .Text$ variable is required – even if you're just using GroupBox to draw a pretty rectangle in your dialog box.

ListBox

That should read ListBox x, y, dx, dy, ArrayVariable$(), .Index

It isn't a .Field -- it's just a plain, old .Index into ArrayVariable$. Ignore the mumbo-jumbo.

The () parenthesis on ArrayVariable$() are required. Dumb.

A multi-dimensional array used as the ArrayVarriable$ argument will only display the first dimension. For example,

will only show the first-dimensioned ten items of MyArray$, i.e., MyArray$(0 thru 10, 0, 0) – I think.

OKButton

Esthetically pleasing settings: dx=65, dy=20

OptionGroup and OptionButton

The .Text$ field is required on OptionButtons, even if you are only using the button to set up a fancy TextBox.

(A fancy TextBox is one that has an OptionButton right in front of it, so the user chooses the option, and specifies text in the same dialog box.)

Text$ discussion.....

If you want an ampersand to show up you must double it. I.e., &A will display as A But &&A will display as &A

Text x, y, dx, dy, Text$

Text$ is limited to 255 characters

TextBox x, y, dx, dy, .Field

First, make that a .Field$ It's always a string variable. And it cannot be dimensioned. For example,

will bomb out. Why? Lousy design decision. You'll find some macros (including several of mine) with line after line after line assigning values to undimensioned variables just to handle this dumb requirement.

The Tech Ref is ridiculously confusing about TextBox. All it does is display .Field$, let the user modify it, and return the modified results back to your macro.

In WinWord 1.0, .Field$ was "poured" into the defined area, starting in the upper left corner. But in version 1.1, 1.10a, etc., .Field$ is "poured" into the defined area, centered from top to bottom. Yecch. Another "improvement" in 1.1: Newline and Paragraph marks do not force line feeds any more. Really screwed up one of the early versions of Enveloper with this design decision.....

The user cannot type in a Shift+Enter – or any other paragraph mark – so you cannot use TextBox for a lot of things. I don't think Alt-0027 will come through either. Check on other Alt-codes to see if they will work before you use TextBox in any situation that requires anything beyond ANSI Windows characters. It's really too bad, too, because TextBox – if it were implemented properly – could be a very powerful command!

.Field$ is limited to 255 characters. But that usually doesn't matter much, because the number of characters in .Field$ "count" against the dialog box max of 512. That can give you runtime errors – Dialog Box Description Too Complex – if you aren't careful.

If you want an ampersand to show up you must double it. I.e., &A will display as A But &&A will display as &A

Generally you will want a dialog box's "cursor" to go to a TextBox first. If you make a TextBox the first item in the UserDialog, the "cursor" will go there, and highlight any default text you may have inserted.

Page 110

Using Trace

This Tech Ref explanation needs a little editing. Try putting a big paragraph mark after the first sentence.

From the second sentence on, the writer is talking about Start, Continue, Step, and StepSub, in addition to Trace.

StringTime$()

Huh? Never got that one to work.

Activate

AppActivate

WindowText$ is limited to 255 characters.

ChDir

Name$ is limited to a maximum of 255 characters.

I have a note that there were some problems with specifying odd drive designators, but a bunch of people claim it ain't so. Let me know if you encounter anything.

ComboBox

That should read ComboBox x, y, dx, dy, ArrayVariable$(), .Result$

It isn't a .Field -- it's just the .Result$, i.e., the thing the user clicks on, returned as a text string. Ignore the mumbo-jumbo.

The () parenthesis on ArrayVariable$() are required. Dumb.

A multi-dimensional array used as the ArrayVarriable$ argument will only display the first dimension. For example,

will only show the first-dimensioned ten items of MyArray$, i.e., MyArray$(0 thru 10, 0, 0) – I think.

ContinueMacro

Another undocumented WinWord command.

Function unknown.

Page 146

CountMenuItems

Yet another undocumented WinWord command, courtesy of Graham Ullrich.

Exact function unknown.

Page 133

Begin Dialog UserDialog x, y, dx, dy

x seems to be limited <= 510

Dialog boxes are limited to 512 characters (including variable data – so a dialog box can die unexpectedly). They are also limited to 32 "items" where most statements are one "item", but ListBoxes count as two, and ComboBoxes count as three.

DDE – A hodgepodge of DDE tips

Most strings in DDE commands are limited to 255 characters in length.

To crank up EXCEL from inside WinWord:

To crank up WinWord, maximize it, and execute FileNew from inside EXCEL:

To crank up EXCEL, put the formula "=B1" in the EXCEL cell named "test", from inside WinWord (assuming SHEET1.XLS is in C:\WINDOWS, and there is already a cell in SHEET1 called "test"):

")]"

If you paste a link from WinWord to an EXCEL cell, and you want show the cell's value in a sentence, insert the cell, turn View Field Codes ON, and you'll see something like this:

add the formatting instruction \#### like this:

To increase the DDE Timeout time, add a line to WIN.INI that says:

You can send Excel any command listed in the function reference for macros. Notice that double quotes can be imbedded using + and Chr$(34). Notice also, the square brackets ([]) that each command is contained in.

For example, to print a message on the message area send:

DDEExecute(ChanNum, "[message(1,"+Chr$(34)+"hello world"+Chr$(34)+")]")

"Message" is documented on page 151 of Excel 3.0 Function Reference.

DeleteFile

See "Kill", page 211.

Page 151

DisableAutoMacros

Another undocumented WinWord command. Sounds dangerous. Anybody willing to play with it?

DocMove

There is no "ECHO OFF" to turn off WordBASIC screen updates. But some DocMove things will help. Sometimes. Suggestions:

Use Super DocSize 10,10 to make the doc window as small as possible; then use DocMaximize to restore it.

Use DocMove 300,300 to move the doc window out of sight while the macro is running, then use DocMove 0,0 to bring it back.

EditGlossary

If somebody figures out glossaries, definitively, please let me know! I avoid them whenever possible 'cuz they're so flakey.....

If you have to use a Glossary, get Guy Gallo's glossary tools, ExpandGlossary and EditGlossary. They replace the WinWord macros and strip formatting before inserting things into the Glossary, thus avoiding endless confusion.

You can GetGlossary and SetGlossary while a macro is the active window – but you can't EditGlossary .Delete. Why? Dunno...

If you have a bookmark in a glossary, it may get screwed up. Save two different glossary entries with the same bookmark name, and one or the other gets clobbered. Easy solution: forget about storing bookmarked items in the glossary.

To work with Glossaries, mess them up a bit, then restore them to their original condition – AND avoid having the infamous "Save Global Glossary and Command Changes" pop up on exiting WinWord, see the discussion in the **Hacker's Guide** under "IsDirty", page 209.

Here's the ultimate Hacker's trick for Glossaries: If you press Ctrl-space before you put a selection in the Gallery, it gets stored without any font name. If you then insert this fontless entry into your text, WinWord applies the font name that is stored in an entry near the end of NORMAL.DOT. You can actually go in and fiddle with that entry, with a hex editor. You can change it to any font name, as long as it's shorter than the current entry – the name is terminated with a zero byte.

EditGoTo

The Edit Goto command allows you to jump to a specific page. If a number or "p" plus a number (e.g., P3) is entered, Word for Windows will jump to the physical page.

In order to jump to the page that is numbered as 3 (if page 3 is not physically page 3, i.e. page numbering starts on 2) - indicate the page and section number.

For Example:

In a file that is a chapter of a book with a starting page number of 66, to jump to page 69, select Edit GoTo and enter "S1P69"(section 1, page 69). This will take the cursor to the correct page, the physical page 3. Typing "p69", or "69" into the Edit GoTo dialog box would take the cursor to the end of the document since it is looking for a physical page 69.

EditHeaderFooter

This only "opens the header or footer pane for editing" if View Page is turned OFF. If View Page is ON, it will kick you up or down to the header or footer on the current page.

Use in conjunction with ClosePane to edit headers and footers with View Page OFF.

Those weird [$] signs on StartingNum, HeaderDistance and FooterDistance mean that you can supply values as either strings or numbers. For example, both of these are identical:

Note that, in both cases, the "$" does NOT appear on StartingNum.

Default header distance is one-half inch, at least on my system.

HeaderDistance and FooterDistance may both be specified in inches, e.g., .HeaderDistance="1 in" or .HeaderDistance="1" + Chr$(34).

EditHeaderFooterLink

The mysterious "Link" they are talking about here is basically just page numbering. If you set up a section with separate page numbering, then change your mind and want to go back to "normal" page numbering, you run an EditHeaderFooterLink. That copies in the text from the previous header or footer, deleting the "new page numbering" info.

If you want to delete everything in a header or footer and unlink it – your best bet is to leave something (like a space) in the header or footer. Otherwise it won't unlink.

I've had trouble running this guy with View Page ON.

EditPic

This is an undocumented WordBASIC command. Function: who knows?

EditReplace

This just starts at the current cursor location, and heads toward the end of the document, replacing as it goes. If you use .Confirm=0 and you want to replace all the occurrences of a string, make very, very sure you start at StartOfDocument, or use SendKeys to force EditReplace to loop all the way through the document, i.e.,

Be leery of doing a lot of replacing from inside your own home-grown macro. The problem is the "Undo" buffer: WinWord only allocates a small amount of memory to keep track of changes (so you can "Undo" all the changes with one click). If you have a bunch of EditReplace "hits", the Undo buffer can fill very quickly, and your macro will simply crash and burn with a message advising the user to save changes and exit WinWord. (As I recall, this message comes up without tripping the usual WinWord "On Error" conditions.) Doesn't matter how much memory you have on the machine – WinWord isn't smart enough to use all of it.

The Search$ and Replace$ strings in EditReplace are limited to 255 characters. If you need to replace something larger than 255 characters, try this:

EditClear

Insert "Thestringlongerthan255characters"

It's particularly important to keep the 255-character limitation in mind if you are automatically selecting text elsewhere, and don't know offhand if it can exceed 255 bytes.

EditSearch

James Gleick reports that he has encountered situations where an EditSearch in a long document hangs the system – the hourglass goes on and won't go off.

What the Tech Ref is talking about when it says "with prompt" or "without prompt" is the "Reached End of Document, Continue at Beginning?" prompt.

I have encountered situations where the default .Direction is NOT the last .Direction used. Better to specify it each time.

Page 163

EditSummaryInfo

Several places it says ".Comments$". It should be ".Comment$", without the "s".

The .NumChars argument counts characters in headers, footers, footnotes, and annotations. The reason why this number is so different from the "FileSave" message in the Status Bar? The FileSave character counter only counts characters in the main part of the document.

For some reason, NumPages is listed as numeric. It isn't. You need to use, e.g.,

NumWords and NumChars probably work the same way.

ExtendSelection

If this command looks a little strange, it's because it does about a dozen different things depending on the context.

Here's an example of ExtendSelection. It searches for a caret, turns on ExtendSelection, searches for the next caret, and clears out everything between the carets:

```
ExtendSelection

EditSearch .Search="^", .Direction=2

EditClear

EditSearch .Search="^", .Direction=2
```

ExitWindows

An undocumented WinWord command. And it works. Oh boy, *does* it!

FileFind

SearchList$ is limited to 255 characters.

This can cause real, live bugs in the operation of FileFind. Avoid it if you can! Here's why:

WinWord, if so instructed, will scan your entire disk for .DOC files. As it finds .DOC files, it appends the directory info for those "found" files to a string that appears at the top of the FileFind Search dialog box. When that string – of all stupid things – hits 255 characters, WinWord stops searching, and doesn't tell you! Bad bug. It's in DOS Word, too, if that's any consolation.

FileNew

Note that there is a bug in FileNew. If you have a macro that runs the built-in function FileNew, and the user clicks "CANCEL" in the built-in FileNew, the "CANCEL" cannot be trapped by an On Error statement. Every other built-in function will let you trap "CANCEL" with an On Error.

FileOpen

If you want to open all files as text – bypassing the "convert file from" dialog box – you need to use the SetConversionState macro or manually edit the WIN.INI file to set the conversion state to NO.

When opening text files or foreign formatted files, Word for Windows will normally prompt with a dialog asking for the file format to convert from. In order to bypass the prompt and have Word for Windows open all foreign formatted files as text files, the conversion state in the WIN.INI needs to be set to "No" or "Conversion=No".

A macro to change the conversion state in the WIN.INI file, is included in the Word for Windows EXAMPLES.DOC file. This macro is called SetConversionState, and can be installed by opening the EXAMPLES.DOC file and clicking on the Install button.

Once installed, choose SetConversionState from the Macro menu to enable or disable the conversion of foreign file formats. When the file format conversion is disabled, all foreign files will load as text files.

The conversion state can also be set manually by editing the Microsoft Word section of the WIN.INI file.

The above lines will disable the file format dialog when importing foreign formatted files, so that all files load as text.

This setting will enable the file format conversion prompt when opening foreign formatted files.

Pages 173-4

FilePrint

This is one of the quirkiest functions in WordBASIC. Be ready for lots of surprises.

If you're going to print a long document, there are a couple of tricks.

First, you should never run WinWord maximized when you're printing a long document. Why? The hourglass will stare at you. Make WinWord just a little smaller than full-window, and you'll be able to click outside of WinWord, and at least play a round of Solitaire while you're waiting for the printer.

Second, you can automatically resize WinWord by changing your FilePrint macro to do this:

Third, PrintManager won't take more than 20 "jobs". If you run FilePrint more than 20 times without any pages printing, the PrintManager will object – saying it can't take any more – but WinWord will keep on plugging out pages. If WinWord is running in a full window, there's nothing you can do but keep clicking OK at the PrintManager's admonitions.

Printing a range of pages is quite an art, if you have multiple sections that restart the page number. The only tricks I've found to work is to either go through the document and wipe out the "Start with new page number" part of all section breaks, or to somehow futz with the .From$ and .To$ settings.

Valid .From$ and .To$ settings include "p3s2" or "3s2", both indicating page 3 of section 2.

Oh – selecting an entire page, then printing it will print the headers and footers of the first page of that section, not the headers and footers of the selected page.

Bug alert: If you have EPS file support, and you run FilePrint but cancel it, your ruler will change from inches (or whatever) to points! That's a world-class weird bug. To fix it, edit your FilePrint macro, add one line after "Bye:"

NB.: this only happens with EPS support installed.

There is a problem printing with "mixed" bins on the IIID. I think it also happens with the IIISi and IID, but haven't been able to confirm it.

The problem: "mixed" bin selection works right the first time (i.e., FilePrint .PaperFeed=7). But it doesn't work right the second (or subsequent) time.

Solution: use this instead of FilePrint .PaperFeed=7

(Notes: That's a capital letter "Oh", there is no % in front of the "m", and the curly brackets around the "Enter" are just typed at your keyboard – no insert field funnies.)

This little SendKeys kludge manually selects the Option button, then picks "mixed". If you manually select "mixed" this way every time, the printer will print mixed.

I warned you that FilePrint is quirky!

FilePrinterSetup

Be very, very cautious when writing SendKeys parameters for FilePrinterSetup. It's a jungle. You CANNOT assume that if your SendKeys works on all the major printers, it will necessarily work on all the lesser printers. And SuperPrint screws everything up.

Caveat Hacker!

The FilePrinterSetup discussion should be prefaced by a classic remark:

Abandon hope, all ye who enter here.....

When you embark on FilePrinterSetup stuff, you are indeed entering the first ring of Hell.

FilePrintPreview

When FilePrintPreview is on, you cannot use a macro to work with bookmarks. Why? I dunno.

There has been a lot of discussion about automatically maximizing the WinWord screen before running FilePrintPreview. The problem stems from the odd way Winword handles AppMaximize. (And I'm sure it doesn't help that the normal way of leaving FilePrintPreview mode is to CANCEL!) Very strange behavior. Here's the latest I have, by way of a solution:

That's a real kludge, because you can't DO anything in FilePrintPreview, and it changes the way FilePrintPreview works (i.e., it requires you to push enter to get out, in spite of the beautiful CANCEL button up at the top). Yecch.

Another alternative:

Which isn't much better.

Oh – there's a typo in **FilePrintPreview$()**. The example should say:

'cuz it returns an integer.

FileSaveAs

WinWord normally saves all documents in "Fast Save" format. Older versions of Grammatik, among other programs, have trouble dealing with Fast Save format files. (Selecting Create Backup automatically disables the Fast Save option.)

If you change your FileSaveAs macro to:

You'll disable Fast Save.

If you want to have a macro save the active document under a new name but skip the Summary Info dialog box – even when Customize has "Prompt for Summary Info" – here's how to do it

Files$()

FileSpec$ is limited to a max of 255 characters.

Page 178

Font

Name$ is limited to a maximum of 255 characters.

FormatCharacter

The weird [$] thing that follows the Points, Position, and Spacing arguments just mean that you can use either character or numeric values. For example, these are identical:

Note that the "$" is never attached to the argument name.

There's a weird bug in FormatCharacter. Unless you change the arguments, it always comes up with a "0 point Superscript". That just happens to be the same thing as "normal", but try explaining that one to a novice! Easy solution:

Hey, it works.....

The weird [$] thing that follows all of the arguments (!) just means that you can use either character or numeric values. For example, these are identical (if your unit of measurement is inches):

Note that the "$" is never attached to the argument name.

FormatParagraph

Note that FormatParagraph affects the white space surrounding "Format Position" positioned pictures and paragraphs.

Note also that FormatParagraph can affect cell height in tables.

The weird [$] thing that follows most of the arguments just means that you can use either character or numeric values. For example, these are identical (if your unit of measurement is inches):

Note that the "$" is never attached to the argument name.

FormatPicture

The weird [$] thing that follows the Crop arguments just means that you can use either character or numeric values. For example, these are identical (if your unit of measurement is inches):

Note that the "$" is never attached to the argument name.

FormatPosition

See the **Hacker's Guide** "Most Commonly Asked WinWord Questions" for a step-by-step approach to positioning a picture. The same procedure works for other paragraphs, too.

When a paragraph contains only a picture, Format Position works as one would expect: the paragraph is "cropped" (for lack of a better term) to the size of the picture, to facilitate moving it around in Page Preview.

But when a paragraph contains text (or a dingbat), it is considered to extend to the full, pre-defined width of the paragraph (which defaults to six inches). That's why people working with Drop Caps have to do a Format Paragraph to trim the para (in this case of Drop Caps, the entire paragraph is just one letter) down to size, before moving it around in Page Preview.

The paragraph width in Format Position needs to be set to the width of the drop cap or character (e.g., 0.5 inches), in order to get text beside the letter. Hopefully drop caps will be a little more straight forward in the future. For now, set the paragraph width in Format Position.

The weird [$] thing that follows the Horizontal, Vertical and ParagraphWidth arguments just means that you can use either character or numeric values. For example, these are identical (if your unit of measurement is inches):

Note that the "$" is never attached to the argument name.

FormatSection

When choosing Format Section and selecting Line Numbers, the line numbers will be clipped if you have a left margin < 0.62 inches. Minimum left margin for two-digit line numbers is 0.62 inches. Minimum for three-digit numbers is 0.72 inches. Minimum for four-digit line numbers is 0.82 inches.

The weird [$] thing that follows several of the arguments just means that you can use either character or numeric values. For example, these are identical (if your unit of measurement is inches):

Note that the "$" is never attached to the argument name.

FormatTable

The weird [$] thing that follows several of the arguments just means that you can use either character or numeric values. For example, these are identical (if your unit of measurement is inches):

Note that the "$" is never attached to the argument name.

GroupBox

That .Text$ variable is required – even if you're just using GroupBox to draw a pretty rectangle in your dialog box.

InputBox$()

Prompt$, Title$ and Default$ are all limited to a total of 255 characters.

I don't know if the variable itself (equivalently, the input text) is also limited to 255 characters; try it if there's any chance somebody will type in that much.

The Gadflies had fun with this one a while back.

```
x$="&Ampersand"

SetProfileString("Microsoft Word", "Test", x$)
```

Want to watch WinWord go hyperexponential? Ah, the chaos.....

InsertBookmark

Bookmarks are not allowed in headers or footers.

The name of a bookmark must be contain letters, numbers, or the _underline character; max 20 characters, and the first character must be alpha. (I.e., don't start a bookmark name with a number!)

Page 201

InsertBreak

This one's a little tough to explain.....

Say you're in a section. And that section has a specific starting page number. Let's call the starting page number "n". So far so good?

Now. Say you insert a section break inside that section. The page numbering is a little strange. The new section will start at page "n", no matter where in the section the break is inserted.

Confusing? Try debugging it!

Page 205

InsertIndex

Take a look at the {index} field, in the **Hacker's Guide** page 359. There's all sorts of fun stuff that can go bump in the night – most notably what happens when you have so many page numbers that they won't all fit on one line.

InStr()

An important typo.

It should read

Note that if you omit the [Index], you must omit the comma as well!

IsDirty()

Okay. I'm gonna step you through how to shoot yourself in the foot. If you misuse it, don't go blamin' me!

You can change Glossary entries – or anything else stored in NORMAL.DOT, for that matter – and still have your macro leave NORMAL.DOT "not dirty". (Why would you want to do that? To save your users from spending hours watching "Saving NORMAL.DOT 0% Complete".) Watch out. If you start muckin' around with IsDirty, you can clobber **intentional** mods to NORMAL.DOT.

After you're through doing whatever you want to do to NORMAL.DOT – add and delete a Glossary entry, for example – just run these two lines from a TEMPLATE macro:

You may have to add subdirectory info to "normal.dot".

That's a tip from the bleedin' edge, believe me.....

KeyCode

This is another undocumented WinWord command. Function unknown.

KeyMacro$

This one is undocumented, too! Function unkown as well.

LineDown()

LineDown() is broke and needs fixin'. Bad. Don't use it unless you have absolutely no other choice.

LineDown(n), where n > 1, will return a FALSE if NO LineDown is possible. But it will return a TRUE if ANY LineDown is possible.

Say what?

Okay. Imagine your cursor is on line 2 of a three-line document. Your macro says:

Even though WinWord can't go down five lines – there's only one line left in the document – you'll still get a TRUE test, i.e., "HeeHee. Gotcha!"

Also watch out for an infinite loop testing LineDown() when the last line of a document is selected. I'm trying to figure this one out, myself, right now.

ListBox

That should read ListBox x, y, dx, dy, ArrayVariable$(), .Index

It isn't a .Field -- it's just a plain, old .Index into ArrayVariable$. Ignore the mumbo-jumbo.

The () parenthesis on ArrayVariable$() are required. Dumb.

A multi-dimensional array used as the ArrayVarriable$ argument will only display the first dimension. For example,

will only show the first-dimensioned ten items of MyArray$, i.e., MyArray$(0 thru 10, 0, 0) – I think.

MacroAssignToKey

To well and truly unassign an assigned key, you have to issue the command twice!

For some reason, Microsoft didn't document any but a small portion of the key codes. Here's the list I've compiled, after some exhaustive tests. Any code that is not on this list is not implemented – at least on my system (with a Northgate Omnikey Plus keyboard). And note that some keyboards map things a little differently.

For the key assignments where Guy and I didn't get the same results, extreme caution is urged. I dunno what's going on.....

Note that these codes only apply to the MacroAssignToKey command. If you try to use about half these suckers from the Macro AssignToKey drop-down menu, you're out of luck.

3
8
9
12
13
27
32
33
34
35
36
37
38
39
40
45
46
47
48
49
50
51
52
53
54
55
56

57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
96
97
98
99
100
101
102
103
104
105
106
107
108

109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
186
187
188
189
190
191
192
219
220
221
222

Page 220

MacroDesc$

This is an undocumented – but very much present! – WinWord command. It's apparently related to the description displayed in the status bar when you select a macro; why it's undocumented, and how it's used, I dunno.

MacroRecord

There appears to be no way to turn the macro recorder off from inside WordBASIC.

You can pause it, however. Refer to PauseRecord.

MacroRun

(Is this command ever necessary?)

When writing macros, you will find that using Macro Run will consume much more memory than when using one macro with many subroutines. There is much more overhead (memory used) as a result of doing "macro run" versus calling a subroutine.

If you execute MacroRun of a macro stored in a template and the active window is not associated with that template, the driver macro simply stops.

One solution: save the current window name and Activate the template window before executing MacroRun. On return, Activate the saved window name:

```
Sub TemplateMacroRun(TemplateWindow$, MacroName$, XSaveAll)
    SaveWindowName$ = WindowName$()
    Activate TemplateWindow$
    MacroRun MacroName$, XSaveAll
    Activate SaveWindowName$
End Sub
```

It works pretty well unless the macro you execute closes the previously active window. This method also allows you to execute macros from a number of different templates.

MsgBox

MsgBox()

Maximum Message$ and Title$ length is 255 characters.

The "Attention icon" is a big exclamation point.

The "Information icon" is a lower-case "i" with a circle around it.

Quite some time ago – when I was just learning WordBASIC – I tried negative values for Type. They didn't work for me back then. It's easier to use Print anyway.

Note that the cancel button here is fundamentally different from the one in a Dialog Box. If the user clicks CANCEL in a MsgBox, no OnError condition is raised. Rather, you have to test the returned value of MsgBox() to see which button was clicked.

NextCell

NextCell not only moves the cursor to the next cell, it selects the contents of the next cell as well. It's a bit strange because it doesn't work like all the other functions -LineDown, CharRight, that sort of stuff. And there's no CellDown or CellUp.

Anyway, I needed to test for an empty cell. Tried all sorts of things, but this finally worked (!).....

```
NextCell
If Selection$()=Chr$(13)+Chr$(7) Then Goto EmptyCell
```

One more note: beware! EditGoto "\Cell" does NOT seem to give the same results as NextCell. The EditGoto picks up the cell marker.

OnTime

This section of the Tech Ref desperately needs re-writing.

When$ can be in one of two formats, either "12:34" or "12:34:56". You must have the colon in the right place. You must have precisely two digits for the hour, two for the minute, and (if you use seconds), two for the second.

It's all on a military-style 24-hour clock. So, 8:15 a.m. is "08:15", and 8:15 p.m. is "20:15"; two minutes past midnight is "00:02". WARNING: there was an autobackup macro circulating at one point that failed to "flop over" from 23:59 to 00:00 – it went to 24:00.

(Guess that's what you get for running autobackups between midnight and 1 am!)

To repeat the Tech Ref, for emphasis – you can have just ONE OnTime macro running. If you start another OnTime macro, the first one is clobbered completely, without any warning.

When$ and Name$ are limited to 255 characters.

Open

Name$ is limited to a maximum of 255 characters.

An example:

OptionGroup and OptionButton

The .Text$ field is required on OptionButtons, even if you are only using the button to set up a fancy TextBox.

(A fancy TextBox is one that has an OptionButton right in front of it, so the user chooses the option, and specifies text in the same dialog box.)

Print

Expression is limited to a total of 255 characters.

This Print statement works like most other BASIC print statements, e.g., there is a carriage return appended to the end of each print Expression unless the Expression ends with a semicolon.

Redim

The only way that's been found to "pass" an array to a Function or subroutine uses Redim. Uh.... judiciously, shall we say?

Try this:

```
N=N + SharedArray(i)
```

RenameMenu

NewText$ is limited to 255 characters. If you rename a menu with a long name, the menu bar will jump to two (maybe more?) lines. But, no, there is no way to add another – ninth – menu. All you get is numbers 0 thru 7.

Select Case

Ever wonder why your macros pop up a box that say "Select Without End Select"? It's because of a "design feature" of the Select statement.

WinWord has to "execute" an End Select prior to finishing a macro – otherwise the bogus "Select Without End Select" box pops up. For example, this will give you a bogus message:

```
Case 1

Goto Bye

Case Else
```

But if you just change it a little bit, you won't get that stupid message:

```
Case 1

Flag=-1

Case Else
```

SendKeys

Keys$ is limited to 255 characters.

SendKeys is nothing but a kludge. But sometimes it's a necessary kludge. In particular, it's an approach of last resort for situations where WinWord support is severely lacking – FilePrinterSetup being a good example.

SendKeys will not work (or at least it won't work the way you expect) when you are editing a macro, using the STEP button.

The keys you want to send may be surrounded by (), e.g., these are the same:

If you want to send a space, use brackets surrounding a space, e.g.

will send a space (or simulate pressing the space bar, if you prefer).

To send a plus key, try

where Alt-043 is formed by turning ON NumLock, holding down Alt, typing 043 on the number pad, then releasing Alt.

SetGlossary

Glossary names are limited to a maximum of 255 characters.

You can use SetGlossary and GetGlossary to store variable values.

```
SetGlossary Name$, Value$, 0
```

```
GetPara$=GetGlossary$(Name$)
```

The problem with this approach is that it sets NORMAL.DOT "dirty", i.e., IsDirty will be on, and the user will be prompted to save global glossary and command changes next time they exit WinWord.

However, if you keep track of whether NORMAL.DOT was dirty before you started storing values in the glossary, use the tip listed in the **Hacker's Guide** on Page 209 (IsDirty), and use SetDirty properly, there's no reason why you can't take advantage of this approach.

On the other hand, SetProfileString may be a cleaner way of doing what you want to do.

Shell

App$ is limited to 255 characters.

To shell out to DOS, try

Page 263

Spike

WinWord adds a paragraph mark to the end of each selection that you place on the Spike.

StepMacro

Another undocumented WinWord command. Seems to work like the macro editor function.

StepMacroSUBs

Yet another undocumented WinWord command. Also seems to work like the macro editor function.

Page 267

Super

The last sentence should read:

"If, for example, you have a global macro named FilePrint, a Super FilePrint statement runs the built-in command."

(The original wording is incorrect if you have both a global and a template macro named FilePrint.)

There was quite a discussion of Super in a recent PC Magazine article. The article claimed that Super did nothing. Horseradish. The manual is right, PC Mag is wrong. (That's mighty unusual, in my experience!)

Text x, y, dx, dy, Text$

Text$ is limited to 255 characters

If you want an ampersand to show up you must double it. I.e., &A will display as A But &&A will display as &A

TextBox x, y, dx, dy, .Field

First, make that a .Field$ It's always a string variable. And it cannot be dimensioned. For example,

will bomb out. Why? Lousy design decision. You'll find some macros (including several of mine) with line after line after line assigning values to undimensioned variables just to handle this dumb requirement.

The Tech Ref is ridiculously confusing about TextBox. All it does is display .Field$, let the user modify it, and return the modified results back to your macro.

In WinWord 1.0, .Field$ was "poured" into the defined area, starting in the upper left corner. But in version 1.1, 1.10a, etc., .Field$ is "poured" into the defined area, centered from top to bottom. Yecch. Another "improvement" in 1.1: Newline and Paragraph marks do not force line feeds any more. Really screwed up one of the early versions of Enveloper with this design decision..... The user cannot type in a Shift+Enter – or any other paragraph mark – so you cannot use TextBox for a lot of things. I don't think Alt-0027 will come through either. Check on other Alt-codes to see if they will work before you use TextBox in any situation that requires anything beyond ANSI Windows characters. It's really too bad, too, because TextBox – if it were implemented properly – could be a very powerful command!

.Field$ is limited to 255 characters. But that usually doesn't matter much, because the number of characters in .Field$ "count" against the dialog box max of 512. That can give you runtime errors – Dialog Box Description Too Complex – if you aren't careful.

If you want an ampersand to show up you must double it. I.e., &A will display as A But &&A will display as &A

Generally you will want a dialog box's "cursor" to go to a TextBox first. If you make a TextBox the first item in the UserDialog, the "cursor" will go there, and highlight any default text you may have inserted.

TraceMacro

Another undocumented WinWord command.

UnLockFields

The Tech Ref has a mighty cryptic description of UnLockFields. Here's an example that may help.

If you insert a date field into your document, format and update it, you'll get the date.

Unfortunately, the next time you update fields for that document, the date will change.

To keep the date field from changing – in fact, to keep any field from being updated during the usual update fields operation – just stick your cursor anywhere inside the field, then issue UnLockFields. (Equivalently, reformed WurdPerfect addicts can push Ctrl-Shift-F9.) Once a field is "unlocked" it won't be updated any more.

UtilCalculate

Go to great lengths to make sure there are no paragraph marks in your selection, prior to running UtilCalculate.

Paragraph marks drive UtilCalculate crazy; a paragraph mark in front of a negative number may hang your system.

UtilHyphenate

For some unknown reason, UtilHyphenate leaves the document selected. That can be very disconcerting to a novice. Always do something like this:

to de-select the document.

UtilSort

Until Microsoft fixes UtilSort, it's virtually useless.

The problem is that there is only a fixed amount of memory that WinWord will use to sort. If WinWord hits a situation where it needs more memory, it just dies with a "Not Enough Memory" message.

This is particularly galling when you have 7 Megs of free memory.....

(I had problems with this very same "quirk" in Word 4 and Word 5, as I recall. Maybe some day they'll fix it?)

UtilSpelling

If you run UtilSpelling in anything but View Draft mode, you won't be able to see the words that need to be corrected! Lousy design.

Here's an oddity to wow the folks in the office:

Type a paragraph or two. Include the word "WurdPerfect" in the first sentence, and mis-spell a half-dozen or so other words in the rest of the document. Click on Utility / Spelling. When WinWord stops on "WurdPerfect", type in "WordPerfect" and click Change. Next time WinWord stops on a mis-spelled word, type in anything and click Change. HeeHeeHee. Does your mis-spelled word change to "WordPerfect", too?

No, it isn't a Cuckoo's Egg, though many of us were suspicious. Until we found out that the same thing happens with "DisplayWrite" and "BillGates". Ends up that any word with a capital in the middle will send UtilSpelling off into never-never land.

In the Description for the argument Word$, the Tech Ref says that if Word$ is omitted, WinWord "searches forward from the insertion point for the next unmatched word." That's only partially correct. Actually, WinWord launches a full-scale spelling check starting at the insertion point.

Incorrectly placed punctuation marks are ignored when spell checking in Word for Windows. Therefore, misspelled words such as "of,and" and "this.is" are not found by the spell checker. The spell checker will find misspelled words on either side of a punctuation mark, but correctly spelled words containing punctuation marks are ignored.

Many people feel the spellchecker is one of the weakest parts of WinWord.

Page 279

Val()

There's a typo.

The last "Statement" should be:

ViewPreferences

When you turn on ViewSpaces – you're used to seeing those nice, little dots representing spaces, right? Most of the time, that's exactly what you get.

Not so if you are using the Symbol of ZapfDingbat font. A space-dot in Symbol comes out to be a big, ugly bullet. A space-dot in ZapfDingbats comes up as a serif "2" in a solid circle. They're both character #183.

Apparently, with View Spaces ON, WinWord captures character #32 (the space) and displays it as character #183. That's OK with most fonts, but not very good in Symbol of ZapfDingbats – and Character 183 will show up as a ñ in any screen font built on the Roman-8 Symbol Set.

Something similar happens to paragraph marks.

Page 289

WindowClose

There is no such command. See DocClose, page 151.

Page 292

Error 5

One of the most inscrutable error messages.

If you get it when you are in a Dialog Box, you probably forgot to include an OKButton. Every dialog box must have an OKButton.

Page 311

Transferring Data between WinWord and EXCEL

The example macro initiating a DDE conversation between Word for Windows and Excel on page 311 of the Tech Ref does not complete all of the operations expected.

The Macro uses Dynamic Data Exchange calls to activate Excel, copy a selection from Word for Windows into an Excel spreadsheet, generate a chart in Excel and copy the chart to the clipboard. The macro then closes down the chart and terminates the DDE channel between the two applications. However, the macro does not close down Excel, so that when the macro is through, Excel is still up on the screen with the spreadsheet open. The DDE channel is closed so that the two applications are no longer communicating, but Excel is left with the focus when the macro is through running.

The following macro is the example given in the above references with two slight modifications.

Beep

MsgBox "make a selection first", "chart selection", 16

Goto finish

[close(0)]"

The additions to the above macro include: 1. An extra [close(0)] command in the second DDEExecute channum statement to close down the Excel spreadsheet without saving. 2. The SendKeys "%fx" command that sends the File Close command to the application Excel.

The result of adding these commands to the example macro allow the macro to end leaving Word for Windows with the focus. This seems to be the intention of the original macro since the last

command:

Print "Excel chart is on the clipboard"

indicates that this line should display in the Word for Windows status bar when the macro is run. However, this line cannot be displayed in Word for Windows unless the above changes are made to the macro, giving Word for Windows the focus at the completion of the macro.

Page 336

= (Expression) Field

There are reports of problems with Expression fields calculated on tables where there are more than 25 rows in the table.

ASK Field

The ask field is similar to the fillin field in that it can prompt you for information, but this information can then be referred to later in several places. You could put this into a macro (use the Insert Field command), but it is more designed to be a print merge field that would normally just be inserted into the template. If you were going to do this with a macro, you might use an inputbox$() statement. The advantage to an ASK statement is that it is easy to do and the result of the ASK statement can be used several times in the document.

If you are looking for a means to print merge information with some fields that will vary (you want to be prompted for the information when you choose File Print Merge) and have some information be pulled from a data document, an ASK field will probably do the job (no macro needed).

AUTONUM Field

When you use {autonum ...} fields, the numbers generated do not show up in the Table of Contents, or most anywhere else. AUTONUMLGL has the same problem.....

Using Autonum is fine for generating sequential numbers, but if any reference is made to them using a bookmark in the diagram title (containing the figure number) in any associated text then the autonumbering is disrupted, as the reference will increment the figure number.

For example: If you have a diagram labelled Figure {autonum} where the autonum has a value of 2, make that a bookmark, and then insert a reference to that bookmark into the text preceeding the diagram then the text will refer to 'Figure 2' when in fact the diagram becomes 'Figure 3'.

What you might want to use in this situation is a Seq field.

Field Result

{seq "figure"} alternator
{seq "figure"} boy
{seq "figure"} text
3 text
{seq "figure"} vacation
{seq "figure"} luck
5 luck

Then to refer to a figure you can do the following:

See Figure {seq "figure" "text"}

There is a bookmark defined just before the {seq "figure"} text line above - or in otherwords, with your cursor just before {seq "figure"} choose Bookmark from the Insert menu and type in the word "text" (without quotes). Look to page 227 of the reference manual for more info on the SEQ field.

AUTONUMLGL Field

If you set a bookmark to the result of an {autonumlgl ...} field, you will not get the number of the paragraph you are referring to, but rather the result of the last {autonumlgl...} field plus 1.

If you select the selection using legal numbering and switch the numbering from automatic to manual with the Utilities Renumber command, you will be able to set bookmarks to the numbers and refer to them later in the document.

When you save a file with {autonumlgl ..} fields in ASCII or WordStar format, the numbers disappear.

Page 346

DATA Field

The Tech Ref neglects to mention the most baffling part of a Data field – the need to double-up on all the backslashes, e.g.:

single backslashes are switches in WinWord fields. Poor choice of delimiters.

Page 355

Fillin Field

The Tech Ref says, "You can use the \d switch with this field to specify a default response to the prompt. If you want the default response to be blank, use the \d switch alone, without any text following it."

This procedure will cause the message "Error! Switch argument not specified" if you do not include an opening and closing quotation mark after the \d switch. For example: {fillin \d ""}

IF Field

If you're trying to figure out how to eliminate the blank line in a print merge when the merge data is null (typical example: don't print a line if Address2 is blank), here's how:

"}{city}, {state} {zip}

The trick is to include a paragraph mark inside the quote, after {Address2}. WinWord takes you quite literally, and will only "print" the paragraph mark if {Address2} is not null.

INCLUDE Field

All sorts of oddities reported here.

{INCLUDE} seems to INSERT a file-it's not passive like DosWord's INCLUDE, or a programmer's #INCLUDE for that matter, all of which insert the file in the print stream without copying its contents.

The problems with that include (no pun intended) that you may want to mix documents from different templates, causing collisions between page formatting and paragraph styles, for example. Plus, {INCLUDE} seems to have a few glitches, such as losing font info on the included doc (kind of like what the glossary does).

Playing with {INCLUDE} trying to make EPS import work, one intrepid user found that importing HUMONGOUS PostScript files gave WinWord fits...aside from making a 3K file 100K suddenly, WinWord seemed to have terrible trouble with page layout after that, like it didn't quite know what to do with that 100K of text destined to occupy just 2 inches of column space.

Page 359

INDEX Field

If you have an index entry with enough page numbers so more than one line is required to hold all of 'em, well, heaven help ya.

There doesn't appear to be a workaround.

Until this is fixed, watch out for long lines of page numbers, and limit the references to any single index entry.

On Page 360, the reference to the \h switch at the top of the page conflicts with the example {index \h "-A-"} given below. The example is correct. Strangely, formatting of "-A-" in that example is ignored. And WinWord requires that you place the text inside quotation marks.

If you want a blank line before and after the character, for example, this will NOT work:

but this does work:

where the @ here is Alt-0011, formed by turning ON NumLock, holding down the Alt key, typing 0011 on the number pad, and releasing the Alt key.

PRINT Field

The print field is ignored if it is located – physically – outside of the printable region of the document.

That's weird, but true!

Example: on a LaserJet, you must allow 0.25 inch clearance all around the document. The remaining 8.0 x 10.5-inch area is called the "printable region".

If you place a {print ...} field inside the top 0.25 inch of the document, and print the document with any LaserJet print driver, the {print ...} field will be completely ignored.

Page 372

REF Field

Ever wonder when you HAVE to use {ref ... }?

Well, when you are merging, these bookmarks are reserved. So if you have a bookmark with one of these names in your document, you need to use {ref ...} to get your document's bookmark:

SEQ Field

Switch heaven. Try this one on for size.....

Here's an example using two sequences, one for sections and another for figures within a section. Since the "tc" line (which is hidden) builds the table of contents entry, the displayed line must be set to NOT increment the sequence numbers.

Section heading:

{seq figure \r 0\h}{seq section \#0.0 \r 1\h} SIMULATION{xe "Sim"}

Reference underneath the graphic:

{tc "Figure {seq section \c#0} {seq figure \n\#0} System Overview" \f F}
Figure {seq section \c#0}{seq figure \c\#0} - System Overview

Table of Contents for the "figures"

{toc \f f \f}

The \h switch here – which is NOT documented – is necessary to make the results of the sequence calculation hidden. Take it out, and the number is printed, even though the field is hidden!

I nominate this sequence of WinWordWizardry for the coveted "Most obscure thing since APL" award (Keith Pleas accepting).....

Page 379

Table of Contents Entry Field

For a report on oddities with {autonum ...} and {tc}, take a look at the **Hacker's Guide** {autonum ...} discussion, page 344

The following macro allows one to designate highlighted text as a table of contents entry without removing the text from the body of the document. With this macro the TC entry doesn't need to be typed twice: once for the body of text , once in the insert field tc entry. The following macro can be assigned to a menu and or to a key combination code to quickly designate selected text to be included in a table of contents.

When all of the table of contents entries have been designated with this macro, choose Insert Table Of Contents and choose Use Table Entry Fields. Table of Contents entries are formatted as hidden text. To edit these entries, you must either have ShowAll on, or Hidden Text selected in View Preferences.

~~HACKER'S GUIDE TO THE UNIVERS:~~
~~WinWord TECH REF EDITION~~
Ends Here

Compiled and Distributed by:

Woody Leonhard
Piencliffe International
Post Office Drawer 7337
Coal Creek Canyon
Golden, Colorado 80403-0100

CompuServe 74730,1734 – Internet 74730.1734@COMPUSERVE.COM

~~, Have Fun! ,~~