This file lists all the macros included in the file **FORTRAN.DOT** with description and source code. It is an accessory to the related WUGNET article. Hopefully, this will give you some further ideas on the use of WordBasic. I also include comments on how to apply many of these macros to real world uses -- few people with a choice run FORTRAN on distant computers.

My experience: Prior to writing these macros, I wrote about 20 of them and looked at maybe 100 more. This is not a large experience base to depend upon. I expect that their are easier ways to handle much of this code. If you know of any, please tell me.

Some notes on WordBasic: In addition to a unhelpful manual, WordBasic has a few other problems: the really important commands cannot be run within the macro editor, macro recording is the only way to identify a number of features (example, removing bookmarks) and shared data is limited. Good news is that you can have 9 windows with files and 4 macro data files. Remember to keep those macro data files closed or you will create an error.

FDC : FDC is the extension I have given formatted fortran files. (If Word for PM were out, I'd probably be adjusting the EA's).

The code is horribly unstructured; don't tell my professors. Also, I left out a lot of potential error checking. And, I am using brute force macros . This kept the macros short enough not to be mind-boggling; too many people have been intimidated by do everything macros.

Added Bonus: If permisssion is granted, a listing of all the current Winword Knowledge Base articles will be appended. Most of the articles do relate cover portions of WordBasic and really are useful.

Acknowledgments: To the Winword Gadflies and their Microsoft counterparts (Guy Gallo, Barry Simon, James Gleick, Doug Timpe, Steve Itijima, and Michael Cockrill) for making WordBasic a tad less daunting.

Enough ChitChat -- Its time to meet the macros
A Modern Day Stone Age Editor
(Just think: next time I might clone EDLIN)

AutoNew: Starts every time you run the file and inserts our template text (hidden in a glossary). Placing the text in a glossary makes it much easier to edit than embedded in a template. (But the template is much better if you must insert large amounts of boilerplate.)

RWWABOUT: Simple message box with design info. It could be extended for simple help. (Larger help requires the help compiler kit).

FDCNEW: Starts a new file using FORTRAN.DOT as the template. This macro and FdcOpen could be placed on the global menus to permit Fdc operation from any position.

FDCOPEN: Uses the standard file.open dialog box to open a FDC file.

FDCSAVE: Forces saves in both a Word format and a text format. (Text for the compiler; Word to keep all the support straight.) Modify this for automatic backups in other disks.

FDCCLOSE: Closes the Fdc file and makes sure changes are also saved as text.

FdcDoLoop: Uses value stored in a bookmark to create a new line number and then updates the bookmark. Good for any special incrementing counters.

FdcLineNumFile: One of four line number insertion routines. This one merely checks if the number has been used according to a data file. When a valid number is chosen, it updates the disk file. (This is a good way to share reverse dictionaries between files.) It will not adjust for discarded line numbers. A bookmark based routine was dropped,. While clearly the best for information needed by only one document, it was big, hard to update, and it had to remain hidden.

FdcLineNumSearch: Same as above; except we run a search for any matching text. This is best if you a few elements all appearing in a similar format. It is slow, however.

FdcLineNum: The simple version of the last two.

InsertComment Continue, and Normal: Each of these three macros just inserts a glossary entry. This is required to do a keybinding.

MainFrame: We do three things on this one: save the current file, switch to a running TERMINAL, or start a new one.

DataFile: This is my pride and joy. DataFile is a form fillin macro. Unlike a related macro I did (which inserted text into a table on the current document), the need to open a new window and save took a surprisingly long time to debug. (Note, a early version did not create another file. However, I had some problems forcing Winword to save the file correctly, using WordBasic file handling routines.)

And now -- the macros themselves:

```
Sub MAIN
REM AutoNew Macro  -- creates template text
EditGlossary .Name = "FHeader", .Context = 1, .Insert
End Sub
```

```
    Sub MAIN
REM RWWABOUT macro -- displays info.
MsgBox "FORTAN.DOT created by R. William Wells", "TEMPLATE ABOUT", 0
End Sub
```

```
    REM FdcNew -- creates new  file using template
Sub MAIN
FileNew 0, "FORTRAN"
End Sub
```

```
    REM FdcOpen -- Uses File.Open to open FDC files
Sub MAIN
Dim lg As FileOpen
GetCurValues lg
lg.Name = "*.FDC"
Dialog lg
Super File Open lg
End Sub
```

```
    REM FdcSave -- Saves 2 files: one FDC with formatting and one FOR pure text
Sub MAIN
Temp = Instr(,WindowName$(),".")
Temp$ = Left$(WindowName$(), Temp)
FileSaveAs Temp$ + ".FDC", 0
FileSaveAs Temp$ + ".FOR", 2
End Sub

    Sub MAIN
REM FdcClose -- special File.Close for Fdc files
If(IsDirty() =(- 1)) Then
Message$ = "Save any changes to " + WindowName$(0) + "?"
Value = MsgBox(Message$, 35) + 2
Else
Value = 2
End  If
Select Case Value
Case 1
FdcSave
FileClose WindowName$()
Case 2
FileClose WindowName$()
Case Else
Cancel
End Select
End Sub

    Sub MAIN
REM FdcDoLoop -- creates Do Continue structures with Line #'s held in Bookmark
REM Start With Basic Info
Do$ = GetGlossary$("FdcContinueLine", 1)
StartOfLine
Insert Do$
REM Need Bookmark To Return Here
InsertBookmark .Name = "FdcCurrent"
REM Grab the last line value add 1, insert into document
Line$ = GetBookmark$("FdcDoLine")
Temp = Val(Line$) + 1
Line$ = Str$(Temp)
LineUp 3
EndOfLine
Insert Line$
LineDown 2
StartOfLine
REM Line$ will be at most 4 characters; start of line must have 6
Insert(" " + Left$(Line$ + "     ", 4) + " ")
REM Update old Line Value and Return to the starting point
```

```
EditGoTo .Destination = "FdcDoLine"
InsertBookmark .Name = "FdcDoLine", .Delete
EditClear
Insert Line$
WordLeft 1, 1
InsertBookmark .Name = "FdcDoLine"
EditGoTo .Destination = "FdcCurrent"
InsertBookmark .Name = "FdcCurrent", .Delete
End Sub

    REM FdcLineNumFile -- creates a file of already used line #s
Sub MAIN
REN Let's find out if the file exists and open it
OldLine$ = "    "
On Error Goto There
Tmp$ = File$("*FORTRAN.TXT")
If Tmp$ <> "" Then
Open "FORTRAN.TXT" For Input As #1
Line Input #1, OldLine$
Close 1
REM We will be saving changes - remove old copy
Kill "FORTRAN.TXT"
EndIf
There: InUse = 0
On Error Goto 0
REM Now get new line number and test it
LineNum$ = InputBox$("Enter Line Number")
If Len(LineNum$) <= 4 Then LineNum$ = LineNum$ + "    "
For I = 1 To Len(OldLine$) Step 4
If Mid$(OldLine$, I, 4) = LineNum$ Then
MsgBox "ERROR LINE NUMBER IN USE"
InUse = 1
End If
Next I
If InUse = 1 Then Goto There
REM Insert new number in the front of the line
StartOfLine
LineNum$ = " " + Left$(LineNum$, 4) + " "
Insert LineNum$
REM Update info file
OldLine$ = OldLine$ + Mid$(LineNum$,2,4)
Open "FORTRAN.TXT" For Output As 1
Print 1, OldLine$
Close 1
End Sub

    REM Insert Line Number and see if it is in use
REM Use Bookmark "Current" to ensure we return here
```

```
Sub MAIN
InsertBookmark "Current"
THERE: LineNum$ = InputBox$("Enter Line Number")
If Len(LineNum$) <= 4 Then LineNum$ = LineNum$ + "    "
Search$ = "^p " + Left$(LineNum$, 4) + " "
EditSearch Search$
If EditSearchFound() = - 1 Then Goto THERE
EditGoTo "Current"
StartOfLine
LineNum$ = " " + Left$(LineNum$, 4) + " "
Insert LineNum$
InsertBookmark .Name = "Current", .Delete
End Sub

REM idiot Line # inserter
Sub MAIN
LineNum$ = InputBox$("Enter Line Number")
If Len(LineNum$) <= 4 Then LineNum$ = LineNum$ + "    "
LineNum$ = " " + Left$(LineNum$, 4) + " "
Insert LineNum$
End Sub

REM Iserts Comment Glossary
Sub MAIN
EditGlossary .Name = "FCOMMENT", .Context = 1, .Insert
End Sub

    REM Inserts Continue Line Glossary
Sub MAIN
EditGlossary .Name = "FContinue", .Context = 1, .Insert
End Sub


    REM Inserts Normal Line Glossary (6 spaces)
Sub MAIN
EditGlossary .Name = "FNormal", .Context = 1, .Insert
End Sub

    REM MainFrame -- Saves and transfers files to HP for compile
Sub MAIN
REM Save Current Files in event of crash
FdcSave
REM Well, if its running; don't need another copy
On Error Goto Europe
AppActivate "Terminal", 1
On Error Goto 0
REM Shell out and hook up to mainframe
Europe: Shell "TERMINAL.EXE HP.TRM", 1
```

```
End Sub


REM Macro DataFile creates correctly spaced text to be read with a format statement
REM Really long macro here

Sub MAIN

Dim dlg As UserDialog

REM Create general data entry dialog

Begin Dialog UserDialog 320, 245
Text 8, 13, 290, 18, "&Employee Name."
TextBox 8, 33, 290, 18, .Name
Text 8, 53, 290, 18, "Employee &Number:"
TextBox 8, 73, 290, 18, .number
Text 8, 93, 290, 18, "&Region:"
TextBox 8, 113, 290, 18, .region
Text 8, 133, 290, 18, "&Sales:"
TextBox 8, 153, 290, 18, .sales
Text 8, 173, 290, 18, "&Commission:"
TextBox 8, 193, 290, 18, .comm
OKButton 45, 220, 64, 18
CancelButton 205, 220, 64, 18
End Dialog

REM Use a new file to store the lines of text
FileNew 0, "NORMAL.DOT"
F$ = InputBox$("Number of lines to insert")
F = Val(F$)
For I = 1 To F
Dialog dlg
Name$ = dlg.Name + "                "
Number$ = dlg.number + "                "
Region$ = dlg.region + "                "
Sales$ = dlg.sales + "                "
Comm$ = dlg.comm + "                "
Line$ = Left$(Name$, 16) + Left$(Number$, 5) + Left$(Region$, 2) + Left$(Sales$, 8) + Left$
(Comm$, 4) + Chr$(11)
Insert Line$
Next I
On Error Goto 0

REM Well, it worked. Save as text and close (no save) window
FileSaveAs "DATAFILE.DAT", 2
FileClose 2
```

```
REM standard abort is here
cancelled:
End Sub
```