```
//****************************************** PR
//      Name:       CvtDocs11.wcm
//      Date:       March 21, 2001
//      Purpose:    Convert documents to WordPerfect format.
//      Desc.:      The macro will convert a single file, a complete directory, or a directory including sub-directories.
//      Note:       This macro will re-compile and run in WPWin6.1, WPWin7.0 (16-bit), WPWin7.0 (32-bit), and
WPWin8.0.  For compiling in WP7, see notes on line #45 and line #56.
//      Special Note:    As a safety precaution always create a directory for the output/converted files.  Do not try to
convert to the same directory (or directory structure).  ALWAYS BACK UP YOUR FILES.
//                       Conversion import types will depend on the user's current version/date of WordPerfect.
MSOffice 97 files (.doc and .rtf) will not convert.
//      Warning:         The macro may give sporadic results when using 'LONG' directory names.
//      To Recompile:    Edit the macro in WordPerfect | Type a space | Delete the space | Press the Save & Compile.
//*********************************************
//      Corel Solution Partners
//      801 765-4151
//      1997, Corel Corporation Limited, All Rights Reserved.
//*********************************************

//********* Pre - Initialization *********
      IfPlatform(Win32)
         DLLCall ProtoType SendMessage("user32.dll"; "SendMessageA"; DWORD!; {DWORD(hWnd);
            DWORD(wMsg);  wParam; lParam})
         DLLCall ProtoType SendMessageString("user32.dll"; "SendMessageA"; DWORD!; {DWORD(hWnd);
            DWORD(wMsg);  wParam; Address(lParam)})
         Global(        CB_SETCURSEL := 334;
            LB_ADDSTRING := 384;
            LB_DELETESTRING := 386;
            LB_DIR:=397;
            LB_FINDSTRINGEXACT := 418;
            LB_GETCOUNT := 395;
            LB_GETTEXT := 393;
            INDEX_START := -1;
            DDL_ARCHIVE := 0;
            DDL_DIRECTORY := 16;
            DDL_EXCLUSIVE := 32768;
            lpszDrive:="";
            lpszDir:="";
            lpszFileName:="";
            lpszExt:="";
            FormatType;
            ErrorLogDir;
            WPFormatNum = 4;              //MA
            HTMLFormatNum = 226;      //MA
            vHrt:=NTOC(0f90ah);
            ver:=MacroInfo(InfoItem:PerfectScriptVersion!; CallLevel:Current!))
            If( substr( ver; 1; 1)  = 2 )
               ver:=MacroInfo(InfoItem:PerfectFitVersion!; CallLevel:Current!)
            EndIf

//*************************************************
//Added for version 11
         If(SubStr(Ver;1;2) = 11)
            Global(vVersion := 11)
            DLLCall ProtoType WfsPathSplit("PFit110.dll"; ; VOID; {Address(lpszFullPath); Address(lpszDrive);
               Address(lpszDir); Address(lpszFileName); Address(lpszExt)})
```

```
            DLLCall ProtoType WcvtDetectFileFormat("PFit110.dll"; ; WORD;
                {Address(AnsiString(lpszFileName))})
            DLLCall ProtoType WcvtConvertFile("PFit110.dll"; ; WORD; {Address(AnsiString(lpszFileName1));
                lpazInfile; Address(AnsiString(lpszFileName2)); lpazOutFile; wFlags; lpCvtInfo})
        EndIf


//*************************************************
//Added for version 10
        If(SubStr(Ver;1;2) = 10)
            Global(vVersion := 10)
            DLLCall ProtoType WfsPathSplit("PFit100.dll"; ; VOID; {Address(lpszFullPath); Address(lpszDrive);
                Address(lpszDir); Address(lpszFileName); Address(lpszExt)})
            DLLCall ProtoType WcvtDetectFileFormat("PFit100.dll"; ; WORD;
                {Address(AnsiString(lpszFileName))})
            DLLCall ProtoType WcvtConvertFile("PFit100.dll"; ; WORD; {Address(AnsiString(lpszFileName1));
                lpazInfile; Address(AnsiString(lpszFileName2)); lpazOutFile; wFlags; lpCvtInfo})
        EndIf


//*******************************************************
// following section may need to be commented out if compiling in WP7
        If(SubStr(Ver;1;1) = 9)
            Global(vVersion := 9)
            DLLCall ProtoType WfsPathSplit("PFit90.dll"; ; VOID; {Address(lpszFullPath); Address(lpszDrive);
                Address(lpszDir); Address(lpszFileName); Address(lpszExt)})
            DLLCall ProtoType WcvtDetectFileFormat("PFit90.dll"; ; WORD;
                {Address(AnsiString(lpszFileName))})
            DLLCall ProtoType WcvtConvertFile("PFit90.dll"; ; WORD; {Address(AnsiString(lpszFileName1));
                lpazInfile; Address(AnsiString(lpszFileName2)); lpazOutFile; wFlags; lpCvtInfo})
        EndIf
//*******************************************************

//*******************************************************
// following section may need to be commented out if compiling in WP7
        If(SubStr(Ver;1;1) = 8)
            Global(vVersion := 8)
            DLLCall ProtoType WfsPathSplit("PFit80.dll"; ; VOID; {Address(lpszFullPath); Address(lpszDrive);
                Address(lpszDir); Address(lpszFileName); Address(lpszExt)})
            DLLCall ProtoType WcvtDetectFileFormat("PFit80.dll"; ; WORD;
                {Address(AnsiString(lpszFileName))})
            DLLCall ProtoType WcvtConvertFile("PFit80.dll"; ; WORD; {Address(AnsiString(lpszFileName1));
                lpazInfile; Address(AnsiString(lpszFileName2)); lpazOutFile; wFlags; lpCvtInfo})
        EndIf
//*******************************************************
// following section may need to be commented out if compiling in WP8
//      If(SubStr(Ver;1;1) = 7)
//          Global(vVersion := 7)
//              DLLCall ProtoType WfsPathSplit("shwin70.dll"; ; VOID; {Address(lpszFullPath);
Address(lpszDrive); Address(lpszDir); Address(lpszFileName); Address(lpszExt)})
//              DLLCall ProtoType WcvtDetectFileFormat("shwin70.dll"; ; WORD;
{Address(AnsiString(lpszFileName))})
//              DLLCall ProtoType WcvtConvertFile("shwin70.dll"; ; WORD;
{Address(AnsiString(lpszFileName1)); lpazInfile; Address(AnsiString(lpszFileName2)); lpazOutFile; wFlags;
lpCvtInfo})
//      EndIf
```

```
//*********************************************************
    EndIfPlatform

    IfPlatform(Win)
        DLLCall ProtoType SendMessage("user.dll"; "SendMessage"; DWORD!; {LoWord(hWnd);
            LoWord(wMsg); LoWord(wParam); lParam})
        DLLCall ProtoType SendMessageString("user.dll"; "SendMessage"; DWORD!; {LoWord(chnd);
            LoWord(sMsg); LoWord(flags); Address(sourcedir);})
        DLLCall ProtoType WfsPathSplit("shwin20.dll"; ; VOID;{Address(InSpec); Address(OutDrive);
            Address(OutPathName); Address(OutFile); Address(OutExt)})
        DLLCall ProtoType WcvtDetectFileFormat("shwin20.dll"; ; WORD; {Address(FileName)})
        DLLCall ProtoType WcvtConvertFile("shwin20.dll"; ; WORD; {Address(AnsiString(lpazInfile));
            LoWord(wInFormat); Address(AnsiString(lpazOutFile)); LoWord(wOutFormat); LoWord(wFlags);
            lpCvtInfo})
        Global(     CB_SETCURSEL :=1024+14;
            LB_ADDSTRING :=1024+1;
            LB_DELETESTRING :=1024+3;
            LB_DIR:=1024+14;
            LB_FINDSTRINGEXACT :=1024+35;
            LB_GETCOUNT :=1024+12;
            LB_GETTEXT :=1024+10;
            INDEX_START := -1;
            DDL_ARCHIVE := 0;
            DDL_DIRECTORY := 16;
            DDL_EXCLUSIVE := 32768;
            lpszDrive:="";
            lpszDir:="";
            lpszFileName:="";
            lpszExt:="";
            FormatType;
            ErrorLogDir;
            vHrt:=NTOC(0f90ah);
            vVersion:=6)
    EndIfPlatform

//********* Initialization *********
MakeDlg1()
IfPlatform(Win32)     // define this item only for 32-bit products, position 1
    If(vVersion  >= 8 )
        DialogAddListItem("Expert1"; 104; "WordPerfect 11")
        DialogAddListItem("Expert1"; 104; "WordPerfect 10")
        DialogAddListItem("Expert1"; 104; "WordPerfect 9")
        DialogAddListItem("Expert1"; 104; "WordPerfect 8")
    EndIf
    DialogAddListItem("Expert1"; 104; "WordPerfect 7")
    DialogLoad("Expert1"; "WordPerfect")
EndIfPlatform
hWnd := DialogHandle("Expert1";104)
DialogAddListItem("Expert1"; 104; "WordPerfect 6")
DialogAddListItem("Expert1"; 104; "WordPerfect 5.1")
DialogAddListItem("Expert1"; 104; "HTML")

SendMessage(hWnd; CB_SETCURSEL; 0;0)

Global(SourceDir:="c:\";
    DestinationDir:="c:\";
```

```
        SourceFile:=SourceDir;
        LBItem:="";
        hWnd1;
        hWnd2;
        hWnd3;
        rb2;
        rb3;
        Rb1:=1;
        cb1loop:=TRUE;
        FormatType:=0;
        vRBChecked:=0;
        Flags:="*.*";
        verrorcount:=1;
        MaxVal:=531;
        extFlag:=0;
        useWPD:=0)

If( vVersion >= 8)              // set upper document limit for the document conversion detection
        MaxVal := 227   // v8 document range ends at 226, sgml detects as 227 in v8 so is not in range here
 Else
        MaxVal :=531    // v6/7 document range ends at 530, this includes sgml which will not convert
EndIf

// ******** Macro Begin ******

DialogShow ("Expert1";"WordPerfect"; cb1)
While(cb1loop)    EndWhile
FigureOutWhatToDo(vRBChecked)
Quit

//******** Macro End*********

Label(cb1)    // label for first callback
        If(cb1[5]=274 or cb1[3]= "CancelBttn1")
            cb1loop:=false
            Quit
        EndIf
        If(cb1[3]="NextBttn")
            MakeDlg2()
            vRBChecked := GetTheRBCheck()
            tempFormat := RegionGetSelectedText("Expert1.104")
            If(vRBChecked > 1)
                RegionShowWindow("Expert2.200"; 0)
                RegionShowWindow("Expert2.201"; 0)
                If(tempFormat <> "HTML")
                    RegionEnableWindow("Expert2.206"; 1)
                    RegionEnableWindow("Expert2.207"; 1)
                 Else
                    RegionEnableWindow("Expert2.206"; 0)
                    RegionEnableWindow("Expert2.207"; 0)
                EndIf
             Else
                RegionShowWindow("Expert2.202"; 0)
                RegionShowWindow("Expert2.203"; 0)
                RegionEnableWindow("Expert2.206"; 0)
                RegionEnableWindow("Expert2.207"; 0)
```

```
            EndIf
            RegionShowWindow("Expert1"; 0)
            DialogShow ("Expert2";"WordPerfect"; cb2)
        EndIf
Return

Label(cb2)          // label for second call back loop, no while/endwhile for this loop
        If(cb2[5]=274 or cb2[3]="CancelBttn2")
            DialogDestroy("Expert2")
            cb1loop:=FALSE
            Quit
        EndIf
        If(cb2[3]="FinishBttn")
            rb1 := RegionGetCheck("Expert1.100")
            rb2 := RegionGetCheck("Expert1.101")
            rb3 := RegionGetCheck("Expert1.102")
            tFlags := RegionGetSelectedText("Expert2.207")       // getflags
            useWPD := RegionGetCheck("Expert2.208")
            extFlag:=TestFlags(tFlags)
            If (extFlag = 0)    Return    EndIf
            FormatType := RegionGetSelectedText("Expert1.104")
            rtnCFN:=CheckFileName(vRBChecked)                    // gets the file/path(s)
            If(FALSE = rtnCFN) Return EndIf
            DialogDestroy("Expert2")
            cb1loop:=FALSE
        EndIf
        If(cb2[3]="BackBttn")
            RegionShowWindow("Expert1"; 1)
            DialogDestroy("Expert2")
        EndIf
        If(cb2[3]=208)   // use WPD ext
            ckWPD := RegionGetCheck("Expert2.208")
            If(ckWPD = 1)
                Messagebox(r;"Warning";"If you choose to change all extensions to .WPD, some files with similar
                names might be overwritten.  Deselect this option to avoid this problem."; IconStop!)
            EndIf
        EndIf
Return

//*****************************
//      Function Name: TestFlags
//      Purpose:      Try to make sure flags user types are valid
//*****************************
Function TestFlags(inFlag)
        if(substr(inFlag; 1; 2) <> "*." OR Strlen(inFlag) > 5)
            MessageBox(r;"Flag Error"; "The flags you have specified are not valid.  An example pattern would be
                *.xxx where x is a file extension (three characters).")
            Return(0)
        Else
            Flags := InFlag
            Return(1)
        EndIf
EndFunc

//*****************************
//      Procedure Name:      MakeDlg1
```

```
//      Purpose:      Create dialog 1
//*****************************
Procedure MakeDlg1()
        DialogDefine("Expert1"; 40; 40; 220; 94; Percent!;"WordPerfect Conversion Expert")
        DialogAddRadioButton ("Expert1";"100"; 10; 10; 55; 10; "Convert File"; rb1)
        DialogAddRadioButton ("Expert1";"101"; 10; 23; 72; 10;"Convert Folder"; rb2)
        DialogAddRadioButton ("Expert1"; "102"; 10; 36; 143; 10; "Convert Folder (include subfolders)"; rb3)
        DialogAddVLine ("Expert1"; "VLine4"; 158; 5; 82)
        DialogAddPushButton ("Expert1"; "NextBttn"; 164; 10; 50; 14; 1; "&Next >")
//      DialogAddPushButton ("Expert1"; "btnCancelBttn"; 164; 27; 50; 14; 0; "Cancel")
        DialogAddPushButton ("Expert1"; "CancelBttn1"; 164; 27; 50; 14; 0; "Cancel")
        DialogAddComboBox ("Expert1"; 104; 10; 62; 112; 58; Droplist!; FormatType)
        DialogAddText ("Expert1"; "text"; 10; 52; 50; 9; Left!; "Output Format:")
EndProc

//*****************************
//      Procedure Name:      MakeDlg2
//      Purpose:      Create dialog 2
//*****************************
Procedure MakeDlg2()
IfPlatform(Win32)
        DialogDefine ("Expert2"; 40; 40; 220; 111; Percent!; Caption:"WordPerfect Conversion Expert")
EndIfPlatform
IfPlatform(Win)   // do not show extension option for 16 bit
        DialogDefine ("Expert2"; 40; 40; 220; 105; Percent!; Caption:"WordPerfect Conversion Expert")
EndIfPlatform
        DialogAddVLine ("Expert2"; "VLine4"; 158; 5; 103; )
        DialogAddPushButton ("Expert2"; "FinishBttn"; 164; 10; 50; 14; 1; "&Finish")
        DialogAddPushButton ("Expert2"; "BackBttn"; 164; 27; 50; 14; 0; "< &Back")
        DialogAddPushButton ("Expert2"; "CancelBttn2"; 164; 44; 50; 14; 0; "&Cancel")
        DialogAddText ("Expert2"; "200"; 10; 10; 41; 9; Left!; "Source File:")
        DialogAddText ("Expert2"; "202"; 10; 10; 94; 9; Left!; "Source Folder:")
        DialogAddFileNameBox ("Expert2"; "201"; 10; 20; 144; 13; FilesAndDirs!; SourceFile; "SourceDir"; "*.*")
        DialogAddFileNameBox ("Expert2"; "203"; 10; 20; 144; 14; DirOnly!; SourceDir; "SourceDir"; "*.*")
        DialogAddText ("Expert2"; "Static12"; 10; 42; 71; 9; Left!; "Destination Folder:")
        DialogAddFileNameBox ("Expert2"; "205"; 10; 54; 144; 14; DirOnly!; DestinationDir; "DestinationDir"; "*.*")
        DialogAddText("Expert2"; 206; 10; 75; 25; 10; 1; "Flags:")
        DialogAddComboBox("Expert2"; 207; 40; 73; 35; 58; Dropdown!; Flags)
            DialogAddListItem("Expert2"; "207"; "*.*")
            DialogAddListItem("Expert2"; "207"; "*.wpd")
            DialogAddListItem("Expert2"; "207"; "*.doc")
            DialogAddListItem("Expert2"; "207"; "*.rtf")
            DialogAddListItem("Expert2"; "207"; "*.txt")
        DialogAddCheckBox("Expert2"; 208; 10; 93; 145; 10; "Change output file extension(s) to .WPD?"; cb5)
EndProc

//*****************************
//      Function Name: CheckFileName
//      Purpose:      Make sure filename/path(s) are valid
//*****************************
Function CheckFileName(RBOption)
        DestinationDir := RegionGetWindowText("Expert2.205")
        DestinationDir := CheckPathSlash(DestinationDir)                    // check for trailing "\"
        CheckDDir := CheckDir(DestinationDir; "Destination")
        DestinationDir := ValidatePath(DestinationDir)
        ErrorLogDir:=DestinationDir
```

```
    If(RBOption = 1)                                        // process one file and quit
        SourceFile := RegionGetWindowText("Expert2.201")
        PathSplit(SourceFile)

        If(DestinationDir = (lpszDrive+lpszDir))
            MessageBox(r;"Destination Folder"; "The destination folder must be different from the source file
                folder."+vhrt+"Please select a different destination folder.";IconInformation!)
            Return(False)
        EndIf
        vSuccess:=CheckSourceFile()
        If(vSuccess = False)
            Messagebox(rtn; "Source File"; "This source file does not exist, or its path does not conform to 8.3
                format. Please eliminate any spaces in the path and name, and reselect the file."; IconStop!)
                    //MA
        EndIf

        Return(vSuccess)
    EndIf

    If((RBOption = 2) or (RBOption =3))                     // check validity of path(s)
        Source := RegionGetWindowText("Expert2.203")
        SourceDir := CheckPathSlash(Source)
        If(SourceDir = DestinationDir)
            Messagebox(rtn;"Folders"; "The source and destination folders are the same. Please select a different
                destination folder!"; IconStop!)
            Return(FALSE)
        EndIf
        CheckSDir := CheckDir(SourceDir; "Source")          // validate directory
        If(CheckSDir = False Or CheckDDir = False)
            Return(FALSE)
        EndIf
        SourceDir := ValidatePath(SourceDir)
        Return(TRUE)
    EndIF
EndFunc

//*****************************
//      Function Name: GetTheRBCheck
//      Purpose:     Checked is used between callback dialogs to set dialog options
//*****************************
Function GetTheRBCheck()
        vRb1 := RegionGetCheck("Expert1.100")
        If(vRB1 = 1)
            Checked := 1
        EndIf
        vrb2 := RegionGetCheck("Expert1.101")
        If(vRB2 = 1)
            Checked := 2
        EndIf
        vrb3 := RegionGetCheck("Expert1.102")
        If(vRB3 = 1)
            Checked := 3
        EndIf
    Return(Checked)
EndFunc
```

```
//*****************************
//      Function Name: PathSplit
//      Purpose:      Break path into component parts
//*****************************
Function PathSplit(vFileName)// prototype and global vars declared in FilesToWP7Protos.wcm
      WfsPathSplit(vFileName; lpszDrive; lpszDir; lpszFileName; lpszExt)
      If(useWPD)
         lpszExt := ".wpd"
      EndIf
      Return(lpszFileName + lpszExt)
EndFunc

//*****************************
//      Function Name: CheckPathSlash
//      Purpose:      Make sure paths have trailing backslash
//*****************************
Function CheckPathSlash(InPath)
      If (InPath = "")
         Return (InPath)
      EndIf
      vLen:=strlen(InPath)
      If(substr(InPath; vLen; 1) = "\")                        //check for \ at end of path
         Return(InPath)
       Else
         InPath := InPath + "\"
         Return(InPath)
      EndIf
EndFunc

//*****************************
//      Function Name: CheckDir
//      Purpose:      Make sure input/output directories exist
//*****************************
Function CheckDir(Dir; dirType)
      vSuccess := DoesDirectoryExist(Dir)
      If (vSuccess = FALSE)
            MessageBox(rtn;"Folder";"The "+ dirType + " folder "+Dir+" is not a valid folder." + vHrt + "Please
                reselect the folder!"; IconStop!)
         Return(FALSE)
       Else
         Return(TRUE)
      EndIf
EndFunc

//*****************************
//      Function Name: CheckSourceFile
//      Purpose:      Make sure source file is valid name
//*****************************
Function CheckSourceFile()
      SourceFile := ValidatePath(SourceFile)
      vSuccess:=DoesFileExist(SourceFile)
      Return(vSuccess)
EndFunc

//*****************************
```

```
//      Procedure Name:      FigureOutWhatToDo
//      Purpose:             Based on choice process single file, single dir, or multidir
//******************************
Procedure FigureOutWhatToDo(Checked)
        AssignFormat()
        If(Checked = 1)                                                 // process one file and quit
            vSuccess:=ConvertFile(SourceFile; DestinationDir)
            If(vSuccess <> "NoConversions")
                MessageBox(rtn;"File(s) Converted";"The macro has converted the file(s) and is finished!";
                    IconInformation!)
            EndIf
            Return
        EndIf

        CreateDLG()             //create the processing dialog
        If(Checked = 2)
            vSourceDir := SourceDir + Flags
            Num := SendMessage(hWnd1; LB_DIR; DDL_ARCHIVE; vSourceDir)
            BuildFileArray()
            ProcessSingleDir()
            DialogDestroy("HidLB")
            Return
        EndIf

        If(Checked=3)
            FillFirst()
            ParseDirs()
            BuildDirArray()   // creates a global array based on the count in lb3
            ProcessMultiDir()
            DialogDestroy("HidLB")
            Return
        EndIf
EndProc

//******************************
//      Procedure Name:      ProcessMultiDir
//      Purpose:             Processes directory and subdirectories of files.
//******************************
Procedure ProcessMultiDir()
        vErrorFlag:=0
        PWMessage(1)
        ForNext (DirNum; 1; Dimensions(ArrayOfDirs[];0))
            RegionResetList("HidLB.one")
            vSourceDir := ArrayOfDirs[DirNum] + Flags
            Num := SendMessage(hWnd1; LB_DIR; DDL_ARCHIVE; vSourceDir)
            NewDest:=CreateNewDir(DirNum)
            vProceed:=BuildFileArray()
            If(vProceed = 0)  Go(Skip@)         EndIf
            ForNext (ItemNum; 1; Dimensions(ArrayOfFiles[];0))
                vFileName:=ArrayOfDirs[DirNum] + ArrayOfFiles[ItemNum]
                RegionSetWindowText("PW.1";
                    "Processing File: " + ItemNum+" - "+ArrayOfFiles[itemnum]+ vHrt+
                    "From Folder: "+ArrayOfDirs[DirNum]+vHrt+
                    "To Folder: "+NewDest+"\")
                vSuccess := ConvertFile(vFileName; NewDest + "\")
                If(vSuccess = "NoConversion" OR vSuccess = 65535)
```

```
                CreateErrorEntry(vFileName)
                vErrorCount:=vErrorCount+1
                vErrorFlag := 1
            EndIF
        EndFor

        Discard(ArrayOfFiles[])          // discard local table
        Discard(ArrayOfFiles[])          // discard global table
        Label(Skip@)
    EndFor
    PWMessage(0)

    If(vErrorFlag = 1)
        FileSave()
        Close()
            MessageBox(rtn;"Macro Finished"; "The macro has finished converting the files.  There were " +
                ( vErrorCount-1) + " file(s) that did not convert." + vHrt + "Please check the error log file in the
                destination folder for a list of unconverted files." + vHrt + vHrt + "FileName: " + ErrorLogDir + "!
                Error.WPD"; IconInformation!)
     Else
            MessageBox(rtn;"Macro Finished"; "The macro has finished converting the files."; IconInformation!)
    EndIf
EndProc

//*****************************
//    Procedure Name:    ProcessSingleDir
//    Purpose:           Converts files for a single directory
//*****************************
Procedure ProcessSingleDir()
    // processing done in this loop
    PWMessage(1)
    ForNext (ItemNum; 1; Dimensions(ArrayOfFiles[];0))
        RegionSetWindowText("PW.1";    "Processing File: " + ArrayOfFiles[itemnum]+vHrt+
                    "From Folder: "+sourcedir+vHrt+
                    "To Folder: "+destinationdir)
        vFileName := SourceDir + ArrayOfFiles[ItemNum]          // create source dir and fname
        vSuccess := ConvertFile(vFileName; DestinationDir)
        If(vSuccess = "NoConversion")
                CreateErrorEntry(vFileName)
                vErrorCount:=vErrorCount+1
                vErrorFlag := 1
        EndIF
    EndFor
    PWMessage(0)

    If(vErrorCount > 1)
        FileSave()
        Close()
            MessageBox(rtn;"Macro Finished"; "The macro has finished converting the files.  There was " +
                (vErrorCount-1) + " file(s) that did not convert." + vHrt + "Please check the error log file in the
                destination folder for a list of unconverted files." + vHrt + vHrt + "FileName: " + ErrorLogDir + "!
                Error.WPD"; IconInformation!)
     Else
            MessageBox(rtn;"Macro Finished"; "The macro has finished converting the files."; IconInformation!)
    EndIf
EndProc
```

```
//*****************************
// Creates !error.wpd - error log, files that did not convert
//*****************************
Procedure CreateErrorEntry(vFileName)
        FileExists(vErrorFile; ErrorLogDir+"!Error.wpd")
        If(vErrorFile=FALSE)
            FileNew()
            FileSave(ErrorLogDir+"!Error.wpd")
        EndIf
        Type(vFileName)
        HardReturn
EndProc

//*****************************
//      Function Name: ConvertFile
//      Purpose:
//*****************************
Function ConvertFile(vFileName; DestinationDir)      // input filename; save to dir name;
        InFileSpec := DetectFormat(vFileName)
            If(InFileSpec >  0 and InFileSpec < MaxVal)// this is a document
            vSuccess := ConvertToWP(vFileName; InFileSpec; DestinationDir)
            Return(vSuccess)
         Else
            Return("NoConversion")
        EndIf
EndFunc

//*****************************
//      Function Name: ConvertToWP
//      Purpose:     Converts input file to the desired format and destination dir
//        Changes:   11-05-1999 (donk for WP9 SP1): added If ( InFileSpec = FormatType ) so skip conversion if
original file is already in the correct format.  Changed call to WebPublish(..) to a call to FileSave(..) so that the
HTML conversion selection would work in WP9.
//*****************************
Function ConvertToWP(vFileName; InFileSpec; DestinationDir)          // prototype in FilesToWP7Protos.wcm
        vOnlyFileName := PathSplit(vFileName)
        OutPathFN := DestinationDir  +  vOnlyFileName

        // only do a FileCopy if we're converting the same file type
        If ( InFileSpec = FormatType )
            FileCopy( vFileName; OutPathFN )
            // set our return value
            vSuccess := 0
        Else
            // do the conversions
            If( FormatType <> HTMLFormatNum )
                // ALL other formats
                vSuccess := WcvtConvertFile(vFileName; InFileSpec; OutPathFN; FormatType; 0; 0)
            Else
                // convert to HTML by converting the file to WPD, opening it in WP and saving as HTML
                // calling WcvtConvertFile( vFileName; InFileSpec; OutPathFN; HTMLFormatNum .. ) doesn't seem
                // to work.  Maintained previous assumption that this will only work on Win32.
                IfPlatform(Win32)

            vSuccess := WcvtConvertFile(vFileName; InFileSpec; OutPathFN; WPFormatNum; 0; 0)
```

```
            FileOpen( OutPathFN; WPFormatNum)                                   // open cvt file

            NewHTMLFN := SubStr(OutPathFN; 1; StrPos(OutPathFN; "."))
                    FileSave (Filename: NewHTMLFN + "htm"; ExportType:HTML!; Overwrite:Prompt!)
            EndIfPlatform
                Close()
                DeleteFile(OutPathFN; NoPrompts!)
            EndIf _____// FormatType < 226
        EndIf    //InFileSpec == FormatType

        return(vSuccess)
EndFunc

//*****************************
//      Function Name: DetectFormat
//      Purpose:     Detects format of input file, used by WcvtConvertFile
//*****************************
Function DetectFormat(vFileName)            // prototype in FilesToWP7Protos.wcm
        ret_doc_type := WcvtDetectFileFormat(vFileName)
        return(ret_doc_type)
EndFunc

//*****************************
//      Procedure Name:      CreateDLG
//      Purpose:            Create the dialog - this should not be modified
//*****************************
Procedure CreateDLG()
        DialogDefine("HidLB";50;50;425;120;1|2;"parse dirs")
        DialogAddListBox("HidLB";"one";5;5;100;100;1;vlist)
        DialogAddListBox("HidLB";"two";110;5;100;100;1;vlist)
        DialogAddListBox("HidLB";"three";215;5;200;100;1;vlist)
        IfPlatform(Win32)
            DialogLoad("HidLB"; "WordPerfect")
        EndIfPlatform
        hWnd1 := DialogHandle("HidLB";"one")
        hWnd2 := DialogHandle("HidLB";"two")
        hWnd3 := DialogHandle("HidLB";"three")
EndProc

//*****************************
//      Function Name: BuildFileArray
//      Purpose:     Creates an array containing all items from lb3
//*****************************
Function BuildFileArray()
        vfile:=""
        Num:=SendMessage(hWnd1; LB_GETCOUNT; 0; 0)
        If (Num = 0 And rb2 = 1)
            MessageBox(rtn; "No Files"; "No files were found in the specified folder, macro will terminate"; IconStop!)
            Quit
        EndIf
        If (Num = 0 And rb3=1)   Return(0)      EndIf
        Global (ArrayOfFiles[Num])
        IF(Num<>0)
            ForNext(x;0;Num-1)
                len:=SendMessageString(hWnd1; LB_GETTEXT; x; vfile)
                ArrayOfFiles[x+1]:=vfile
```

```
            EndFor
        EndIF
        Return(1)
EndFunc

//*****************************
//      Procedure Name:     BuildDirArray
//      Purpose:            Creates an array containing all items from lb3
//*****************************
Procedure BuildDirArray()
        path:=""

        Num:=SendMessage(hWnd3; LB_GETCOUNT; 0; 0)
        Global (ArrayOfDirs[Num])
        IF(Num<>0)
            ForNext(x;0;Num-1)
                len:=SendMessageString(hWnd3; LB_GETTEXT; x; path)
                ArrayOfDirs[x+1]:=path
            EndFor
        EndIF
EndProc

//*****************************
//      Procedure Name:     ParseDirs
//      Purpose:            Finds all subdirectories of the source directory
//*****************************
Procedure ParseDirs()
        num:=SendMessage(hWnd2; LB_GETCOUNT; 0; 0)
        Repeat
            ForNext(x;1;num)
                len:=SendMessageString(hWnd2; LB_GETTEXT; 0; sourcedir)
                rtn:=SendMessage(hWnd3; LB_FINDSTRINGEXACT; INDEX_START; sourcedir)
                If(rtn < 0)     SendMessageString(hWnd3; LB_ADDSTRING; 0; sourcedir) EndIf
                SendMessage(hWnd2; LB_DELETESTRING; 0; 0)
                numsub:=SendMessage(hWnd1; LB_DIR;DDL_DIRECTORY+DDL_EXCLUSIVE; SourceDir+flags)
                checklist()
                count:=SendMessage(hWnd2; LB_GETCOUNT; 0; 0)
                If(count = 0) Break Endif
                num:=Count
            EndFor
        Until(count = 0)
EndProc

//*****************************
//      Procedure Name:     FillFirst
//      Purpose:            Fills lb1 with initial values.
//*****************************
Procedure FillFirst()
        SendMessageString(hWnd3; LB_ADDSTRING; 0; sourcedir)
        SendMessage(hWnd1; LB_DIR;DDL_DIRECTORY+DDL_EXCLUSIVE; SourceDir+flags)
        CheckList()
EndProc

//*****************************
//      Procedure Name:     CheckList
//      Purpose:            Deletes '[..]' from lb1, concatenates path, moves strings to lb2 and deletes from lb1
```

```
//*****************************
Procedure CheckList()
	rtn:=SendMessage(hWnd1; LB_FINDSTRINGEXACT; INDEX_START; "[..]")
	While(rtn >= 0)
	    num:=SendMessage(hWnd1; LB_DELETESTRING; rtn; 0)
	    rtn:=SendMessage(hWnd1; LB_FINDSTRINGEXACT; INDEX_START; "[..]")
	EndWhile
	num:=SendMessage(hWnd1; LB_GETCOUNT; 0;0)
	If(num >0)
	    ForNext(x;0;(Num-1))
	        SendMessageString(hWnd1; LB_GETTEXT; 0; LBItem)
	        val:=SendMessage(hWnd1; LB_DELETESTRING; 0; 0)
	        st:=sourcedir+substr(LBItem; 2; strlen(LBItem)-2)+"\"
	        SendMessageString(hWnd2; LB_ADDSTRING; 0; st)
	    EndFor
	EndIf
EndProc

//*****************************
//	Function Name: CreateNewDir
//	Purpose:	Used by multi-dir, to create mirror directories
//*****************************
Function CreateNewDir(DirNum)
	newdir:=DestinationDir + SubStr(ArrayOfDirs[DirNum]; 4; StrLen(ArrayOfDirs[DirNum]) - 4)
	If (Not(DoesDirectoryExist(newdir)))
	    IfPlatform(Win)
	        CreateDirectory(newdir)			// token change from 6 to 7
	    EndIfPlatform
	    IfPlatform(Win32)
	        DirectoryCreate(newdir)
	    EndIfPlatform
	EndIf
	Return(newdir)
EndFunc

//*****************************
//	Procedure Name:	AssignFormat
//	Purpose:		Assign the correct format for WcvtConvertFile
//*****************************
Procedure AssignFormat()
	Switch(FormatType)
	    Caseof "WordPerfect 11":
	        FormatType := 6
	    Caseof "WordPerfect 10":
	        FormatType := 6
	    Caseof "WordPerfect 9":
	        FormatType := 6
	    Caseof "WordPerfect 8":
	        FormatType := 6
	    Caseof "WordPerfect 7":
	        FormatType := 6
	    Caseof "WordPerfect 6":
	        FormatType := 4
	    Caseof "WordPerfect 5.1":
	        FormatType := 3
	    Caseof "HTML":			// html detection version differ from 7 to 8
```

```
            If ( vVersion >= 8 )
                FormatType := 226           // if the version = 8 or 9
             Else
                FormatType := 530           // if the version = 7
            EndIf
        Default:
            FormatType := 4
    EndSwitch
EndProc

//*****************************
// shows/destroys please wait message
//*****************************
Procedure PWMessage(Show)
    If(Show = 1)
        IfPlatform(Win32)
            DialogDefine("PW"; 30;30;230; 60; 2|16; "Please Wait")
        EndIfPlatform
        IfPlatform(Win)
            DialogDefine("PW"; 30;30;230; 65; 2|16; "Please Wait")
        EndIfPlatform
        DialogAddText("PW"; 1; 5; 5; 220; 27; WPChars!|MultiLine!;"")
        IfPlatform(Win32)
            DialogLoad("PW";"WordPerfect")
        EndIfPlatform
        DialogShow("PW";"WordPerfect"; PWMsg)
    EndIf
    If(Show = 0)
        DialogDestroy("PW")
    EndIf
EndProc

//*****************************
// Callback proc for please wait message
//*****************************
Procedure PWMsg()
    If(PWMsg[3]="CancelBttn")
        DialogDestroy("PW")
        Quit
    EndIf
EndProc

//***************************************
//      Name :      ValidatePath(SourcePath)
//      Purpose:    Validate path for call to LB_Dir
//      Input:      SourcePath - a directory/directory+filename
//      Returns:    Returns the validated path (string) - shortened directory names: c:\Valid~1\path\
//      Note:       The LB_DIR message used in the call to SendMessage
//                  can not use long directory names.  This function will create the valid path
//                  for the LB_DIR message, otherwise no files are returned.
//      Uses:       Uses function StringToArray
//***************************************
Function ValidatePath(SourcePath)
    vLen := StrLen(SourcePath)
    If(vLen = 3)
        Return(SourcePath)
```

```
        EndIf
        lpszDrive:=""
        lpszPath:=""
        lpszFileName:=""
        lpszExt:=""
        WfsPathSplit(SourcePath; lpszDrive; lpszPath; lpszFileName; lpszExt)
        ArrayOfDirs[] := StringToArray(lpszPath; "\")
        ForNext(x; 1; Dimensions(ArrayOfDirs[]; 0 ))
            If(StrLen(ArrayOfDirs[x]) > 8)
                ArrayOfDirs[x] := Substr(ArrayOfDirs[x]; 1; 6) + "~1"
            EndIf
            If(x <> 1)
                ResultPath := ResultPath + "\" + ArrayOfDirs[x]
             Else
                ResultPath := ArrayOfDirs[x]
            EndIf
        EndFor
        ResultPath:=lpszDrive+"\"+ResultPath+"\"+ lpszFilename+lpszExt
        lpszDrive:=""
        lpszPath:=""
        lpszFileName:=""
        lpszExt:=""
        Return(ResultPath)
EndFunc

//**************************************
//      Name :              StringToArray (InputString; Separator)
//      Purpose:    Create an array based on elements of a delimited string
//      Input:          InputString - the delimited string
//                  Separator - the separator character that delimits the string
//      Returns:    Returns an array or x dimensions, each element being a section of the delimited string
//      Note:           Call this function using the following syntax:
//                      MyNewArray[]:=StringToArray(vStringName; "|")
//                  To find the number of elements in the array, use the DIMENSIONS command.
//**************************************

Function StringToArray (InputString; Separator)
        TempString:=InputString
        Count := 0
        vLen:=StrLen(TempString)
        If(SubStr(TempString; 1;1) = "\" and SubStr(tempstring; vLen; 1) = "\")
            TempString:=SubStr(TempString; 2; vLen - 2)
            InputString:=TempString
        EndIf
        If( StrPos(TempString; Separator) = 0)
            NewArray[]:={ TempString }
            Return(NewArray[])
        EndIf
        Repeat
            vPos:=StrPos(TempString; Separator)
            vLen:=StrLen(TempString)
            TempString:=SubStr(TempString; vPos+1; vLen-vPos)
            Count:=Count+1
        Until(StrPos(TempString; Separator) = 0)

        Declare (NewArray[(Count+1)])          Count:=0
```

```
    While(StrPos(InputString; Separator) <> 0)
        Count:=Count+1
        vPos:=StrPos(InputString; Separator)
        NewArray[Count]:=SubStr(InputString; 1; vPos-1)
        vLen:=StrLen(InputString)
        InputString:=SubStr(InputString; vPos+1; vLen-vPos)
    EndWhile
    NewArray[Count+1]:=InputString

    Return (NewArray[])
EndFunc
```