```
//*********************************************************
//        MACRO:  _AUTOFIL.WCM
//        PURPOSE:  Automates the fill-in process of WordPerfect templates. This macro also uses
the adrsbook.wcm (Address Book) macro.
//*********************************************************
Application (A1; "WordPerfect"; Default; "UK")
Call(WaitMsgInit@)
Call(WaitMsgDisplay@)
Display(Off!)
DelimiterChar:="|"
MaxPrompts:=12                  // Maximum number of prompts allowed
RowSpacing:=17                  // Controls line spacing in dialog
FirstRow:=8                     // Sets VPos of first edit box
Declare Prompt[MaxPrompts,2]    // Holds prompts and address book link numbers
Declare PromptVar[MaxPrompts]   // Holds contents from template edit boxes
Global(AdrsBookRunMode)
AdrsBookRunMode:=3              // Initialize Address Book mode

//*********************************************************
//        PROGRAM BEGIN
//*********************************************************
Call(OpenAddressBook@)          // Get personal information
If(SelectedName[2]="")
        Go(Stop@)
EndIf
MacroStatusPrompt(On!;"Scanning template...")
Call(MainDialog@)
Call(Replace@)
Label(Stop@)
Call(WaitMsgHide@)
Call(WaitMsgDestroy@)
QuickmarkFind()
Quit
//*********************************************************
//        PROGRAM END
//*********************************************************

//*********************************************************
//        ROUTINE NAME: OpenAddressBook@
//        INPUT VARIABLES: AdrsBookRunMode
//                1=Display addresses
//                2=Display personal information
//                3=Get default personal information
//        RETURN VARIABLES: FldName[], FldVar[], MaxFld[], SelectedName[]
//        USES: None
//        DESCRIPTION: Provides access to address book information.
```

```
//*********************************************************
Label(OpenAddressBook@)
MacroName:="adrsbook.wcm"                    // Set name of address book macro.
ForEach(Location;{?PathMacros;?PathMacrosSupplemental;?PathCurrent})
        If(Location<>"")
                If(SubStr(Location;StrLen(Location);1)<>"\")
                        Location:=Location+"\"
                EndIf
                MacroPath:=Location+MacroName
                FileExists(FExists; MacroPath)
                If(FExists)
                        Run(MacroPath)
                        Return
                EndIf
        EndIf
EndFor
MessageBox(; "Unable to Locate Address Book"; "Cannot find the ^0 (Address Book) macro.
Please make sure the macro is in your default macro directory."; IconStop! | HasParameters!;
MacroName)
Call(WaitMsgHide@)
Call(WaitMsgDestroy@)
Quit
//*********************************************************

//*********************************************************
//      ROUTINE NAME: MainDialog@
//      INPUT VARIABLES: None
//      RETURN VARIABLES: None
//      USES: AutoDlg@, ParseAbbr@,  OpenAddressBook@
//      DESCRIPTION: Locates, displays, and evaluates template prompts.
//*********************************************************
Label(MainDialog@)
Call(ParseAbbr@)
If(NumOfPrompts=0)
        Return
EndIf
// Check for existence of lookup fields
Discard(ButtonText2)
For(Count;1;Count<=NumOfPrompts;Count+1)
        If(Prompt[Count;2]<>0)
                ButtonText2:="&Address Book"
                Break
        EndIf
EndFor
Call(AutoDlg@)
Label(DisplayPrompts@)
```

```
MacroStatusPrompt(On!;"Enter template information")
Display(On!)
DialogDisplay("Prompts";"EB1")
Display(Off!)
Switch(MacroDialogResult)
CaseOf 1:       // "OK"
        For(Count;1;Count<=NumOfPrompts;Count+1)
                If(PromptVar[Count]="")
                        MessageBox(MsgBoxResult; "Confirmation"; "One or more responses are
                        blank.  Is this correct?"; IconQuestion! | YesNo!)
                        If(MsgBoxResult=6)  // Yes
                                Break
                        Else
                                Go(DisplayPrompts@)
                        EndIf
                EndIf
        EndFor
        DialogDestroy("Prompts")
        Return
CaseOf "PB1":                   // Personal Info
        AdrsBookRunMode:=2
        Call(OpenAddressBook@)
        Go(DisplayPrompts@)
CaseOf "PB2":                   // Address Book
        AdrsBookRunMode:=1
        Call(OpenAddressBook@)
        If(SelectedName[1]<>"")
                For(Count;1;Count<=NumOfPrompts;Count+1)
                        If(Prompt[Count;2]<>0)
                                PromptVar[Count]:=FldVar[1;Prompt[Count;2]]
                        EndIf
                EndFor
        EndIf
        Go(DisplayPrompts@)
Default:                // "Cancel" and all others
        DialogDestroy("Prompts")
        Go(Stop@)
EndSwitch
//*********************************************************

//*********************************************************
//      ROUTINE NAME: Replace@
//      INPUT VARIABLES: NumOfPrompts
//      RETURN VARIABLES: None
//      USES: Nothing
//      DESCRIPTION:  Places information from dialogs into the current template.
```

```
//*******************************************************
Label(Replace@)
MacroStatusPrompt(On!;"Please wait...placing information in template")
SingleString:=GetAbbreviationContents("Template Bookmarks")
If(SingleString="")
        MessageBox(; "Template Bookmarks Missing"; "The abbreviation ""Template
        Bookmarks"" is not in the current template.  This abbreviation is necessary to fill out the
        template properly.  You may be able to run TCONVERT.WCM to create this abbreviation
        from the information already contained in the template. "; IconStop!)
        Return
EndIf
IndVar:="FldVar"
IndIndex:="[2;"
BkmrkName:=0
Repeat
        BkmrkName:=BkmrkName+1
        StrPos(NextDelimLoc; DelimiterChar; SingleString)
        LinkNum:=SubStr(SingleString; 1; NextDelimLoc-1)
        SingleString:=SubStr(SingleString; NextDelimLoc+1; StrLen(SingleString)-
        (NextDelimLoc-1))
        If(LinkNum="")
                IndVar:="PromptVar"
                IndIndex:="["
                BkmrkName:=BkmrkName-1
                Next
        EndIf
        BookmarkBlock(BkmrkName)
        Type(Indirect(IndVar+IndIndex+LinkNum+"]"))
Until(NextDelimLoc=0)
While(?Substructure)
        SubDocType:=?CurrentSubDoc
        SubstructureExit()
        If((SubDocType = 10) Or (SubDocType = 11))
                BoxEnd(Save!)
        EndIf
EndWhile
Return
//*******************************************************

//*******************************************************
//      ROUTINE NAME: AutoDlg@
//      INPUT VARIABLES: ButtonText1(Optional); ButtonText2 (Optional);  NumOfPrompts
//      RETURN VARIABLES: ButtonPushed: 1=OK, 2=Cancel, "PB1"=Button 1,
"PB2"=Button 2
//      USES: Nothing
//      DESCRIPTION:  Automatically sizes a dialog based on the number of elements inside
```

```
the specified array variable.
//*********************************************************
Label(AutoDlg@)
// Calculate dialog width
Longest:=0
For(Count;1;Count<=NumOfPrompts;Count+1)
        Len:=StrLen(Prompt[Count;1])
        If(Len>Longest)
                Longest:=Len
        EndIf
EndFor
If((Longest<10) And (Exists(ButtonText2)>0))
        Longest:=10
EndIf
Longest:=Longest*4   //Four dialog units per character
CurrRow:=FirstRow
DlgHeight:=((NumOfPrompts)*RowSpacing)+45
DialogDefine("Prompts";50;50;Longest+179;DlgHeight;19;"Template Information")
// Personal Information Button
DialogAddPushButton("Prompts";"PB1";8;DlgHeight-35;53;13;0;"&Personal Info")
If(Exists(ButtonText2)=1)
        // Address Book Button
        DialogAddPushButton("Prompts";"PB2";64;DlgHeight-35;53;13;0;ButtonText2)
EndIf
For(Count;1;Count<=NumOfPrompts;Count+1)
        StringNum:=NumStr(Count)
        TextID:="TB"+StringNum
        EditID:="EB"+StringNum
        DialogAddText("Prompts";TextID;8;CurrRow+2;Longest+4;10;32;Prompt[Count;1]+":")
        DialogAddEditBox("Prompts";EditID;Longest+17;CurrRow;150;13;33;PromptVar[Coun
        t];80)
        CurrRow:=CurrRow+RowSpacing
EndFor
Return
//*********************************************************

//*********************************************************
//      ROUTINE NAME: ParseAbbr@
//      INPUT VARIABLES: DelimiterChar; MaxPrompts
//      RETURN VARIABLES: NumOfPrompts; Prompt[MaxPrompts;2]
//      USES: Nothing
//      DESCRIPTION: This routine extracts the contents of a delimited abbreviation.  Fields of
the abbreviation must be separated by the character assigned in "DelimiterChar".
//*********************************************************
Label(ParseAbbr@)
NumOfPrompts:=0
```

```
StringToParse:=GetAbbreviationContents("Template Prompts")
If(StringToParse="")
	Return
EndIf
NumOfPrompts:=1
Label(ParseIt@)
DelPos:=StrPos(StringToParse;DelimiterChar)
// Assign prompt to variable
Prompt[NumOfPrompts;1]:=SubStr(StringToParse;1;DelPos-1)
// Cut assigned prompt from main string
StringToParse:=SubStr(StringToParse;DelPos+1;StrLen(StringToParse)-1)
DelPos:=StrPos(StringToParse;DelimiterChar)
If(DelPos=0)
	Prompt[NumOfPrompts;2]:=StringToParse
	Return
EndIf
// Assign link to variable
Prompt[NumOfPrompts;2]:=SubStr(StringToParse;1;DelPos-1)
// Cut assigned link from main string
StringToParse:=SubStr(StringToParse;DelPos+1;StrLen(StringToParse)-1)
If(NumOfPrompts=MaxPrompts)
	Return
EndIf
NumOfPrompts:=NumOfPrompts+1
Go(ParseIt@)
Return
//*********************************************************

//*********************************************************
//	ROUTINE NAME: WaitMsgInit@
//	INPUT VARIABLES: None
//	RETURN VARIABLES: None
//	USES: Nothing
//	DESCRIPTION: Initializes macro wait messages.
//*********************************************************
Label(WaitMsgInit@)
WaitDlgId:="Wait1"
WaitTitle:="Autofill Macro"
WaitMessage:="Please wait..."
DialogDefine(WaitDlgId;50;50;140;50;16+32;WaitTitle)
DialogAddText(WaitDlgId;"WaitText";8;12;124;10;1+4;WaitMessage)
Return

Label(WaitDlgCallBack@)
Return
//*********************************************************
```

```
//*********************************************************
//      ROUTINE NAME:  WaitMsgDisplay@
//      INPUT VARIABLES:  WaitDlgId
//      OUTPUT VARIABLES:  None
//      USES: Nothing
//      DESCRIPTION:  Displays a previously defined wait message dialog.
//*********************************************************
Label(WaitMsgDisplay@)
DialogDisplay(WaitDlgId;0;WaitDlgCallBack@)
Return
//*********************************************************

//*********************************************************
//      ROUTINE NAME:  WaitMsgHide@
//      INPUT VARIABLES:  WaitDlgId
//      OUTPUT VARIABLES:  None
//      USES:  Nothing
//      DESCRIPTION:  Hides a previously defined wait message dialog.
//*********************************************************
Label(WaitMsgHide@)
DialogUndisplay(WaitDlgId;1)
Return
//*********************************************************

//*********************************************************
//      ROUTINE NAME:  WaitMsgDestroy@
//      INPUT VARIABLES:  WaitDlgId
//      OUTPUT VARIABLES:  None
//      USES: Nothing
//      DESCRIPTION:  Destroys a previously defined wait message dialog. Hide the dialog
with WatMsgHide before destroying it.
//*********************************************************
Label(WaitMsgDestroy@)
DialogDestroy(WaitDlgId)
Return
//*********************************************************

//*********************************************************
//      FUNCTION NAME:  GetAbbreviationContents
//      IN::  AbbreviationName
//      RETURNS:  Contents of specified abbreviation.
//      USES: Nothing
//      DESCRIPTION:  Retrieves the contents of the specified abbreviation.
//*********************************************************
Function GetAbbreviationContents(AbbreviationName)
```

```
StringToParse:=""
// Get number of abbreviations in template.
GetData(NumOfAbbrs;Abbreviation!;Count!;CurrentDoc!)
If(NumOfAbbrs=0)
        Return("")
EndIf
// For the number of abbreviations,
For(CurrAbbrNum;1;CurrAbbrNum<=NumOfAbbrs;CurrAbbrNum+1)
        // get name of each abbreviation until the correct one is found
        GetData(CurrAbbrName;Abbreviation!;Name!;CurrentDoc!;CurrAbbrNum)
        If(CurrAbbrName=AbbreviationName)
                // Get contents of abbreviation
                GetData(StringToParse;Abbreviation!;Data!;CurrentDoc!;CurrAbbrNum)
        EndIf
EndFor
Return(StringToParse)
EndFunc
//*********************************************************
```