```
//*************************************************************
//*        MACRO:  AUTOFILA.WCM
//*        PURPOSE:  Automates the fill-in process of WordPerfect templates.
//*                  Modified from the WPC AUTOFILL to allow the Originator
//*                   information  to be stored in the current template
//*                  .J Filshie  November 18, 1993
//*************************************************************
Application (A1; "WordPerfect"; Default; "US")
Disclaimer :="Modified by J Filshie (CIS ID 100016,1311)"
If(?TemplateFile=?CurrentTemplate)          // Check if this is the default template
        MsgBoxCaption:="Warning - Default Template"
        MsgBoxMessage:="The current template is your default template.  Placing Originator Information in the
        default template will make it accessable to all templates that use Personal Information running the autofila
        macro."
        MsgBoxFlags:=49
        Call(MessageBox@)
        If(MsgBoxResult<>1)
                Quit
        EndIf
EndIf
WaitMessage:="Please wait..."
Call(WaitMessage@)
OnError(Error@)
Display(Off!)
// Set general macro variables
UniqueIdentifier:="REC:"
DelimiterChar:="|"
NoAbbr:=False
NoContent:=False
FirstTime:=False                    // Tests if running macro for first time
DialogCreated:=False                        // State of name selection dialog
RowSpacing:=17                      // Controls line spacing in dialogs
FirstRow:=8                                 // Sets VPos of first dialog edit box
NumDefFlds:=7                       // Holds number of default originator information fields
MaxAdrsBookVar:=9
Declare(LookVar[MaxAdrsBookVar])
Declare(EditVars[NumDefFlds])       // Holds temporary copy of DefVars
Declare(DefFlds[NumDefFlds])        // Holds default originator field names
Declare(DefVars[NumDefFlds])        // Holds contents of originator fields
Declare(AddPrompts[24])             // Holds additional field prompts
Declare(AddVars[12])                // Holds additional field contents
// Set names of originator information fields
DefFlds[1]:="Name"
DefFlds[2]:="Title"
DefFlds[3]:="Company"
DefFlds[4]:="Address"
DefFlds[5]:="City, State Zip"
DefFlds[6]:="Telephone"
DefFlds[7]:="Fax"

//*************************************************************
// PROGRAM BEGIN
//*************************************************************
Call(GetOriginator@)                // Get originator info
Call(GetPrompts@)
If(NumOfPrompts<>0)
        Call(DisplayPrompts@)
EndIf
Call(Replace@)
Label(Stop@)
Call(DestroySelectDlg@)
```

```
Call(KillWaitMessage@)
QuickmarkFind()
Quit
//****************************************************************
// PROGRAM END
//****************************************************************

//****************************************************************
//*        ROUTINE NAME:  GetOriginator
//*        INPUT VARIABLES:  None
//*        OUTPUT VARIABLES:  DefVars - originator information.
//*        DESCRIPTION:  Puts originator information into variables.  Prompts for originator information if none is
found.
//****************************************************************
Label(GetOriginator@)
MacroStatusPrompt(On!;"Loading originator information")
TemplateType:=0             // Default template
InfoType:=1
AbbrName:="Originator Information"
Call(ParseAbbr@)
If(NoAbbr)                  // If abbreviation does not exist,
        FirstTime:=True
        Call(SetOriginator@)
        FirstTime:=False
EndIf
Return

Label(SetOriginator@)
If(FirstTime)
        Call(IntroDlg@)
        Call(FirstDlg@)
        Return
Else
        Call(NotFirstDlg@)
        Return
EndIf

Label(IntroDlg@)
MacroStatusPrompt(On!;"Enter Originator in thisTemplate")
MsgBoxCaption:="Enter Originator in thisTemplate"
MsgBoxMessage:="The autofila macro allows you to enter information about the orginator that will personalize the
templates.  You only need to enter this information once for each template."
MsgBoxFlags:=48
Call(HideWaitMessage@)
Call(MessageBox@)
Call(ShowWaitMessage@)
Return

Label(FirstDlg@)
CurrRow:=FirstRow
DialogDefine("DlgId";50;50;227;214;19;"Enter the Originator Information")
For(Count;1;Count<=NumDefFlds;Count+1)                  //For number of prompts possible
        NumStr(StringNum;0;Count)
        TextID:="TB"+StringNum
        EditID:="EB"+StringNum
        DialogAddText("DlgId";TextID;8;CurrRow+2;49;10;1;DefFlds[Count]+":")
        DialogAddEditBox("DlgId";EditID;65;CurrRow;150;13;33;DefVars[Count];80)
        CurrRow:=CurrRow+RowSpacing                     //Increase row pointer
EndFor
Display(On!)
Call(HideWaitMessage@)
```

```
DialogDisplay("DlgId";"EB1")
ButtonPushed:=MacroDialogResult
DialogDestroy("DlgId")
Call(ShowWaitMessage@)
Display(Off!)
If(ButtonPushed=1)          //OK
For(Count;1;Count<=NumDefFlds;Count+1)
        If(DefVars[Count]="")
                MsgBoxCaption:="Confirmation"
                MsgBoxMessage:="One or more originator information fields are blank.  Is this correct?"
                Call(Confirm@)
                If(MsgBoxResult=7)                  //NO
                        Go(FirstDlg@)
                Else
                        If(MsgBoxResult=6)          //YES
                                Go(Cont@)
                        EndIf
                EndIf
        EndIf
EndFor
        Label(Cont@)
        InfoType:=1
        MacroStatusPrompt(On!;"Saving the originator information")
        Call(CheckForPipe@)
        If(BadEntryFound)
                Go(FirstDlg@)
        EndIf
        Call(SaveOriginator@)
        Return
EndIf
If(ButtonPushed=2)          //Cancel
        Call(KillWaitMessage@)
        Quit
EndIf

Label(NotFirstDlg@)
CurrRow:=FirstRow
//Create a temporary copy of originator information
For(Count;1;Count<=NumDefFlds;Count+1)
        EditVars[Count]:=DefVars[Count]
EndFor
DialogDefine("DlgId";50;50;227;164+RowSpacing;19;"Edit the Originator Information")
For(Count;1;Count<=NumDefFlds;Count+1)          //For number of prompts possible
        NumStr(StringNum;0;Count)
        TextID:="TB"+StringNum
        EditID:="EB"+StringNum
        DialogAddText("DlgId";TextID;8;CurrRow+2;49;10;1;DefFlds[Count]+":")
        DialogAddEditBox("DlgId";EditID;65;CurrRow;150;13;33;DefVars[Count];80)
        CurrRow:=CurrRow+RowSpacing  //Increase row pointer
EndFor
DialogAddText("DlgId";"CT";8;CurrRow+2;210;10;4;"Current Template:" +?CurrentTemplate)
DialogAddPushButton("DlgId";"PB1";8;129+RowSpacing;68;13;0;"&Save to Template")
Display(On!)
MacroStatusPrompt(On!;"Edit the originator information")
Call(HideWaitMessage@)
DialogDisplay("DlgId";"EB1")
ButtonPushed:=MacroDialogResult
DialogDestroy("DlgId")
Call(ShowWaitMessage@)
Display(Off!)
If(ButtonPushed="PB1")              // Save as Default
```

```
        InfoType:=1
        MacroStatusPrompt(On!;"Saving the originator information")
        Call(CheckForPipe@)
        If(BadEntryFound)
                Go(NotFirstDlg@)
        EndIf
        Call(SaveOriginator@)
        Return
EndIf
If(ButtonPushed=1)          //OK
        MacroStatusPrompt(On!;"Please wait...")
        Return
EndIf
If(ButtonPushed=2)          // Cancel
        MacroStatusPrompt(On!;"Please wait...")
        For(Count;1;Count<=NumDefFlds;Count+1)
                DefVars[Count]:=EditVars[Count]
        EndFor
        Return
EndIf

Label(SaveOriginator@)
AbbrString:=""
For(Count;1;Count<=NumDefFlds-1;Count+1)
        AbbrString:=AbbrString+DefVars[Count]+DelimiterChar       // Build AbbrCreate string
EndFor
AbbrString:=AbbrString+DefVars[Count]                          // Assign last field
If(Not(FirstTime))
        AbbreviationDelete("Originator Information";0)         // Delete from default template
EndIf
AbbreviationCreate("Originator Information";0;AbbrString)    // Create originator abbr. in default
Return
//***************************************************************

//***************************************************************
//*        ROUTINE NAME:  GetPrompts
//*        INPUT VARIABLES:  None
//*        OUTPUT VARIABLES:  AddPrompts - template prompts; NumOfPrompts - Number of template prompts.
//*        DESCRIPTION:  Puts originator information into variables.  Prompts for originator information if none is
found.
//***************************************************************
Label(GetPrompts@)
MacroStatusPrompt(On!;"Scanning template")
AbbrName:="Template Prompts"
TemplateType:=0
InfoType:=2
Call(ParseAbbr@)
If(NoAbbr)
        NumOfPrompts:=0
Else
        NumOfPrompts:=NumOfFields
EndIf
Return
//***************************************************************

//***************************************************************
//*        ROUTINE NAME:  DisplayPrompts
//*        INPUT VARIABLES:  NumOfPrompts - number prompts in current template; AddPrompts - template
prompts.
//*        OUTPUT VARIABLES:  AddVars - contents of edit boxes.
//*        DESCRIPTION:  Displays the prompts in the current template.
```

```
//*************************************************************
Label(DisplayPrompts@)
For(Count;2;Count<=NumOfPrompts;Count+2)        // Check for any lookup fields
        If(AddPrompts[Count]<>0)
                ButtonText2:="&Address Book"
                Go(ReDisplay@)
        EndIf
EndFor
Label(ReDisplay@)
DlgTitle:="Template Information"
ButtonText1:="&Originator Info"
InfoType:=2
PromptText:="Enter template information"
Call(AutoDlg@)
If(ButtonPushed=1)                  //OK
        For(Count;1;Count<=NumOfPrompts/2;Count+1)
                If(AddVars[Count]="")
                        MsgBoxMessage:="One or more of the template items are blank. Is this correct?"
                        Call(Confirm@)
                        If(MsgBoxResult=6)        //Yes
                                Return
                        Else
                                Go(ReDisplay@)
                        EndIf
                EndIf
        EndFor
EndIf
If(ButtonPushed=2)                  //Cancel
        Go(Stop@)
EndIf
If(ButtonPushed="PB1")              //Originator Information
        Call(SetOriginator@)
        Go(ReDisplay@)
EndIf
If(Exists(ButtonText2)=1)           //Lookup
        If(ButtonPushed="PB2")
                Call(NameSelectDlg@)
                If(SelectedName<>"")
                        For(Count;2;Count<=NumOfPrompts;Count+2)
                                If(AddPrompts[Count]<>0)
                                        AddVars[Count/2]:=LookVar[AddPrompts[Count]]
                                EndIf
                        EndFor
                EndIf
                Go(ReDisplay@)
        EndIf
EndIf
Return
//*************************************************************

//*************************************************************
//        ROUTINE NAME: AutoDlg
//        INPUT VARIABLES: DLGTitle; ButtonText1(Optional); ButtonText2 (Optional);  PromptText;
NumOfPrompts
//        OUTPUT VARIABLES: ButtonPushed: 1=OK, 2=Cancel, "PB1"=Button 1, "PB2"=Button 2
//        DESCRIPTION:  Automatically sizes a dialog based on the number of elements inside the specified array
variable.
//*************************************************************
Label(AutoDlg@)
Call(WideCalc@)          //Another routine to determine width of dialog.
CurrRow:=FirstRow
```

```
DlgHeight:=((NumOfPrompts/2)*RowSpacing)+45
If(FirstTime)
	DialogDefine("DlgId";50;50;HFac+179;DlgHeight;17;DLGTitle)
Else
	DialogDefine("DlgId";50;50;HFac+179;DlgHeight;19;DLGTitle)
EndIf
If(Exists(ButtonText1)=1)
	DialogAddPushButton("DlgId";"PB1";8;DlgHeight-35;53;13;0;ButtonText1)
EndIf
If(Exists(ButtonText2)=1)
	DialogAddPushButton("DlgId";"PB2";64;DlgHeight-35;53;13;0;ButtonText2)
EndIf
For(Count;1;Count<=NumOfPrompts;Count+2)	//For number of prompts possible
	NumStr(StringNum;0;Count)
	TextID:="TB"+StringNum
	EditID:="EB"+StringNum
	DialogAddText("DlgId";TextID;8;CurrRow+2;HFac+4;10;1;AddPrompts[Count]+":")
	DialogAddEditBox("DlgId";EditID;HFac+17;CurrRow;150;13;33;AddVars[(Count DIV 2)+1];80)
	CurrRow:=CurrRow+RowSpacing		//Increase row pointer
EndFor
MacroStatusPrompt(On!;PromptText)
Display(On!)
Call(HideWaitMessage@)
DialogDisplay("DlgId";"EB1")
ButtonPushed:=MacroDialogResult
DialogDestroy("DlgId")
Call(ShowWaitMessage@)
Display(Off!)
Return

Label(WideCalc@)
HFac:=0
For(Count;1;Count<=NumOfPrompts;Count+2)
	StrLen(Len;AddPrompts[Count])
	If(Len>HFac)
		HFac:=Len
	EndIf
EndFor
If((HFac<10) And (Exists(ButtonText2)>0))
	HFac:=10
EndIf
HFac:=HFac*4	//Four dialog units per character
Return
//************************************************************

//************************************************************
//	ROUTINE NAME: ParseAbbr
//	INPUT VARIABLES: TemplateType: 0=Current 1=Default 2=Supplemental; AbbrName;  InfoType:
Default=1 and Additional=2
//	RETURN VARIABLES: NoAbbr (If abbreviation not found); NoContent (If abbreviation was empty);
DefVars, or AddVars depending on the EditVar input variable; NumOfFields.
//	DESCRIPTION: This routine extracts the contents of a delimited abbreviation.  The separate fields of the
abbreviation must be separated by a pipe symbol.
//************************************************************
Label(ParseAbbr@)
NumOfFields:=0
NoAbbr:=False
NoContent:=False
// Get number of abbreviations in template.
GetData(NumOfAbbrs;Abbreviation!;Count!;TemplateType)
If(NumOfAbbrs=0)			// If template contains no abbreviations,
```

```
        NoAbbr:=True            // set flag,
        Discard(NumOfAbbrs)
        Return                  // go back.
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!;TemplateType;CurrAbbrNum)
        If(CurrAbbrName=AbbrName)
                // Get contents of Abbreviation
                GetData(PullString;Abbreviation!;Data!;TemplateType;CurrAbbrNum)
                Go(ParseIt@)
        EndIf
EndFor
NoAbbr:=True
Go(EndAbbr@)

Label(ParseIt@)
If(PullString="")
        NoContent:=True
        Go(EndAbbr@)
EndIf
Label(PullField@)
StrPos(DelPos;DelimiterChar;PullString)
If(DelPos=0)
        Go(LastField@)
EndIf
// Assign the field to a variable
NumOfFields:=NumOfFields+1
If(InfoType=1)
        SubStr(DefVars[NumOfFields];1;DelPos-1;PullString)
Else
        SubStr(AddPrompts[NumOfFields];1;DelPos-1;PullString)
EndIf
// Cut assigned field from main string
StrLen(Len;PullString)
SubStr(PullString;DelPos+1;Len-DelPos;PullString)
Go(PullField@)

// Assign last field
Label(LastField@)
NumOfFields:=NumOfFields+1
If(InfoType=1)
        DefVars[NumOfFields]:=PullString
Else
        AddPrompts[NumOfFields]:=PullString
EndIf
Label(EndAbbr@)
Discard(NumOfAbbrs;CurrAbbrNum;CurrAbbrName;PullString;DelPos;Len)
Return
//***************************************************************

//***************************************************************
//*      ROUTINE NAME:  Replace
//*      INPUT VARIABLES:  NumDefFlds; DefFlds; DefVars; NumOfPrompts; AddPrompts; AddVars
//*      OUTPUT VARIABLES:  None
//*      DESCRIPTION:  Places information from dialogs into the current template.
//***************************************************************
Label(Replace@)
MacroStatusPrompt(On!;"Please wait...placing information in template")
For(Count;1;Count<=NumDefFlds;Count+1)
```

```
        PosDocVeryTop()
        OnNotFound(NextOriginator@)
        SearchString("<"+DefFlds[Count]+">")
        ReplaceString(DefVars[Count])
        ReplaceForward(Extended!)
        While(?Substructure)
                SubstructureExit
        EndWhile
        Label(NextOriginator@)
EndFor
If(NumOfPrompts<>0)
        For(Count;1;Count<=NumOfPrompts;Count+2)
                PosDocVeryTop()
                OnNotFound(NextAdditional@)
                SearchString("["+AddPrompts[Count]+"]")
                ReplaceString(AddVars[(Count DIV 2)+1])
                ReplaceForward(Extended!)
                While(?Substructure)
                        SubstructureExit
                EndWhile
                Label(NextAdditional@)
        EndFor
EndIf
Return
//*****************************************************************

//*****************************************************************
//*      ROUTINE NAME:  CheckForPipe
//*      INPUT VARIABLES:  None
//*      OUTPUT VARIABLES:  BadEntryFound
//*      DESCRIPTION:  This routine will check through the strings created in the edit dialog for DelimiterChar.  If
it is found, a prompt will ask the user to remove it from the offending string.
//*****************************************************************
Label(CheckForPipe@)
BadEntryFound:=False
For(Count; 1; Count<=NumDefFlds; Count+1)
        StrPos(DelPos;DelimiterChar;DefVars[Count])
        If(DelPos<>0)
                StringWithBadChar:=DefFlds[Count]
                BadEntryFound:=True
                MsgBoxCaption:="Illegal Character in "+StringWithBadChar+" Field"
                MsgBoxMessage:="You cannot use the pipe character (vertical bar) in any of the information.
                Please remove the character from the "+StringWithBadChar+" field."
                MsgBoxFlags:=48
                Call(HideWaitMessage@)
                Call(MessageBox@)
                Call(ShowWaitMessage@)
                Return
        EndIf
EndFor
Return
//*****************************************************************

//*****************************************************************
//*      ROUTINE NAME:  Confirm
//*      INPUT VARIABLES:   MsgBoxMessage
//*      OUTPUT VARIABLES:  MsgBoxResult  6=Yes  7=No
//*      DESCRIPTION:  Checks for blank fields from user input and asks confirmation.
//*****************************************************************
Label(Confirm@)
MsgBoxCaption:="Confirmation"
```

```
MsgBoxFlags:=36
Call(HideWaitMessage@)
Call(MessageBox@)
Call(ShowWaitMessage@)
Return

//*************************************************************
//*       ROUTINE NAME:  NameSelectDlg
//*       INPUT VARIABLES:   NameSelectDlgTitle; You can also set SelectedName with a default value.
//*       OUTPUT VARIABLES:  NameSelectDlgResult; SelectedName
//*       DESCRIPTION:  This routine will present a dialog to the user for the selection of a name from the
database.
//*************************************************************
Label(NameSelectDlg@)
If(DialogCreated=False)
        Call(CreateSelectDlg@)
EndIf
MacroStatusPrompt(On!;"Select or create an address book record")
Display(On!)
Call(HideWaitMessage@)
DialogDisplay("SelectionDlg";"List")
NameSelectDlgResult:=MacroDialogResult
Call(ShowWaitMessage@)
Display(Off!)
If(NameSelectDlgResult="Be")      // Edit
        If(SelectedName="")
                Go(SelectionErrorPrompt@)
        EndIf
        Call(EditInformation@)
        Go(NameSelectDlg@)
Else
        If(NameSelectDlgResult="Bn")      // Create
                Call(NewNameDlg@)
                Go(NameSelectDlg@)
        Else
                If(NameSelectDlgResult="Bd")      //Delete
                        If(SelectedName="")
                                Go(SelectionErrorPrompt@)
                        EndIf
                        Call(DeleteRecord@)
                        Go(NameSelectDlg@)
                Else
                        If(NameSelectDlgResult="Bs")      // Select
                                If(SelectedName="")
                                        Go(SelectionErrorPrompt@)
                                Else
                                        Call(RetrieveInformation@)
                                        Return
                                EndIf
                        Else                                      // Close
                                MacroStatusPrompt(On!;"Please wait...")
                                SelectedName:=""
                                Call(ClearFields@)
                                Return
                        EndIf
                EndIf
        EndIf
EndIf
Label(SelectionErrorPrompt@)
MsgBoxCaption:="Error - No Record Selected"
MsgBoxMessage:="You must select a record to use this option."
```

```
MsgBoxFlags:=48
Call(HideWaitMessage@)
Call(MessageBox@)
Call(ShowWaitMessage@)
Go(NameSelectDlg@)
//****************************************************************

//****************************************************************
//*       ROUTINE NAME:  DestroySelectDlg@
//*       INPUT VARIABLES:  DialogCreated
//*       OUTPUT VARIABLES:  None
//*       DESCRIPTION:  Destroys the name selection dialog.
//****************************************************************
Label(DestroySelectDlg@)
If(DialogCreated)
        DialogDestroy("SelectionDlg")
        DialogCreated:=False
EndIf
Return
//****************************************************************

//****************************************************************
//*       ROUTINE NAME:  CreateSelectDlg@
//*       INPUT VARIABLES:  None
//*       OUTPUT VARIABLES:  DialogCreated; hwndList
//*       DESCRIPTION:  Creates the name selection dialog.
//****************************************************************
Label(CreateSelectDlg@)
DialogCreated:=True
MacroStatusPrompt(On!; "Please wait...retrieving address book from template")
GetData(RecordCount; Abbreviation!; Count!; 1)
DialogDefine("SelectionDlg";50;50;198;145;16; "Address Book")
DialogAddListBox("SelectionDlg";"List";8;11;130;84;3;SelectedName)
For(RecordNumber;1;RecordNumber<=RecordCount;RecordNumber+1)
        GetData(RecordName; Abbreviation!; Name!; 1; RecordNumber)
        SubStr(DelPos;1;4;RecordName)
        If(DelPos=UniqueIdentifier)
                RecordNameLength:=StrLen(RecordName)
                SubStr(RecordName;5; RecordNameLength; RecordName)
                DialogAddListItem("SelectionDlg";"List";RecordName)
        EndIf
EndFor
DialogAddHLine("SelectionDlg"; "H1"; 6; 103; 178)
DialogAddText("SelectionDlg"; "T1"; 8; 105; 176; 20; 0; "Select a name from the list and press a function button, or
press Add to create a new record.")
DialogAddPushButton("SelectionDlg";"Bs";146;9;38;13;1;"&Select")
DialogAddPushButton("SelectionDlg";"Be";146;27;38;13;0;"&Edit")
DialogAddPushButton("SelectionDlg";"Bn";146;45;38;13;0;"&Add")
DialogAddPushButton("SelectionDlg";"Bd";146;63;38;13;0;"&Delete")
DialogAddPushButton("SelectionDlg";"Bc";146;81;38;13;0;"&Close")
DialogHandle(hwndList; "SelectionDlg";"List")
        // Select the first name in the list
DLLCall(Userlink;"SendMessage";nVoid:INTEGER;{LoWord(hwndList);LoWord(1031); LoWord(0);0})
Return
//****************************************************************

//****************************************************************
//*       ROUTINE NAME:  DeleteRecord
//*       INPUT VARIABLES:  SelectedName
//*       OUTPUT VARIABLES:  None
//*       DESCRIPTION:  This routine will delete a record of the database after checking for confirmation.
```

```
//*****************************************************************
Label(DeleteRecord@)
MsgBoxCaption:="Confirm Deletion"
MsgBoxMessage:="Delete """+SelectedName+"""?"
MsgBoxFlags:=36
Call(HideWaitMessage@)
Call(MessageBox@)
Call(ShowWaitMessage@)
If(MsgBoxResult=6)
        // get the number of the current selection
    DLLCall(Userlink;"SendMessage";nIndex:INTEGER;
    {LoWord(hwndList);LoWord(1033);LoWord(0);0})
        // delete it from the listbox
    DLLCall(Userlink;"SendMessage";nVoid:INTEGER;
    {LoWord(hwndList);LoWord(1027);LoWord(nIndex);0})
    If(nIndex>0)
        NextSel:=nIndex-1
    Else
        NextSel:=0
    EndIf
        // select the next item up
    DLLCall(Userlink;"SendMessage";nVoid:INTEGER;{LoWord(hwndList);LoWord(1031);
    LoWord(NextSel);0})
    AbbreviationDelete(UniqueIdentifier+SelectedName; 1)
    Discard(SelectedName)
EndIf
Return
//*****************************************************************

//*****************************************************************
//*     ROUTINE NAME:  RetrieveInformation
//*     INPUT VARIABLES:  SelectedName
//*     OUTPUT VARIABLES:  LookVar[]; RecordExists
//*     DESCRIPTION:  This Routine will read in the information for a selected record if the record exists.  If the
record does not exist, the variable RecordExists is set to False, and SingleString will be set to a null string.
//*****************************************************************
Label(RetrieveInformation@)
MacroStatusPrompt(On!; "Please wait...retrieving information for "+SelectedName)
RecordExists:=False
GetData(RecordCount; Abbreviation!; Count!; 1)
For(RecordNumber;1;RecordNumber<=RecordCount;RecordNumber+1)
    GetData(RecordName; Abbreviation!; Name!; 1; RecordNumber)
    If(RecordName=UniqueIdentifier+SelectedName)
        RecordExists:=True
        GetData(SingleString; Abbreviation!; Data!; 1; RecordNumber)
        Call(ParseSingleString@)
        Return
    Endif
EndFor
Call(ClearFields@)
Return
//*****************************************************************

//*****************************************************************
//*     ROUTINE NAME:  EditInformation
//*     INPUT VARIABLES:  SelectedName
//*     OUTPUT VARIABLES:  LookVar[]
//*     DESCRIPTION:  Edits an entry in the database book and prompts user for the input for that entry.
//*****************************************************************
Label(EditInformation@)
Call(RetrieveInformation@)
```

```
MacroStatusPrompt(On!;"Edit address book record")
Call(InfoEditDlg@)
If(InfoEditDlgResult=1)
        Call(MakeSingleString@)
        If(RecordExists)  // If the record already existed, then first delete it.
                AbbreviationDelete(UniqueIdentifier+SelectedName; 1)
        EndIf
        AbbreviationCreate(UniqueIdentifier+SelectedName; 1; SingleString)
Endif
Return
//****************************************************************

//****************************************************************
//*      ROUTINE NAME:  NewNameDlg
//*      INPUT VARIABLES:  None
//*      OUTPUT VARIABLES:  LookVar[]
//*      DESCRIPTION:  Creates a new entry in the database book and prompts user for the input for that entry.
//****************************************************************
Label(NewNameDlg@)
Call(ClearFields@)
MacroStatusPrompt(On!;"Create address book record")
Call(InfoEditDlg@)
If(InfoEditDlgResult=2)
        Return
EndIf
Call(MakeSingleString@)
NameToAdd:=LookVar[1]
If(NameToAdd="")
        Go(TryANewName@)
EndIf
Label(WriteNewName@)
OnError(TryANewName@)
AbbreviationCreate(UniqueIdentifier+NameToAdd; 1; SingleString)
OnError(Error@)
SelectedName:=NameToAdd
DialogAddListItem("SelectionDlg"; "List"; NameToAdd)
Return

Label(TryANewName@)
OnError(Error@)
DialogDefine("NameDlg";50;50;172;94;19;"Record Name")
DialogAddEditBox("NameDlg";"EditBox1";8;8;150;13;33;NameToAdd;27 )
If(NameToAdd="")
        DialogAddText("NameDlg";"Text1";8;27;150;30;1;"Please enter a unique name for this record.")
Else
        DialogAddText("NameDlg";"Text1";8;27;150;30;1;"A record with this name already exists in the Address
        Book.  Please enter a new name for this record.")
EndIf
Display(On!)
Call(HideWaitMessage@)
DialogDisplay("NameDlg";"EditBox1")
Button:=MacroDialogResult
DialogDestroy("NameDlg")
Call(ShowWaitMessage@)
Display(Off!)
If(Button<>1)     //          check for escape key or cancel button
        Return
Endif
If(NameToAdd="")
        Go(TryANewName@)
Endif
```

```
Go(WriteNewName@)
//***************************************************************

//***************************************************************
//*       ROUTINE NAME:  InfoEditDlg
//*       INPUT VARIABLES:  Any of the string variables may be set to a default value before calling this function.
//*       OUTPUT VARIABLES:  LookVar[]; InfoEditDlgResult
//*       DESCRIPTION:  This Routine will bring up a dialog to edit all of the portions of an entry into the
database.
//***************************************************************
Label(InfoEditDlg@)
DialogDefine("ToDlg"; 50; 50; 230; 182; 19; "Address Book")
DialogAddText("ToDlg"; "Tn"; 8; 10; 49; 10; 1; "&Name:")
DialogAddEditBox("ToDlg"; "En"; 65; 8; 150; 13; 33; LookVar[1]; 80)
DialogAddText("ToDlg"; "Ts"; 8; 27; 49; 10; 1; "&Salutation:")
DialogAddEditBox("ToDlg"; "Es"; 65; 25; 150; 13; 33; LookVar[2]; 80)
DialogAddText("ToDlg"; "Tt"; 8; 44; 49; 10; 1; "&Title:")
DialogAddEditBox("ToDlg"; "Et"; 65; 42; 150; 13; 33; LookVar[3]; 80)
DialogAddText("ToDlg"; "Tc"; 8; 61; 49; 10; 1; "&Company:")
DialogAddEditBox("ToDlg"; "Ec"; 65; 59; 150; 13; 33; LookVar[4]; 80)
DialogAddText("ToDlg"; "Ta"; 8; 78; 49; 10; 1; "&Address:")
DialogAddEditBox("ToDlg"; "Ea"; 65; 76; 150; 13; 33; LookVar[5]; 80)
DialogAddText("ToDlg"; "Tz"; 8; 95; 49; 10; 1; "City, State &Zip:")
DialogAddEditBox("ToDlg"; "Ez"; 65; 93; 150; 13; 33; LookVar[6]; 80)
DialogAddText("ToDlg"; "Tp"; 8; 112; 49; 10; 1; "Tele&phone:")
DialogAddEditBox("ToDlg"; "Ep"; 65; 110; 60; 13; 33; LookVar[7]; 20)
DialogAddText("ToDlg"; "Tf"; 133; 112; 14; 10; 1; "&Fax:")
DialogAddEditBox("ToDlg"; "Ef"; 155; 110; 60; 13; 33; LookVar[8]; 20)
DialogAddText("ToDlg"; "Ti"; 8; 129; 68; 10; 1; "Account/&ID Number:")
DialogAddEditBox("ToDlg"; "Ei"; 84; 127; 131; 13; 33; LookVar[9]; 80)
Display(On!)
Call(HideWaitMessage@)
DialogDisplay("ToDlg"; "En")
InfoEditDlgResult:=MacroDialogResult
DialogDestroy("ToDlg")
Call(ShowWaitMessage@)
Display(Off!)
If(InfoEditDlgResult=1)
        Call(CheckForBadEntry@)
        If(BadEntryFound)
                Go(InfoEditDlg@)
        EndIf
EndIf
Return
//***************************************************************

//***************************************************************
//*       ROUTINE NAME:  CheckForBadEntry
//*       INPUT VARIABLES:  LookVar[]
//*       OUTPUT VARIABLES:  BadEntryFound
//*       DESCRIPTION:  This routine will check through the strings created in the edit dialog for DelimiterChar.  If
it is found, a prompt will ask the user to remove it from the offending string.
//***************************************************************
Label(CheckForBadEntry@)
BadEntryFound:=False
StrPos(LocOfBad; DelimiterChar; LookVar[1])
StringWithBadChar:="Name"
If(LocOfBad=0)
 StrPos(LocOfBad; DelimiterChar; LookVar[2])
 StringWithBadChar:="Salutation"
 If(LocOfBad=0)
```

```
  StrPos(LocOfBad; DelimiterChar; LookVar[3])
  StringWithBadChar:="Title"
  If(LocOfBad=0)
    StrPos(LocOfBad; DelimiterChar; LookVar[4])
    StringWithBadChar:="Company"
    If(LocOfBad=0)
      StrPos(LocOfBad; DelimiterChar; LookVar[5])
      StringWithBadChar:="Address"
      If(LocOfBad=0)
        StrPos(LocOfBad; DelimiterChar; LookVar[6])
        StringWithBadChar:="City, State Zip"
        If(LocOfBad=0)
          StrPos(LocOfBad; DelimiterChar; LookVar[7])
          StringWithBadChar:="Telephone"
          If(LocOfBad=0)
            StrPos(LocOfBad; DelimiterChar; LookVar[8])
            StringWithBadChar:="Fax"
            If(LocOfBad=0)
              StrPos(LocOfBad; DelimiterChar; LookVar[9])
              StringWithBadChar:="Account/ID"
              If(LocOfBad=0)
                Return
              EndIf
            EndIf
          EndIf
        EndIf
      EndIf
    EndIf
  EndIf
 EndIf
EndIf
BadEntryFound:=True
MsgBoxCaption:="Illegal Character in "+StringWithBadChar+" Field"
MsgBoxMessage:="You cannot use the pipe character (vertical bar) in any of the information.  Please remove the
character from the "+StringWithBadChar+" field."
MsgBoxFlags:=48
Call(HideWaitMessage@)
Call(MessageBox@)
Call(ShowWaitMessage@)
Return
//***************************************************************

//***************************************************************
//*        ROUTINE NAME:  MakeSingleString
//*        INPUT VARIABLES:  LookVar[]
//*        OUTPUT VARIABLES:  SingleString
//*        DESCRIPTION:  This function takes all of the separate strings from the dialog and concatenates them
together with separation characters for storage in the abbreviations.
//***************************************************************
Label(MakeSingleString@)
MacroStatusPrompt(On!; "Please wait...writing record to address book")
SingleString:=LookVar[1]+DelimiterChar+LookVar[2]+DelimiterChar+LookVar[3]+DelimiterChar+LookVar[4]+D
elimiterChar+LookVar[5]+DelimiterChar+LookVar[6]+DelimiterChar+LookVar[7]+DelimiterChar+LookVar[8]+De
limiterChar+LookVar[9]
Return
//***************************************************************

//***************************************************************
//*        ROUTINE NAME:  ParseSingleString
//*        INPUT VARIABLES:  SingleString
//*        OUTPUT VARIABLES:  LookVar[]
```

```
//*       DESCRIPTION:  This routine will take the variable SingleString and will parse out the individual variables
for the edit dialog, etc.
//*****************************************************************
Label(ParseSingleString@)
PullString:=SingleString
For(LoopCounter; 1; LoopCounter<=MaxAdrsBookVar; LoopCounter+1)
        StrPos(NextDelimLoc; DelimiterChar; PullString)
        StrLen(PullStringLength; PullString)
        If(NextDelimLoc=0)
                NextDelimLoc:=PullStringLength+1
        EndIf
        SubStr(LookVar[LoopCounter]; 1; NextDelimLoc-1; PullString)
        SubStr(PullString; NextDelimLoc+1; PullStringLength; PullString)
EndFor
Return
//*****************************************************************

//*****************************************************************
//*       ROUTINE:  ClearFields
//*       INPUT VARIABLES:  LookVar[]
//*       OUTPUT VARIABLES:  LookVar[]
//*       DESCRIPTION:  Sets contents of LookVar[] to all null strings.
//*****************************************************************
Label(ClearFields@)
LookVar[1]:=""
LookVar[2]:=""
LookVar[3]:=""
LookVar[4]:=""
LookVar[5]:=""
LookVar[6]:=""
LookVar[7]:=""
LookVar[8]:=""
LookVar[9]:=""
Return
//*****************************************************************

//*****************************************************************
//*       ROUTINE NAME:  Error@
//*       INPUT VARIABLES:  None
//*       OUTPUT VARIABLES:  None
//*       DESCRIPTION:  Displays a message box when an unknown error occurs.
//*****************************************************************
Label(Error@)
MsgBoxFlags:=16
MsgBoxCaption:="Error"
MsgBoxMessage:="An unknown error has occurred.  The macro will now terminate."
Call(MessageBox@)
Go(Stop@)
//*****************************************************************

//*****************************************************************
//*       ROUTINE NAME:  MessageBox
//*       INPUT VARIABLES:  MsgBoxCaption; MsgBoxMessage; MsgBoxFlags
//*       OUTPUT VARIABLES:  MsgBoxResult
//*       DESCRIPTION:  Displays a message box.
//*****************************************************************
Label(MessageBox@)
Display(On!)
DllLoad(MsgBoxLink; "shwin20.dll")
DllCall(MsgBoxLink; "WmbMsgBox"; MsgBoxResult:WORD; {LoWord(0); LoWord(0);
AnsiString(MsgBoxCaption); AnsiString(MsgBoxMessage); MsgBoxFlags; 0; LoWord(0)})
```

```
DllFree(MsgBoxLink)
Display(Off!)
Return
//***************************************************************

//***************************************************************
//*       ROUTINE:  WaitMessage
//*       INPUT VARIABLES:  WaitMessage
//*       OUTPUT VARIABLES:  WaitMessageResult
//*       DESCRIPTION:  Presents a Please Wait message to user.  Call KillWaitMessage to turn it off.
//***************************************************************
Label(WaitMessage@)
WaitTitle:="Autofila Macro"
DialogDefine(WaitTitle; 50; 50; 160; 60; 16; WaitTitle)
DialogAddText(WaitTitle; "T3"; 8; 8; 138; 10; 4; Disclaimer)
DialogAddText(WaitTitle; "T2"; 8; 23; 138; 10; 4; WaitMessage)
DialogHandle(hwnd; WaitTitle;)
DllLoad(UserLink; "User.exe")
DllCall(UserLink; "ShowWindow"; WaitMessageResult:BOOL; {LoWord(hwnd); Loword(1)})
Return
//***************************************************************


//***************************************************************
//*       ROUTINE:  KillWaitMessage
//*       INPUT VARIABLES:  none
//*       OUTPUT VARIABLES:  none
//*       DESCRIPTION:  Turns off wait message.
//***************************************************************
Label(KillWaitMessage@)
DialogDestroy(WaitTitle)
DllFree(UserLink)
Return
//***************************************************************

//***************************************************************
//*       ROUTINE:  HideWaitMessage
//*       INPUT VARIABLES:  none
//*       OUTPUT VARIABLES:  none
//*       DESCRIPTION:  Hides wait message.
//***************************************************************
Label(HideWaitMessage@)
DllCall(UserLink; "ShowWindow"; WaitMessageResult:BOOL; {LoWord(hwnd); Loword(0)})
Return
//***************************************************************

//***************************************************************
//*       ROUTINE:  ShowWaitMessage
//*       INPUT VARIABLES:  none
//*       OUTPUT VARIABLES:  none
//*       DESCRIPTION:  Hides wait message.
//***************************************************************
Label(ShowWaitMessage@)
DllCall(UserLink; "ShowWindow"; WaitMessageResult:BOOL; {LoWord(hwnd); Loword(1)})
Return
//***************************************************************
```