```
//*********************************************************
//*       MACRO NAME:    Print2WinFax Ver 0.4, 12 February 1997
//*       AUTHOR:  J Filshie
//*       PURPOSE:  To fax a document via WinFax Pro. It will look for styles
//*                 containing sending data and, if they are present, it will use DDE.
//*                 Otherwise it will use the Winfax printer driver.
//*********************************************************
Application (A1; "WordPerfect"; Default; "US")
//*************************************************
//        MAIN PROGRAM BEGIN
//*************************************************
If (NOT CheckVersion (7;0))
        Quit
EndIf
CheckDoc
Initialize
Debug = False // setting this to True will display some progress information
PosDocVeryTop()
DisplayHiddenText
If (GetFaxField(FaxFields[1;1]) = "")
        UseDDE = False
        RestoreHiddenTextState
        PrintDlg()
Else
        UseDDE = True
        GetFieldInfo
        RestoreHiddenTextState
        WaitMsgInit
        StartWinFax
        If (NOT StartConversation())
                Cleanup
                Quit
        EndIf
        DDEExecute (hControl; "GoIdle")
        WaitForPrinter
        If (Debug)
                ShowDetails
        EndIf
        PokeInfo
        Print (FullDocument!)
        DDEExecute (hControl; "GoActive")
EndIf
If (UseDDE)
        WaitMsgDisplay ("Please wait while WinFax processes the document")
        WaitForPrinter
        DDETerminateAll
```

```
        CleanUp
        OnCancel (UserQuit)
        Prompt ("Print2WinFax"; "The document has been sent to WinFax"; InformationIcon! |
        NoButtons!)
        Wait (40)
        EndPrompt()
Else
        Cleanup
EndIf
Label (UserQuit)
Return //from Print2WinFax
//*****************************************************
//      MAIN PROGRAM END
//*****************************************************

//*********************************************************
//      PROCEDURE NAME:  CheckDoc
//      INPUT VARIABLES:  none
//      OUTPUT VARIABLES:  none
//      DESCRIPTION: If no document on screen, quit with error message
//*********************************************************
PROCEDURE CheckDoc ()
If(?DocBlank)
        MessageBox(; "ERROR"; "You should have a document on the screen before Faxing";
        IconStop!)
        Quit
EndIf
ENDPROC


//*********************************************************
//      PROCEDURE NAME:  Initialize
//      INPUT VARIABLES:  none
//      OUTPUT VARIABLES:  various see below
//      DESCRIPTION: Main initialization procedure
//              also selects the WinFax driver.
//*********************************************************
PROCEDURE Initialize ()
GLOBAL FaxPrinter:="WinFax" // Change this to the name of the Fax driver
GLOBAL CurrentPrinter:=?CurrentPrinter
GLOBAL CR := NToc(63754)
GLOBAL hWinFax; Debug = False
GLOBAL UseDDE = False
GLOBAL ServiceName = "FAXMNG32"
GLOBAL hTransmit; hControl
GLOBAL sWfxKey = "Software\Delrina\WinFax\7.0\WinFax"
```

```
GLOBAL ExeName = "faxmng32.exe"
GLOBAL MaxFields = 8
GLOBAL FaxFields[MaxFields;2]
If (NOT SelectFaxDriver (FaxPrinter))
        MessageBox(; "ERROR"; "The Fax driver ""^0"" was not found or could not be selected"
        +CR+ "Check the printer list to see if it exists and change the name in this macro if it is
        incorrect"; IconStop! | HasParameters!; FaxPrinter)
        RestoreDefaultPrinter
        RestoreHiddenTextState
        Quit
Else
        PrintDestination (Destination: DriverPort!)
EndIf
FaxFields[1;1] = "WfxFaxNum"
FaxFields[2;1] = "WfxRecipient"
FaxFields[3;1] = "WfxTime"
FaxFields[4;1] = "WfxDate"
FaxFields[5;1] = "WfxCompany"
FaxFields[6;1] = "WfxSubject"
FaxFields[7;1] = "WfxKeyword"
FaxFields[8;1] = "WfxBillCode"
ClearFields
ENDPROC

//*********************************************************
//        PROCEDURE NAME:  ClearFields
//        INPUT VARIABLES:  none
//        OUTPUT VARIABLES:  FaxFields[n;2]
//        DESCRIPTION: Null the fax data array
//*********************************************************
PROCEDURE ClearFields ()
ForNext (Count; 1; MaxFields)
        FaxFields[Count; 2] = ""
EndFor
ENDPROC


//*********************************************************
//        FUNCTION NAME:  GetFaxField
//        INPUT VARIABLES:  StyleName
//        OUTPUT VARIABLES:  none
//        RETURN VALUE:  Fax data
//        DESCRIPTION:  Search for a paired paragraph or character style.
//                Return the text string between the begin/end codes or null.
//*********************************************************
FUNCTION GetFaxField (StyleName)
```

```
OnNotFound (NoField)
SpecificSearchStringBegin ()
StyleSystemOn (Style: StyleName)
SpecificStringEnd ()
MatchPositionAfter ()
SearchNext (SearchMode: Extended!)
If (?RightCode = 1757)          //move over style code in a para style
        PosCharNext()
EndIf
SelectMode (State: On!)
SearchString (StrgToLookFor: "[Style]")
MatchExtendSelection ()
SearchNext (SearchMode: Regular!)
sText = ?SelectedText
SelectMode (Off!)
OnNotFound ()
Return (sText)

Label (NoField)
OnNotFound ()
Return ("")
ENDFUNC

//*********************************************************
//      FUNCTION NAME:  SelectFaxDriver
//      INPUT VARIABLES:  FaxPrinter
//      OUTPUT VARIABLES:  none
//      RETURN VALUE:  Boolean
//      DESCRIPTION:  True if Fax Printer is selected
//*********************************************************
FUNCTION SelectFaxDriver (FaxPrinter)
PrinterSelectByName (FaxPrinter)
If (?CurrentPrinter = FaxPrinter)
        bReturn = True
Else
        bReturn = False
EndIf
Return (bReturn)
ENDFUNC



//*********************************************************
//      PROCEDURE NAME:  RestoreDefaultPrinter
//      INPUT VARIABLES:  CurrentPrinter (original printer)
//      OUTPUT VARIABLES:  none
```

```
//        DESCRIPTION: Reselect the original printer driver.
//*********************************************************
PROCEDURE RestoreDefaultPrinter ()
PrinterSelectByName (PrinterName: CurrentPrinter)
PrintDestination (Destination: DriverPort!)
If (?CurrentPrinter <> CurrentPrinter)
        MessageBox (;"ERROR"; "Unable to re-select the ""^0"" driver" +CR+ "You will have to
        select it manually after faxing"; IconStop!|HasParameters!; CurrentPrinter)
EndIf
Return
ENDPROC

//*********************************************************
//        PROCEDURE NAME:  GetFieldInfo
//        INPUT VARIABLES:  FaxFields[n;1]
//        OUTPUT VARIABLES:  FaxFields[n;2]
//        DESCRIPTION: Search the document for the style names listed in
//                FaxFields[n;1] and return the data in FaxFields[n;2]
//*********************************************************
PROCEDURE GetFieldInfo ()
ForNext (Count; 1; MaxFields)
        PosDocVeryTop()
        FaxFields[Count;2] = GetFaxField (FaxFields[Count;1])
EndFor
ENDPROC

//*********************************************************
//*      PROCEDURE:StartWinFax
//*      INPUT: None
//*      OUTPUT: hWinFax
//*      DESC: Check to see if WinFax is open. If not, then locate the full
//*              path in Registry and open it with Appexecute
//*********************************************************
PROCEDURE StartWinFax ()
hWinFax = AppLocate ("WinFax PRO")
If (hWinFax <> 0)
        vReturn = 0
Else
        WaitMsgDisplay ("Starting WinFax")
        ExePathKey =  RegistryOpenKey (CurrentUser!; sWfxKey)
        ExePath = RegistryQueryValue (ExePathKey; "ExePath")
        If (ExePathKey <> 0)
                If (Debug)
                        Wait (5)
                        MessageBox (vButton;"Debug - StartWinFax"; "ExePath: "
                        +ExePath;OKCancel!)
```

```
                If (vButton = 2)
                        Cleanup
                        Quit
                Endif
        EndIf
        Faxpath = ExePath + ExeName
        vReturn =AppExecuteExt (Faxpath; MinimizeNoActivate!)
        If (vReturn < 32)
                MessageBox (; "SendFax Error"; WinExecErrMess (vReturn); IconStop!)
                Cleanup
                Quit
        Else
                Repeat
                        hWinFax = AppLocate ("WinFax PRO")
                        Wait (10)
                Until (hWinFax <> 0)

                Wait (5)
                AppShow (hWinFax; MinimizeNoActivate!)
        EndIf
        WaitMsgHide
Else
        MessageBox(;"SendFax"; "Unable to locate WinFax exe path in the registry";
        IconStop!)
        Cleanup
        Quit
EndIf
EndIf
Return //from StartWinFax
ENDPROC

//*********************************************************
//      FUNCTION NAME:  WinExecErrMess
//      INPUT VARIABLES:  ErrorValue
//      RETURN VALUE:  Errormessage
//      DESCRIPTION:  Return an error message string
//*********************************************************

FUNCTION WinExecErrMess (ErrorValue)
SWITCH (ErrorValue)
        Caseof 0: ErrorMessage = "Out of memory"
        Caseof 2: ErrorMessage = "File not found"
        Caseof 3: ErrorMessage = "Path not found"
        Caseof 5: ErrorMessage = "Attempt to dynamically link a task"
        Caseof 6: ErrorMessage = "Library requires separate data segments for each task"
        Caseof 10: ErrorMessage = "Incorrect Windows version"
```

```
        Caseof 11: ErrorMessage = "Invalid .EXE file"
        Caseof 12: ErrorMessage = "OS/2 application"
        Caseof 13: ErrorMessage = "DOS 4.0 application"
        Caseof 14: ErrorMessage = "Unknown .EXE type"
        Caseof 15: ErrorMessage = "Attempt to load an earlier .EXE version in protected mode"
        Caseof 16: ErrorMessage = "Attempt to load a second instance of a .EXE"
        Caseof 17: ErrorMessage = "Attempt to load a second instance of a .EXE in large frame
EMS mode"
        Caseof 18: ErrorMessage = "Attempt to load protected mode program in real mode"
        Default:ErrorMessage = "Unknown error"
ENDSWITCH
Return (ErrorMessage)
ENDFUNC

//*********************************************************
//      PROCEDURE NAME:  PokeInfo
//      INPUT VARIABLES:  FaxFields[n;2]
//      OUTPUT VARIABLES:  None
//      DESCRIPTION: Send the above variables to Winfax with a DDEPoke
//*********************************************************
PROCEDURE PokeInfo ()
Recipient = "recipient(""" +FaxFields[1;2]+ """," +""""" +FaxFields[3;2]+ """," +"""""
+FaxFields[4;2]+ """," +""""" +FaxFields[2;2]+ """," +""""" +FaxFields[5;2]+ ""","  +"""""
+FaxFields[6;2]+ """," +""""" +FaxFields[7;2]+ """," +""""" +FaxFields[8;2]+ """)"
DDEPoke (hTransmit; "Sendfax"; Recipient)
Return          //from SendFax
ENDPROC

//*********************************************************
//      PROCEDURE NAME:  DisplayHiddenText
//      DESCRIPTION: Display hidden text in case any Wfx styles are hidden
//*********************************************************
PROCEDURE DisplayHiddenText ()
GLOBAL HiddenTextState = ?HiddenTextActive
If (HiddenTextState)
        HiddenTextShowAll (On!)
EndIf
ENDPROC

//*********************************************************
//      PROCEDURE NAME:  RestoreHiddenText
//      DESCRIPTION: Restore hidden text to original state
//*********************************************************
PROCEDURE RestoreHiddenTextState ()
If (HiddenTextState)
        HiddenTextShowAll (Off!)
```

```
EndIf
ENDPROC

//*********************************************************
//      FUNCTION NAME:  ConversationID
//      INPUT VARIABLES:  Service and Topic names
//      OUTPUT VARIABLES:  none
//      RETURN VALUE:  Conversation handle
//      DESCRIPTION:  Attempts to initiate a conversation with the Service/Topic
//              tries for up to about 5 secs to start a conversation. Returns handle or null
//*********************************************************
FUNCTION ConversationID (Service; Topic)
WaitMsgDisplay ("Initiating conversation with WinFax")
InitLoop = 0
Repeat
        InitLoop = InitLoop + 1
        Wait (5)
        DDEINITIATE (ConversationID; Service; Topic)
Until ((ConversationID <> 0) OR (InitLoop = 10))
WaitMsgHide
Return (ConversationID)
ENDFUNC


//*********************************************************
//      FUNCTION NAME:  CheckVersion
//      INPUT VARIABLES:  MajorVer; MinorVer
//      RETURN VALUE:  Bool
//      DESCRIPTION:  If the currect major/minor version is less than
//              MajorVer.MinorVer return False and Error message
//*********************************************************
FUNCTION CheckVersion (MajorVer; MinorVer)
If ((?MajorVersion <MajorVer) OR ((?MajorVersion <= MajorVer) AND (?MinorVersion <
MinorVer)))
MessageBox(; "Check Version"; "This macro will only work with WPWin ^0.^1 or later"; OK! |
DefButton1! | IconStop!|HasParameters!;{MajorVer; MinorVer})
Return (False)
else
Return (True)
Endif
ENDFUNC


//*********************************************************
//      FUNCTION NAME:  StartConversation
//      INPUT VARIABLES: none
```

```
//        OUTPUT VARIABLES:  hTransmit; hControl
//        RETURN VALUE:  Boolean
//        DESCRIPTION:  Attempts to initiate a conversation with WinFax for
//                TRANSMIT and CONTROL topics and assigns the handle
//                False if unable to start conversation
//*********************************************************
FUNCTION StartConversation ()
hTransmit = ConversationID  (ServiceName;"TRANSMIT")
hControl = ConversationID  (ServiceName;"CONTROL")
If ((hTransmit = 0) OR (hControl = 0))
        MessageBox (; "ERROR"; "Unable to establish a conversation with WinFax PRO";
        IconStop!)
        RestoreDefaultPrinter
        bReturn = False
Else
        bReturn = True
EndIf
Return (bReturn)
ENDFUNC

//*********************************************************
//        PROCEDURE NAME:  WaitForPrinter
//        DESCRIPTION: Wait for WinFax to permit sending fax information
//*********************************************************
PROCEDURE WaitForPrinter ()
Repeat
        Wait (10)
Until (DDERequest (hControl; "CanIPrint") = "Yes")
ENDPROC

//*********************************************************
//        PROCEDURE NAME:  ShowDetails
//        INPUT VARIABLES:  Fax Data
//        OUTPUT VARIABLES:  none
//        DESCRIPTION: Used for debugging. Displays the data that will be sent
//                to WinFax
//*********************************************************
PROCEDURE ShowDetails ()
Wait (5)
        MessageBox (vButton; "Debug - GetFields"; "Company: " +FaxFields[5;2]+CR+ "Name:
        " +FaxFields[2;2]+CR+ "FaxNumber: " +FaxFields[1;2]+CR+ "Subject: "
        +FaxFields[6;2]+CR+ "Time: " +FaxFields[3;2]+CR+ "Date: "+ FaxFields[4;2]+CR+
        "Keyword: "+ FaxFields[7;2]+CR+ "BillingCode: "+ FaxFields[8;2];OKCancel!)
        If (vButton = 2)
                Quit
        Endif
```

```
ENDPROC

//*********************************************************
//      PROCEDURE NAME: Cleanup
//      DESCRIPTION: Restore original conditions
//*********************************************************
PROCEDURE Cleanup ()
If (UseDDE)
        WaitMsgHide
        WaitMsgDestroy
EndIf
RestoreDefaultPrinter
RestoreHiddenTextState
ENDPROC


//*************************************
//      PROCEDURE: WaitMsgInit ()
//      PURPOSE: Initializes macro wait messages.
//*************************************
PROCEDURE WaitMsgInit ()
If (Not DoesDialogExist ("WaitMsg"))
        DialogDefine ("WaitMsg"; 20; 20; 200; 28; Percent!; "Print2WinFax")
        DialogAddText ("WaitMsg"; "WaitText"; 8; 8; 184; 10; Left!; "")
        DialogSave ("WaitMsg")
EndIf
DialogLoad ("WaitMsg"; "WordPerfect")
Return
ENDPROC

//*************************************
//      PROCEDURE: WaitMsgDisplay (Text)
//      PURPOSE: Displays a previously defined wait message dialog.
//*************************************
PROCEDURE WaitMsgDisplay (Text)
RegionSetWindowText ("WaitMsg" + ".WaitText"; Text)
DialogShow ("WaitMsg"; "WordPerfect"; WaitDlgCallBack)
ENDPROC

//*************************************
//      PROCEDURE: WaitDlgCallBack
//      PURPOSE: Callback function for the wait message.
//*************************************
PROCEDURE WaitDlgCallBack ()
If (WaitDlgCallBack[5] = 274 And WaitDlgCallBack[6] = 61536)
        Assert (CancelCondition!)
```

```
EndIf
ENDPROC

//*************************************
//      PROCEDURE: WaitMsgHide
//      PURPOSE: Hides a previously defined wait message dialog.
//*************************************
PROCEDURE WaitMsgHide ()
DialogUndisplay ("WaitMsg"; "WaitText")
ENDPROC

//*************************************
//      PROCEDURE: WaitMsgDestroy
//      PURPOSE: Destroys a previously defined wait message dialog.
//*************************************
PROCEDURE WaitMsgDestroy ()
DialogDestroy ("WaitMsg")
ENDPROC
```