Getting and Setting the Default Printer

- JanetTerra

Table of Contents

Liberty BASIC's Printerdialog Command

The *printerdialog* function brings up the standard Windows printer dialog box. From the Liberty BASIC helpfile

- PRINTERDIALOG
- Description
- This command opens the standard Windows Common Printer Dialog. If the user chooses a printer and accepts, the next print job will go to this printer. Accepting a printer also sets the global variables PrinterName$, PrintCollate and PrintCopies to reflect what the user chose for the Printer Name, Collate and Copies. If no printer is accepted, then PrinterName$ is set to an empty string.

Unfortunately, Liberty BASIC doesn't interact well with Windows printer dialog. Despite what information PrinterName$ holds, all documents, whether text (*lprint, dump*) or graphics (*vga*, *svga*, *xga*)

will be sent to the default printer. A work-around is to set the desired printer as default before printing the document. This work-around is based upon contributions of - StPendl. All of the code used in this article has been compiled from postings made by - StPendl, both at Liberty BASIC Conforums and at the Official Liberty BASIC Support Group. The compiled snippets have been modified slightly for consistency with descriptive variable names, but otherwise remain intact.

The code in this article is only valid for **Windows 2k/XP/Vista/2k3/2k8**. This code will **not** work for **Windows 9x/ME**. For code that will work with Windows 9x/ME, see Problem with printerdialog at the Liberty BASIC Community Forum.

The first step is to identify (get) the current default printer.

Get Default Printer

The information derived from the winspool.drv DLL will be placed in a struct. The struct must first be defined, here it's *pcchBuffer*, and the length of the struct element *value* set to _MAX_PATH. The call to the DLL is then made and the name of the default printer is placed into *pcchBuffer.value.struct*. Note that *byRef* is used so that the changes to *currentDefaultPrinter$* are made both locally and globally. The function itself returns a number, 0 for a failure, non-zero for success.

```
    GetDefaultPrinter = GetDefaultPrinter(currentDefaultPrinter$)
    if GetDefaultPrinter = 0 then
        print "Call failed"
    else
        print "DefaultPrinter = ";currentDefaultPrinter$
    end if
end

function GetDefaultPrinter(byref currentDefaultPrinter$)
' Returns zero if call fails
    struct pcchBuffer, value as ulong
    currentDefaultPrinter$ = space$(_MAX_PATH)
    pcchBuffer.value.struct = _MAX_PATH

    open "winspool.drv" for dll as #winspool

    calldll #winspool, "GetDefaultPrinterA", _
        currentDefaultPrinter$ as ptr, _
        pcchBuffer as struct, _
        GetDefaultPrinter as long
```

```
    close #winspool
end function
```

Stefan's code has always been accompanied with an error catching routine. Should the call not work, Liberty BASIC can identify the cause of the failure.

```
    GetDefaultPrinter = GetDefaultPrinter(currentDefaultPrinter$)
    if GetDefaultPrinter = 0 then
        print "Call failed"
    else
        print "DefaultPrinter = ";currentDefaultPrinter$
    end if
end

function GetDefaultPrinter(byref currentDefaultPrinter$)
' Returns zero if call fails
    struct pcchBuffer, value as ulong
    currentDefaultPrinter$ = space$(_MAX_PATH)
    pcchBuffer.value.struct = _MAX_PATH

    open "winspool.drv" for dll as #winspool

    calldll #winspool, "GetDefaultPrinterA", _
        currentDefaultPrinter$ as ptr, _
        pcchBuffer as struct, _
        GetDefaultPrinter as long
    if GetDefaultPrinter = 0 then
        call DisplayError
    else
        currentDefaultPrinter$ = left$(
currentDefaultPrinter$, pcchBuffer.value.struct - 1)
    end if
    close #winspool
end function

sub DisplayError
    ErrorCode = GetLastError()

    dwFlags = _FORMAT_MESSAGE_FROM_SYSTEM
    nSize = 1024
    lpBuffer$ = space$(nSize); chr$(0)
    dwMessageID = ErrorCode

    calldll #kernel32, "FormatMessageA", _
```

```
        dwFlags      as ulong, _
        lpSource     as ulong, _
        dwMessageID  as ulong, _
        dwLanguageID as ulong, _
        lpBuffer$    as ptr, _
        nSize        as ulong, _
        Arguments    as ulong, _
        result       as ulong

    print "Error "; ErrorCode; ": "; left$(lpBuffer$, result)
end sub

function GetLastError()
    calldll #kernel32, "GetLastError", _
    GetLastError as ulong
end function
```

Storing the current default printer in a variable is important to allow that printer to be reassigned as default once the document has been printed.

List All Printers

The next step is to get a list of all available printers. This requires a little more work and three more structs. Stefan's function loops around to obtain all the printer names, until there are no names left. The printer names are concatenated, deliminated with a semicolon, in the string variable PrinterInfo$. The loop ends when error #122 (The data area passed to a system call is too small) is encounered. Once again, *byRef* is used so that PrinterInfo$ remains the same locally and globally.

Enumerating the printers requires allocating and searching blocks of memory, resulting in rather complex code. Also, listing printers available to the computer by a local (physical) connection requires different variables than listing printers available to the computer by a network or wireless connection. The EnumPrinters() function must be accessed twice, first for local printers then for network printers. The appropriate variables should be passed to the function each time.

- Flag to List Local Printers
- PRINTER.ENUM.LOCAL = hexdec("2")

- Flag to List Network Printers
- PRINTER.ENUM.CONNECTIONS = hexdec("4")

The first pass stores the retrieved information in *LocalPrinterInfo$* and the second pass stores the retrieved information in *NetworkPrinterInfo$*. These two stringes are then concatenated with a semicolon to hold all

printers in *PrinterInfo$*. Finally, an array is constructed to hold the individual printer names.

```
' Count and enumerate all printers
' Need to access function twice, first for local printers, second for
network printers
    PRINTER.ENUM.LOCAL = hexdec("2")
    PRINTER.ENUM.CONNECTIONS = hexdec("4")

    nLocalPrinters = EnumPrinters(PrinterInfo$, PRINTER.ENUM.LOCAL)
print "nLocalPrinters = ";nLocalPrinters
    LocalPrinterInfo$ = PrinterInfo$
print "LocalPrinterInfo$ = ";LocalPrinterInfo$
print
    nNetworkPrinters = EnumPrinters(
PrinterInfo$, PRINTER.ENUM.CONNECTIONS)
print "nNetworkPrinters = ";nNetworkPrinters
    NetworkPrinterInfo$ = PrinterInfo$
print "NetworkPrinterInfo$ = ";NetworkPrinterInfo$
print

' Add both to total and combine the 2 printer strings
    nPrinters = nLocalPrinters + nNetworkPrinters
Print "nPrinters = ";nPrinters
    PrinterInfo$ = LocalPrinterInfo$;";";NetworkPrinterInfo$
print "PrinterInfo$ = ";PrinterInfo$

' Place all printers in an array
    dim availablePrinters$(nPrinters)
    for i = 1 to nPrinters
        availablePrinters$(i) = word$(PrinterInfo$, i, ";")
    next i
for i = 1 to nPrinters
print i, availablePrinters$(i)
next i
end

function EnumPrinters(byref PrinterInfo$, nFlags)
' Returns the number of printers found
' Fills the submitted variable with the printer names
' Separated by semicolons (;)
    open "winspool.drv" for dll as #winspool
        struct pcbNeeded, value as ulong
        struct pcReturned, value as ulong
        struct PrinterInfo4, _
```

```
            pPrinterName$ as ptr, _
            pServerName$ as ptr, _
            Attributes as ulong
        PrinterInfo4Len = len(PrinterInfo4.struct)
        Level = 4
        cbBuf = PrinterInfo4Len
        uFlags = _LMEM_MOVEABLE or _LMEM_ZEROINIT
        calldll #kernel32, "LocalAlloc", _
            uFlags as uLong, _
            cbBuf as uLong, _
            hMem as uLong
        calldll #kernel32, "LocalLock", _
            hMem as uLong, _
            pBuffer as uLong

[retryEnumPrinters]
    calldll #winspool, "EnumPrintersA", _
        nFlags as ulong, _
        PrinterName as ulong, _
        Level as ulong, _
        pBuffer as ulong, _
        cbBuf as ulong, _
        pcbNeeded as struct, _
        pcReturned as struct, _
        result as long
        if result = 0 then
            if GetLastError() = 122 then
                cbBuf = pcbNeeded.value.struct
                hOldMem = hMem
                calldll #kernel32, "LocalReAlloc", _
                    hOldMem as ulong, _
                    cbBuf as ulong, _
                    uFlags as ulong, _
                    hMem as ulong
                calldll #kernel32, "LocalLock", _
                    hMem as uLong, _
                    pBuffer as ulong
                goto [retryEnumPrinters]
            else
                call DisplayError
            end if
        else
            EnumPrinters = pcReturned.value.struct
            BufferPointer = pBuffer
            for count = 0 to EnumPrinters - 1
                calldll #kernel32, "RtlMoveMemory", _
```

```
                    PrinterInfo4 as struct, _
                    BufferPointer as ulong, _
                    PrinterInfo4Len as ulong, _
                    result as void
                BufferPointer = BufferPointer + PrinterInfo4Len
                pointer = PrinterInfo4.pPrinterName$.struct
                PrinterInfo$ = winstring(pointer); ";"; PrinterInfo$
            next count
            PrinterInfo$ = left$(PrinterInfo$, len(PrinterInfo$)-1)
        end if
        calldll #kernel32, "LocalFree", _
            hMem as uLong, _
            result as uLong
    close #winspool
end function

function GetLastError()
    calldll #kernel32, "GetLastError", _
    GetLastError as ulong
end function
```

The final step is to designate a different printer as default.

Set Default Printer

The code to set a default printer is the simplest of all, just passing a valid printer name to the winspool.drv dll. Like the GetDefaultPrinter() function, the SetDefaultPrinter() function returns a 0 for failure, a non-zero for success.

```
    selectedDefaultPrinter$ = "My Inkjet Printer"
' Set new default printer
    SetDefaultPrinter = SetDefaultPrinter(selectedDefaultPrinter$)
end

function SetDefaultPrinter(selectedDefaultPrinter$)
' Returns zero if call fails
    open "winspool.drv" for dll as #winspool

    calldll #winspool, "SetDefaultPrinterA",_
        selectedDefaultPrinter$ as ptr,_
        SetDefaultPrinter as long
```

```
    close #winspool
end function
```

This is the same code, but with Stefan's error trapping included.

```
    selectedDefaultPrinter$ = "My Inkjet Printer"
' Set new default printer
    SetDefaultPrinter = SetDefaultPrinter(selectedDefaultPrinter$)
end

function SetDefaultPrinter(selectedDefaultPrinter$)
' Returns zero if call fails
    open "winspool.drv" for dll as #winspool

    calldll #winspool, "SetDefaultPrinterA",_
        selectedDefaultPrinter$ as ptr,_
        SetDefaultPrinter as long

    close #winspool
    if SetDefaultPrinter = 0 then call DisplayError
end function

sub DisplayError
    ErrorCode = GetLastError()

    dwFlags = _FORMAT_MESSAGE_FROM_SYSTEM
    nSize = 1024
    lpBuffer$ = space$(nSize); chr$(0)
    dwMessageID = ErrorCode

    calldll #kernel32, "FormatMessageA", _
        dwFlags      as ulong, _
        lpSource     as ulong, _
        dwMessageID  as ulong, _
        dwLanguageID as ulong, _
        lpBuffer$    as ptr, _
        nSize        as ulong, _
        Arguments    as ulong, _
        result       as ulong

    print "Error "; ErrorCode; ": "; left$(lpBuffer$, result)
end sub

function GetLastError()
```

```
    calldll #kernel32, "GetLastError", _
    GetLastError as ulong
end function
```

Getting, Listing, Setting the Default Printer (Mainwindow)

Using all three components, the programmer now has full control of getting, listing, and setting the default printer.

```
' Get the original default printer
    GetDefaultPrinter = GetDefaultPrinter(origDefaultPrinter$)
print "origDefaultPrinter$ = ";origDefaultPrinter$
print

' Count and enumerate all printers
' Need to access function twice, first for local printers, second for
network printers
    PRINTER.ENUM.LOCAL = hexdec("2")
    PRINTER.ENUM.CONNECTIONS = hexdec("4")

    nLocalPrinters = EnumPrinters(PrinterInfo$, PRINTER.ENUM.LOCAL)
print "nLocalPrinters = ";nLocalPrinters
    LocalPrinterInfo$ = PrinterInfo$
print "LocalPrinterInfo$ = ";LocalPrinterInfo$
print
    nNetworkPrinters = EnumPrinters(
PrinterInfo$, PRINTER.ENUM.CONNECTIONS)
print "nNetworkPrinters = ";nNetworkPrinters
    NetworkPrinterInfo$ = PrinterInfo$
print "NetworkPrinterInfo$ = ";NetworkPrinterInfo$
print

' Add both to total and combine the 2 printer strings
    nPrinters = nLocalPrinters + nNetworkPrinters
Print "nPrinters = ";nPrinters
    PrinterInfo$ = LocalPrinterInfo$;";";NetworkPrinterInfo$
print "PrinterInfo$ = ";PrinterInfo$

' Place all printers in an array
    dim availablePrinters$(nPrinters)
    for i = 1 to nPrinters
        availablePrinters$(i) = word$(PrinterInfo$, i, ";")
    next i
for i = 1 to nPrinters
```

```
print i, availablePrinters$(i)
next i
print

' Select another default printer
    Input "Printer to set as default > ";selectedDefaultPrinter
selectedDefaultPrinter$ = availablePrinters$(selectedDefaultPrinter)
print

' Set new default printer
    SetDefaultPrinter = SetDefaultPrinter(selectedDefaultPrinter$)
end


function EnumPrinters(byref PrinterInfo$, nFlags)
' Returns the number of printers found
' Fills the submitted variable with the printer names
' Separated by semicolons (;)
    open "winspool.drv" for dll as #winspool
        struct pcbNeeded, value as ulong
        struct pcReturned, value as ulong
        struct PrinterInfo4, _
            pPrinterName$ as ptr, _
            pServerName$ as ptr, _
            Attributes as ulong
        PrinterInfo4Len = len(PrinterInfo4.struct)
        Level = 4
        cbBuf = PrinterInfo4Len
        uFlags = _LMEM_MOVEABLE or _LMEM_ZEROINIT
        calldll #kernel32, "LocalAlloc", _
            uFlags as uLong, _
            cbBuf as uLong, _
            hMem as uLong
        calldll #kernel32, "LocalLock", _
            hMem as uLong, _
            pBuffer as uLong

[retryEnumPrinters]
    calldll #winspool, "EnumPrintersA", _
        nFlags as ulong, _
        PrinterName as ulong, _
        Level as ulong, _
        pBuffer as ulong, _
        cbBuf as ulong, _
        pcbNeeded as struct, _
        pcReturned as struct, _
```

```
        result as long
        if result = 0 then
            if GetLastError() = 122 then
                cbBuf = pcbNeeded.value.struct
                hOldMem = hMem
                calldll #kernel32, "LocalReAlloc", _
                    hOldMem as ulong, _
                    cbBuf as ulong, _
                    uFlags as ulong, _
                    hMem as ulong
                calldll #kernel32, "LocalLock", _
                    hMem as uLong, _
                    pBuffer as ulong
                goto [retryEnumPrinters]
            else
                call DisplayError
            end if
        else
            EnumPrinters = pcReturned.value.struct
            BufferPointer = pBuffer
            for count = 0 to EnumPrinters - 1
                calldll #kernel32, "RtlMoveMemory", _
                    PrinterInfo4 as struct, _
                    BufferPointer as ulong, _
                    PrinterInfo4Len as ulong, _
                    result as void
                BufferPointer = BufferPointer + PrinterInfo4Len
                pointer = PrinterInfo4.pPrinterName$.struct
                PrinterInfo$ = winstring(pointer); ";"; PrinterInfo$
            next count
            PrinterInfo$ = left$(PrinterInfo$, len(PrinterInfo$)-1)
        end if
        calldll #kernel32, "LocalFree", _
            hMem as uLong, _
            result as uLong
    close #winspool
end function


function GetDefaultPrinter(byref currentDefaultPrinter$)
' Returns zero if call fails

    struct pcchBuffer, value as ulong
    currentDefaultPrinter$ = space$(_MAX_PATH)
    pcchBuffer.value.struct = _MAX_PATH
```

```
    open "winspool.drv" for dll as #winspool

    calldll #winspool, "GetDefaultPrinterA", _
        currentDefaultPrinter$ as ptr, _
        pcchBuffer as struct, _
        GetDefaultPrinter as long

    close #winspool

    if GetDefaultPrinter = 0 then
        call DisplayError
    else
        currentDefaultPrinter$ = left$(
currentDefaultPrinter$, pcchBuffer.value.struct - 1)
    end if
end function

function SetDefaultPrinter(selectedDefaultPrinter$)
' Returns zero if call fails
    open "winspool.drv" for dll as #winspool

    calldll #winspool, "SetDefaultPrinterA",_
        selectedDefaultPrinter$ as ptr,_
        SetDefaultPrinter as long

    close #winspool
    if SetDefaultPrinter = 0 then call DisplayError
end function

sub DisplayError
    ErrorCode = GetLastError()

    dwFlags = _FORMAT_MESSAGE_FROM_SYSTEM
    nSize = 1024
    lpBuffer$ = space$(nSize); chr$(0)
    dwMessageID = ErrorCode

    calldll #kernel32, "FormatMessageA", _
        dwFlags      as ulong, _
        lpSource     as ulong, _
        dwMessageID  as ulong, _
        dwLanguageID as ulong, _
        lpBuffer$    as ptr, _
        nSize        as ulong, _
        Arguments    as ulong, _
        result       as ulong
```

```
    print "Error "; ErrorCode; ": "; left$(lpBuffer$, result)
end sub

function GetLastError()
    calldll #kernel32, "GetLastError", _
    GetLastError as ulong
end function
```

Getting, Listing, Setting the Default Printer (GUI)

A demo using a dialog window to select the user's choice of printer. The program returns the original default printer as default after each print.

```
' Get the original default printer
    GetDefaultPrinter = GetDefaultPrinter(origDefaultPrinter$)
    origDefaultPrinter$ = origDefaultPrinter$
    defaultPrinter$ = origDefaultPrinter$

' Count and enumerate all printers
' Need to access function twice, first for local printers, second for
network printers
    PRINTER.ENUM.LOCAL = hexdec("2")
    PRINTER.ENUM.CONNECTIONS = hexdec("4")

    nLocalPrinters = EnumPrinters(PrinterInfo$, PRINTER.ENUM.LOCAL)
    LocalPrinterInfo$ = PrinterInfo$
    nNetworkPrinters = EnumPrinters(
PrinterInfo$, PRINTER.ENUM.CONNECTIONS)
    NetworkPrinterInfo$ = PrinterInfo$

' Add both to total and combine the 2 printer strings
    nPrinters = nLocalPrinters + nNetworkPrinters
    PrinterInfo$ = LocalPrinterInfo$;";";NetworkPrinterInfo$

' Place all printers in an array
    dim availablePrinters$(nPrinters)
    for i = 1 to nPrinters
        availablePrinters$(i) = word$(PrinterInfo$, i, ";")
    next i

    WindowWidth = 809
    WindowHeight = 600
    UpperLeftX = Int((DisplayWidth - WindowWidth) / 2)
```

```
    UpperLeftY = Int((DisplayHeight - WindowHeight) / 2)
    menu #main, "&File", "&Print",[printerSelection], |,"E&xit", [
quit]
    Graphicbox #main.g, 1, 0, 800, 550
    open "Printer Selection" for Window as #main
    #main "trapclose [quit]"
    call screenDisplay

wait

[quit]
    close #main
end

[printerSelection]
    WindowWidth = 250
    WindowHeight = 150
    UpperLeftX = 8
    UpperLeftY = 8
    listbox #dlg.sel, availablePrinters$(), [printScreen], 14, 20,
214, 54
    button #dlg.prnt, "Print", [printScreen], UL, 14, 84, 70, 28
    button #dlg.cncl, "Cancel", [closeDlg], UL, 160, 84, 70, 28
    stylebits #dlg, _WS_POPUP or _WS_THICKFRAME, _WS_CAPTION, 0, 0
    open "Select Printer" for dialog_modal as #dlg
    #dlg "trapclose [closeDlg]"
    #dlg.sel "select ";defaultPrinter$
wait

[printScreen]
    #dlg.sel "selection? selPrinter$"
    defaultPrinter$ = selPrinter$
' Set the default printer as the selected printer
    SetDefaultPrinter = SetDefaultPrinter(selPrinter$)
    #main.g "print svga"
' Return the default printer to the original default printer
    SetDefaultPrinter = SetDefaultPrinter(origDefaultPrinter$)

[closeDlg]
    close #dlg
wait

sub screenDisplay
    #main.g "down; font verdana 14 bold; place 300 200"
    #main.g, "\Hello World"
    #main.g, "flush"
```

```
end sub

function EnumPrinters(byref PrinterInfo$, nFlags)
' Returns the number of printers found
' Fills the submitted variable with the printer names
' Separated by semicolons (;)
    open "winspool.drv" for dll as #winspool
        struct pcbNeeded, value as ulong
        struct pcReturned, value as ulong
        struct PrinterInfo4, _
            pPrinterName$ as ptr, _
            pServerName$ as ptr, _
            Attributes as ulong
        PrinterInfo4Len = len(PrinterInfo4.struct)
        Level = 4
        cbBuf = PrinterInfo4Len
        uFlags = _LMEM_MOVEABLE or _LMEM_ZEROINIT
        calldll #kernel32, "LocalAlloc", _
            uFlags as uLong, _
            cbBuf as uLong, _
            hMem as uLong
        calldll #kernel32, "LocalLock", _
            hMem as uLong, _
            pBuffer as uLong

[retryEnumPrinters]
    calldll #winspool, "EnumPrintersA", _
        nFlags as ulong, _
        PrinterName as ulong, _
        Level as ulong, _
        pBuffer as ulong, _
        cbBuf as ulong, _
        pcbNeeded as struct, _
        pcReturned as struct, _
        result as long
        if result = 0 then
            if GetLastError() = 122 then
                cbBuf = pcbNeeded.value.struct
                hOldMem = hMem
                calldll #kernel32, "LocalReAlloc", _
                    hOldMem as ulong, _
                    cbBuf as ulong, _
                    uFlags as ulong, _
                    hMem as ulong
                calldll #kernel32, "LocalLock", _
                    hMem as uLong, _
```

```
                    pBuffer as ulong
                goto [retryEnumPrinters]
            else
                call DisplayError
            end if
        else
            EnumPrinters = pcReturned.value.struct
            BufferPointer = pBuffer
            for count = 0 to EnumPrinters - 1
                calldll #kernel32, "RtlMoveMemory", _
                    PrinterInfo4 as struct, _
                    BufferPointer as ulong, _
                    PrinterInfo4Len as ulong, _
                    result as void
                BufferPointer = BufferPointer + PrinterInfo4Len
                pointer = PrinterInfo4.pPrinterName$.struct
                PrinterInfo$ = winstring(pointer); ";"; PrinterInfo$
            next count
            PrinterInfo$ = left$(PrinterInfo$, len(PrinterInfo$)-1)
        end if
        calldll #kernel32, "LocalFree", _
            hMem as uLong, _
            result as uLong
    close #winspool
end function


function GetDefaultPrinter(byref currentDefaultPrinter$)
' Returns zero if call fails

    struct pcchBuffer, value as ulong
    currentDefaultPrinter$ = space$(_MAX_PATH)
    pcchBuffer.value.struct = _MAX_PATH

    open "winspool.drv" for dll as #winspool

    calldll #winspool, "GetDefaultPrinterA", _
        currentDefaultPrinter$ as ptr, _
        pcchBuffer as struct, _
        GetDefaultPrinter as long

    close #winspool

    if GetDefaultPrinter = 0 then
        call DisplayError
    else
```

```
        currentDefaultPrinter$ = left$(
currentDefaultPrinter$, pcchBuffer.value.struct - 1)
    end if
end function

function SetDefaultPrinter(selectedDefaultPrinter$)
' Returns zero if call fails
    open "winspool.drv" for dll as #winspool

    calldll #winspool, "SetDefaultPrinterA",_
        selectedDefaultPrinter$ as ptr,_
        SetDefaultPrinter as long

    close #winspool
    if SetDefaultPrinter = 0 then call DisplayError
end function

sub DisplayError
    ErrorCode = GetLastError()

    dwFlags = _FORMAT_MESSAGE_FROM_SYSTEM
    nSize = 1024
    lpBuffer$ = space$(nSize); chr$(0)
    dwMessageID = ErrorCode

    calldll #kernel32, "FormatMessageA", _
        dwFlags      as ulong, _
        lpSource     as ulong, _
        dwMessageID  as ulong, _
        dwLanguageID as ulong, _
        lpBuffer$    as ptr, _
        nSize        as ulong, _
        Arguments    as ulong, _
        result       as ulong

    print "Error "; ErrorCode; ": "; left$(lpBuffer$, result)
end sub

function GetLastError()
    calldll #kernel32, "GetLastError", _
    GetLastError as ulong
end function
```

Printer Dialog Clone

- robmcal offers code for a look alike printer dialog box.