

Re: vb6 run from and output to command window

Source: <http://www.tech-archive.net/Archive/VB/microsoft.public.vb.general.discussion/2005-01/0444.html>

From: Björn Holmgren (*bjohol_at_hotmail.com*)

Date: 01/05/05

Date: Wed, 5 Jan 2005 08:33:21 +0100

"Andrew3000" <Andrew3000@discussions.microsoft.com> wrote in message news:D23DBD95-6B14-4560-B0B6-5D79BBA7E860@microsoft.com...

> *I've got a small vb6 app which I run from via a command / DOS window.*

>

> *I would like to output some text (the result of running the application)*

to

> *the same command window that runs the application, effectively mimicing the*

> *LPRINT function of the old DOS versions of VB.*

>

> *Can anyone advise pls?*

VB doesn't really create true console applications (because the linker doesn't support it). As always (well, almost) there are ways of overcoming this limitation.

The first thing you need to do is to implement some API functions to allocate the console window, and read/write from/to it. There are three standard devices you can use – Input, Output and Error – and 5 related API functions: AllocConsole to allocate the console window, FreeConsole to de-allocate it, GetStdHandle to retrieve a handle to any of the three standard devices, and finally ReadConsole and WriteConsole to read and write from/to the devices (obviously Input supports only ReadConsole and Output and Error only WriteConsole).

After you have completed and compiled your console application, you will also need to "convert" the executable module to a true console application. There are different ways of doing this. You could use a third-party linker – I don't remember the names of any of them right now (maybe someone else has a link) – or (as I will demonstrate here), use the EDITBIN utility shipped with Visual C++ (included in Visual Studio).

Now let's create a small demonstration application. First, create a new VB project and add a standard module. Paste the following code in the module's code window: (this contains all the API declarations you need as well as helper functions for allocating, freeing, reading from and writing to the console.

```
'-----  
Private Declare Function AllocConsole Lib "kernel32" () As Long  
Private Declare Function FreeConsole Lib "kernel32" () As Long  
Private Declare Function WriteConsole Lib "kernel32" Alias _  
    "WriteConsoleA" (ByVal hConsoleOutput As Long, lpBuffer As Any, _  
    ByVal nNumberOfCharsToWrite As Long, _  
    lpNumberOfCharsWritten As Long, lpReserved As Any) As Long  
Private Declare Function ReadConsole Lib "kernel32" Alias _  
    "ReadConsoleA" (ByVal hConsoleInput As Long, lpBuffer As Any, _  
    ByVal nNumberOfCharsToRead As Long, lpNumberOfCharsRead As Long, _  
    lpReserved As Any) As Long  
Private Declare Function GetStdHandle Lib "kernel32" (_  
    ByVal nStdHandle As Long) As Long  
  
Private Const STD_ERROR_HANDLE = -12&  
Private Const STD_INPUT_HANDLE = -10&  
Private Const STD_OUTPUT_HANDLE = -11&  
  
Private hOutput As Long  
Private hInput As Long  
Private hError As Long  
  
Function ConsoleAlloc() As Boolean  
    AllocConsole  
  
    hOutput = GetStdHandle(STD_OUTPUT_HANDLE)  
    hInput = GetStdHandle(STD_INPUT_HANDLE)  
    hError = GetStdHandle(STD_ERROR_HANDLE)  
  
    ConsoleAlloc = ((hOutput <> 0) And (hInput <> 0) And (hError <> 0))  
End Function  
  
Function ConsoleFree()  
    Dim lResult As Long  
  
    lResult = FreeConsole()  
  
    ConsoleFree = (lResult <> 0)  
End Function  
  
Function ConsoleWrite(sOutput As String) As Boolean  
    Dim lOutput As Long  
    Dim lWritten As Long  
    Dim lResult As Long  
  
    lOutput = Len(sOutput)  
    lResult = WriteConsole(hOutput, ByVal sOutput, lOutput, lWritten, 0)  
  
    ConsoleWrite = (lResult <> 0)  
End Function
```

```
Function ConsoleWriteLn(sOutput As String) As Boolean
    ConsoleWriteLn = ConsoleWrite(sOutput & vbCrLf)
End Function

Function ConsoleError(sOutput As String) As Boolean
    Dim lOutput As Long
    Dim lWritten As Long
    Dim lResult As Long

    lOutput = Len(sOutput)
    lResult = WriteConsole(hError, ByVal sOutput, lOutput, lWritten, 0)

    ConsoleError = (lResult <> 0)
End Function

Function ConsoleErrorLn(sOutput As String) As Boolean
    ConsoleErrorLn = ConsoleError(sOutput & vbCrLf)
End Function

Function ConsoleRead(sInput As String) As Boolean
    Dim lInput As Long
    Dim lResult As Long
    Dim lRead As Long

    sInput = Space(255)
    lInput = Len(sInput)

    lResult = ReadConsole(hInput, ByVal sInput, lInput, lRead, 0)

    If lRead > 1 Then sInput = Left(sInput, lRead - 2)

    ConsoleRead = (lResult <> 0)
End Function
```

Second, create a new module and paste the following code in the code window:

```
-----  
Sub Main()
    Dim sInput As String

    ConsoleAlloc
    ConsoleWriteLn "Welcome to this demonstration of a console application  
in VB6"
    ConsoleWriteLn  
"-----"  
    ConsoleWriteLn ""
    ConsoleWrite "Please write someting: "  
    ConsoleRead sInput
    ConsoleWriteLn ""
    ConsoleWriteLn "This is what you wrote;"
```

```
ConsoleWriteLn Chr(34) & sInput & Chr(34)
ConsoleWriteLn ""
ConsoleWrite "Press Enter to continue..."
ConsoleRead sInput
ConsoleFree
End Sub
```

'-----

Now, go to project properties and set "Sub Main" as the startup object. Run the program and verify that a command window pops up and that you can enter some text at the prompt (and that that text is echoed back).

The next step is to compile (make) your executable file. Now try opening a command prompt and running your program standalone. See what happens? The program will always allocate a new console window instead of reusing the console from which the program was started. This is because the program is not linked as a true console application.

The following procedure will "convert" your program into a true console application. It assumes you also have Visual C++ installed. If you don't, then you need to find a third party linker to do the job for you.

Locate the following files in your Visual Studio installation folder and copy them to the project folder:
EDITBIN.EXE (VC98\BIN)
LINK.EXE (VC98\BIN)
MSPDB60.DLL (VB98)

Open a command prompt, cd to the project folder and type the following command:
EDITBIN /SUBSYSTEM:CONSOLE <NameOfYourExeFile>

Now the application will create a new console window only when not started from an existing console.

--
Björn Holmgren