

Visual Basic 6 Support Information

Vaunix Technology Corporation

May 6, 2021

These notes apply to calling the Lab Brick LDA attenuator DLL from Microsoft's VB6. VB6 is a legacy Microsoft product.

1) VB6 can only call out to external dlls which use the older Win32 API calling conventions (what is normally declared as `__stdcall` in C). Newer DLLs, like the Lab Brick DLL use a newer calling convention which is easier to work with in the more recent programming tools. To solve this issue a special version of the LDA attenuator DLL has been built that uses the older calling convention so that the user can call it from VB6. A copy of the DLL, and its related .lib file (which probably isn't needed but is included as a reference) are attached.

Guidance on how to use the DLL functions from an application can be found in the LDA SDK documentation, and by reviewing the C language example programs in the SDK.

2) VB6 requires that the user declare external functions before they are called. While this topic is covered in some detail in the VB6 documentation, it would be helpful to have an example of what the declarations should look like for your program. Therefore, a set of declarations for the functions in the LDA attenuator DLL has been created and is included in an attached file.

3) The general approach would be to include the declarations in the application source file, and then call the DLL to perform the desired functions. Use the C code example provided with the original DLL as a guide for what needs to be done. Basically, the user program needs to allocate an array of Longs to hold the device IDs which the DLL returns to the program. Then the program picks the device ID for the attenuator to be controlled, opens it, and issues commands. Remember to close it when finished.

4) Neither the special version of the DLL nor the declarations have been tested under VB6. The DLL has been tested from a C program, and the format of its exported functions has been carefully inspected. The declarations can only really be tested under VB6. The simple functions that pass Longs should be fine, since the declarations are straightforward. The declaration, and the argument passing for the `fnLDA_GetDevInfo` function are tricky. Since C programs refer to arrays by the address of the first element of the array, the application call the DLL needs to pass the address of first element of the array you create in your VB6 program. Note that in this case you don't want to pass the value of the array element, but its address – hence the argument does not have the `By Value` keyword.

You should check that this is correct – it seems so from the documentation, but it is something the user should double check. Similarly, the `fnLDA_GetModelName` function returns a string, and that can be tricky in VB6 – so if you need to use that function you will have to make sure the argument passing works correctly.

All of these cases are described in the Microsoft documentation, searching the MSDN documentation should let you find whatever you need. Finally, note that the C `int` variable is actually a Long in VB6 – a VB6 `int` is a 16 bit integer, while `ints` in the C compiler used to create

the DLL are 32 bits. Some of the functions take a Boolean value as an argument. The C compiler represents these as 32 bit quantities so they are also declared in the VB6 function declarations as Longs. The value of zero is false, a non-zero value (such as 1) is true.

The values returned by the functions will have their high bit set if an error occurred, and therefore will appear as negative numbers in VB6.

Vaunix Technology Corporation