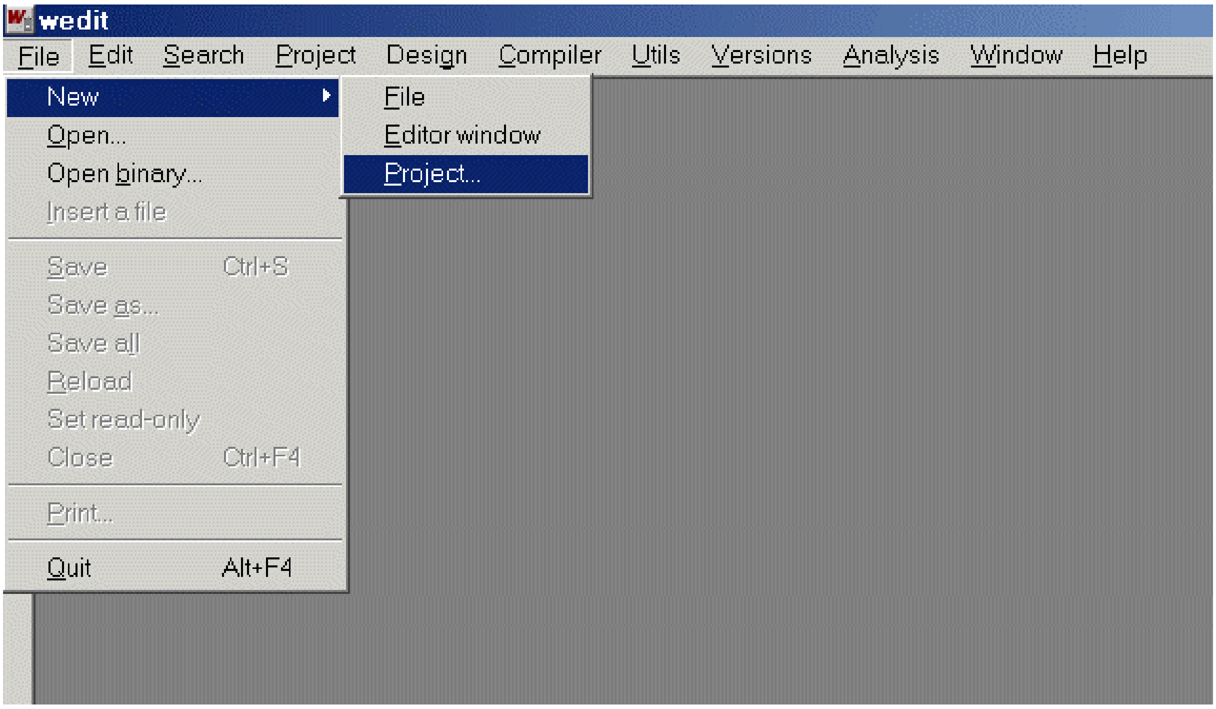
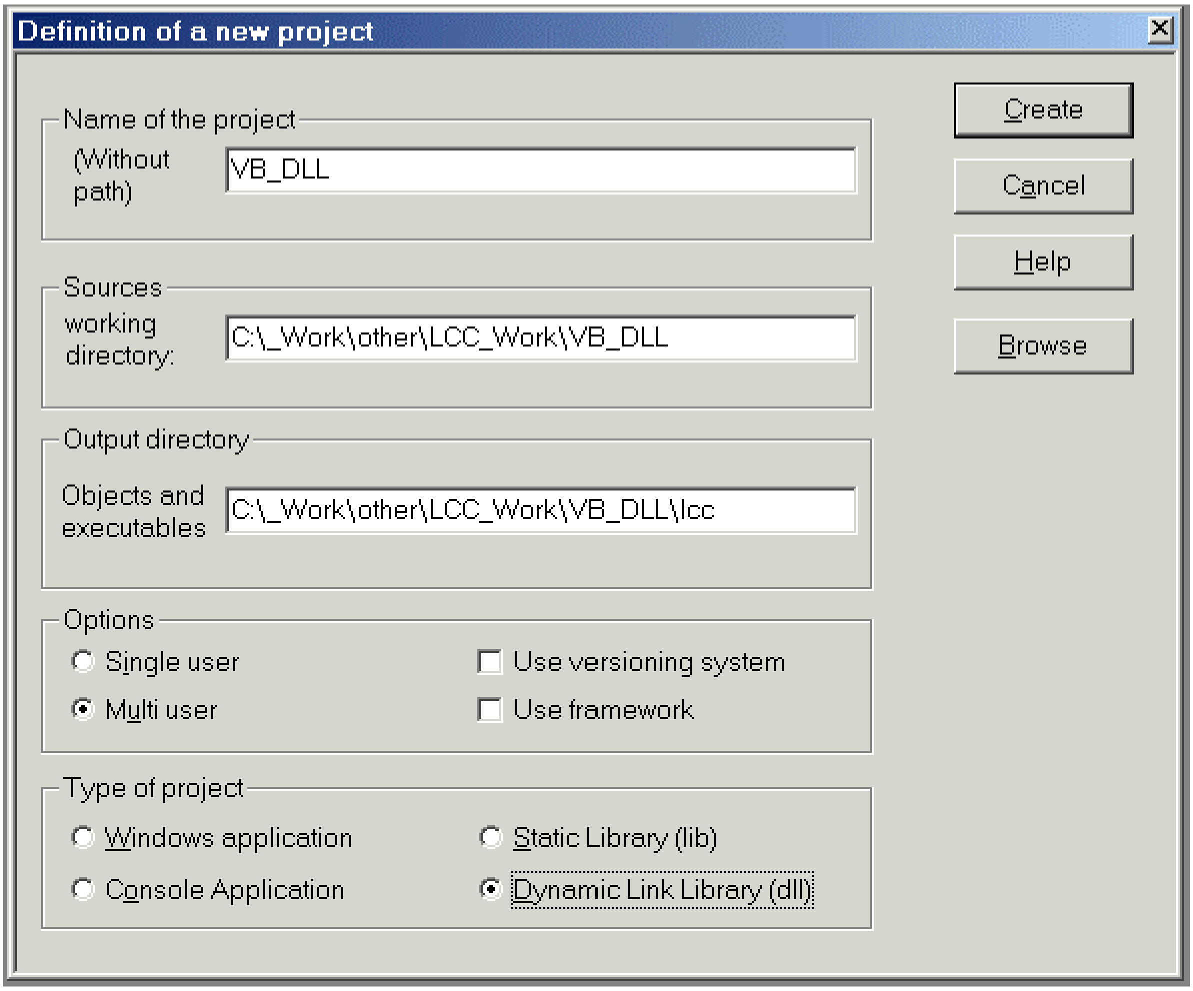
Creating a DLL for Excel or Visual Basic with lcc-win32

Create a directory of your choice (I had troubles on my old Windows if I do it through wedit).

Open wedit and select to create a new project (may be you are asked to give a login name):



Name your project and select the path through the "browse" button (it will insert your selection).



Name of the project

(Without path) VB_DLL

Sources

working directory: C:_Work\other\LCC_Work\VB_DLL

Output directory

Objects and executables C:_Work\other\LCC_Work\VB_DLL\lcc

Options

○ Single user

● Multi user

□ Use versioning system

□ Use framework

Type of project

○ Windows application

○ Console Application

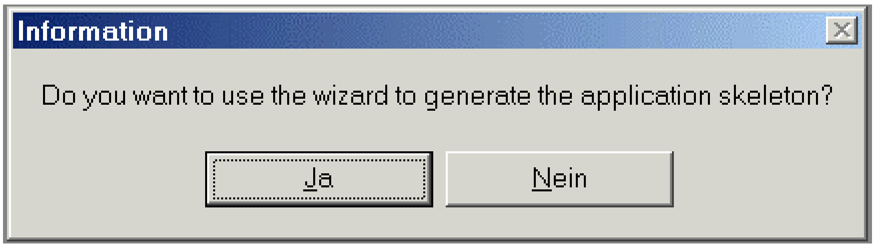
○ Static Library (lib)

● Dynamic Link Library (dll)

Choose the option "Multi user" and "Dynamic Link Library" as project type.

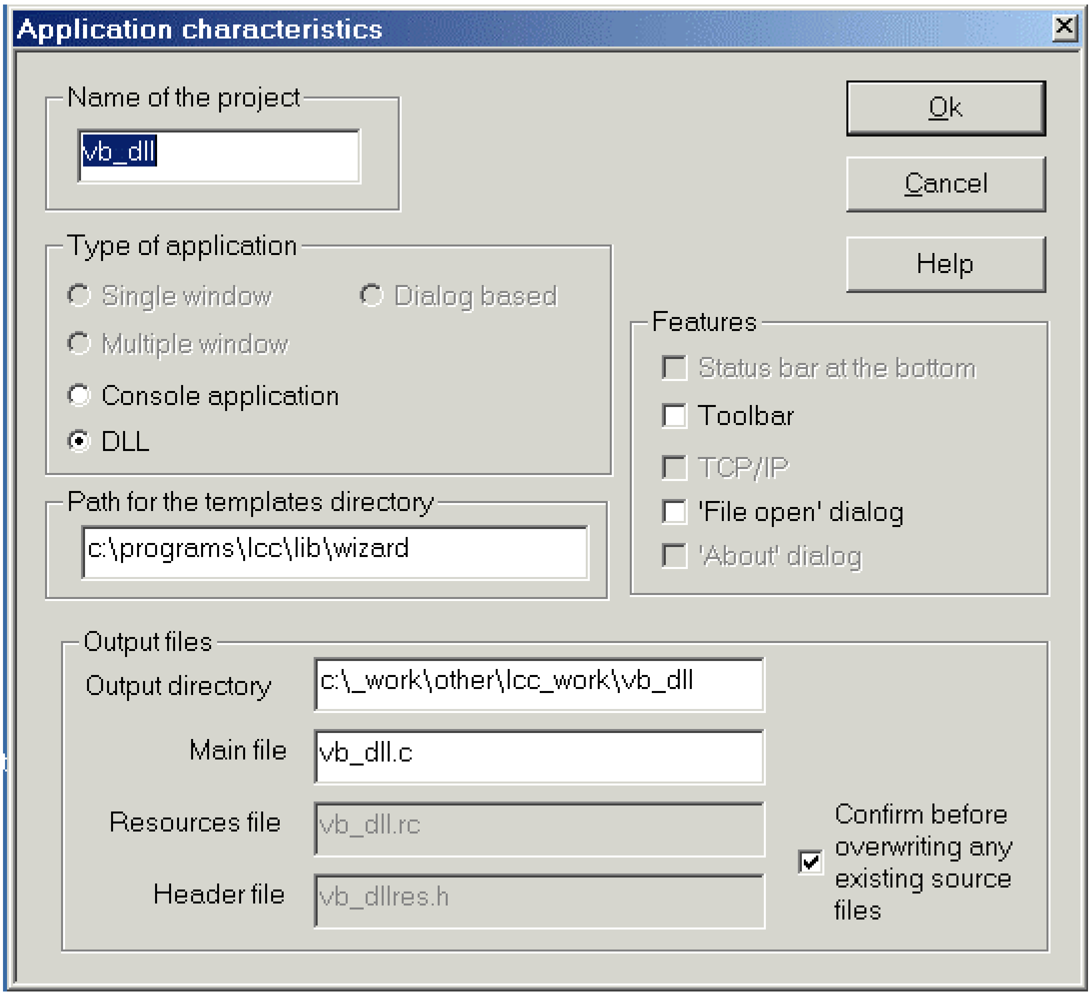
Hit the create button.

You will be asked whether you want to use a wizard to help you. Say "yes". After that confirm again:



Information

Do you want to use the wizard to generate the application skeleton?



Application characteristics

Name of the project: vb_dll

Type of application
- ○ Single window
- ○ Dialog based
- ○ Multiple window
- ○ Console application
- ● DLL

Path for the templates directory: c:\programs\lcc\lib\wizard

Features
- □ Status bar at the bottom
- □ Toolbar
- □ TCP/IP
- □ 'File open' dialog
- □ 'About' dialog

Output files

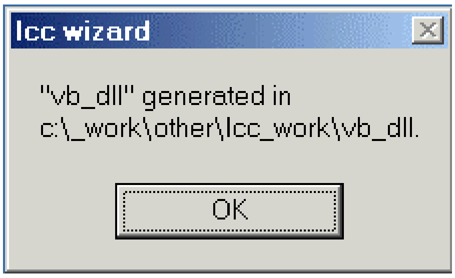
Output directory: c:_work\other\lcc_work\vb_dll

Main file: vb_dll.c

Resources file: vb_dll.rc

Header file: vb_dllres.h

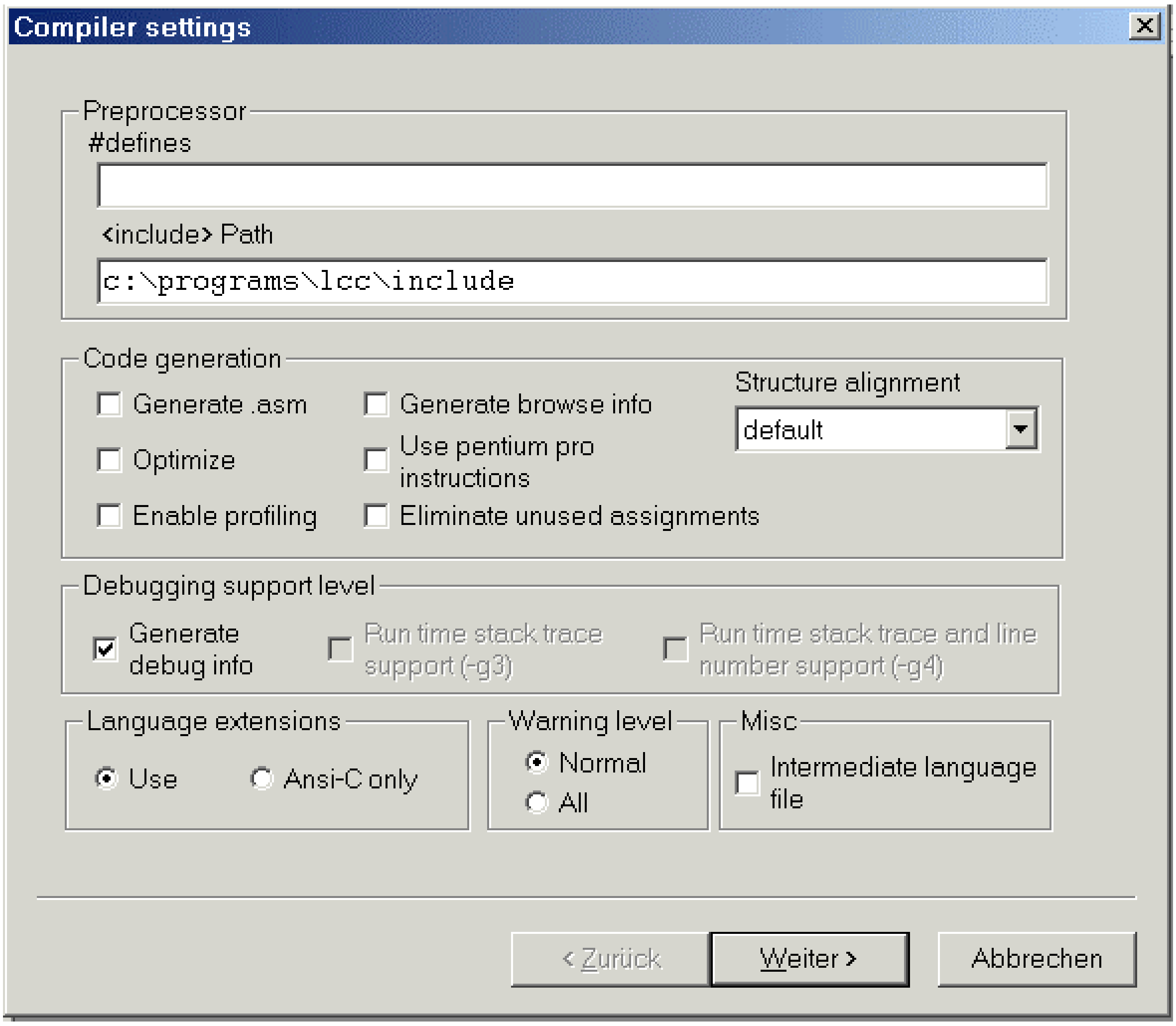
☑ Confirm before overwriting any existing source files



lcc wizard

"vb_dll" generated in c:_work\other\lcc_work\vb_dll.

After confirming you want to agree with the compiler settings:



Compiler settings

Preprocessor

#defines

<include> Path

c:\programs\lcc\include

Code generation

□ Generate .asm
□ Generate browse info
□ Optimize
□ Use pentium pro instructions
□ Enable profiling
□ Eliminate unused assignments

Structure alignment: default

Debugging support level

☑ Generate debug info
□ Run time stack trace support (-g3)
□ Run time stack trace and line number support (-g4)

Language extensions

● Use ○ Ansi-C only

Warning level

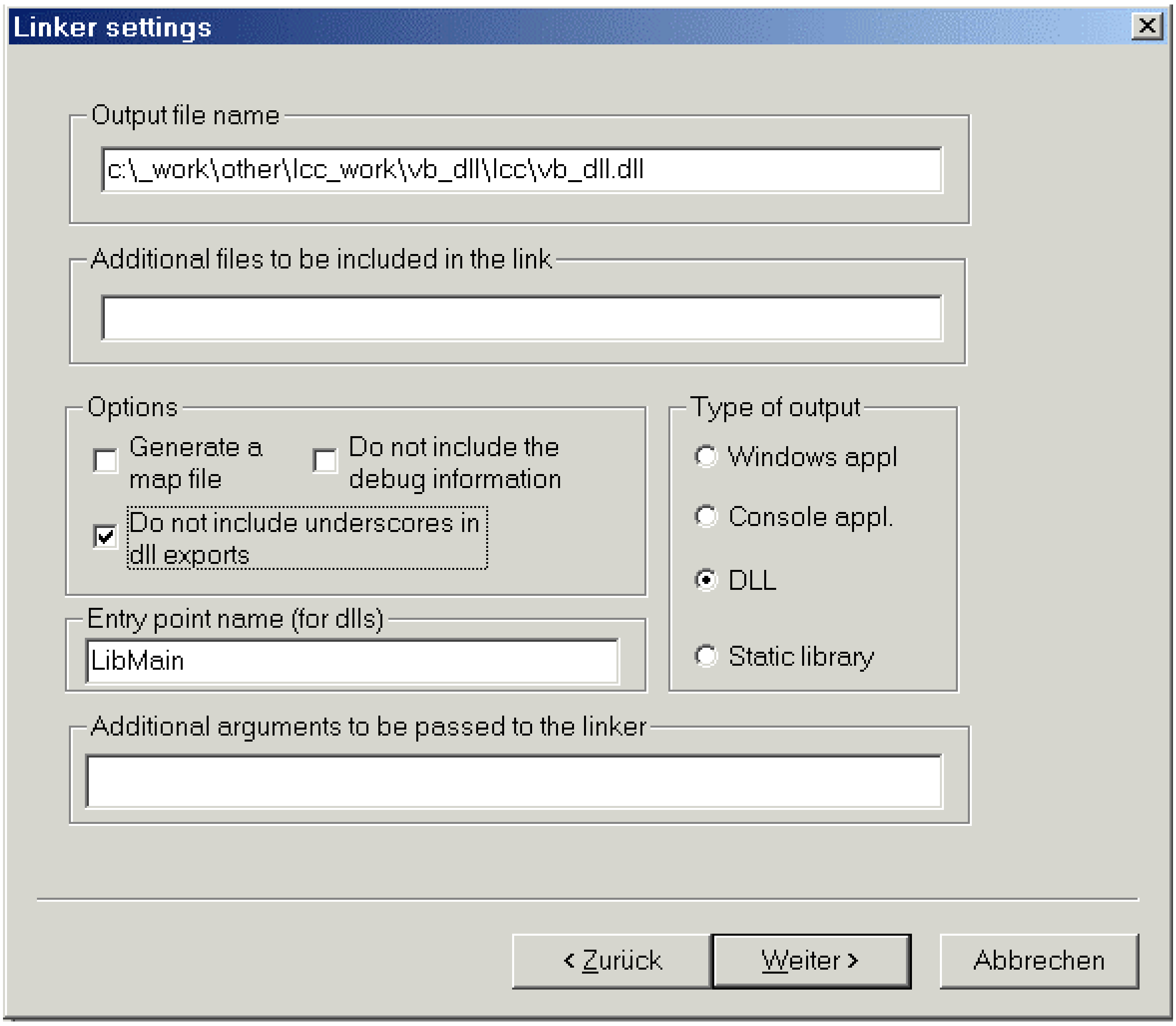
● Normal
○ All

Misc

□ Intermediate language file

while in the Linker settings activate the check box "**Do not include underscores in dll exports**".

If you would have "underscores" then the function(s) to be called from the DLL will have different names and even if that is systematic it is tedious to find what to state in the declarations for Excel.



Linker settings

Output file name

c:_work\other\lcc_work\vb_dll\lcc\vb_dll.dll

Additional files to be included in the link

Options

☐ Generate a map file

☐ Do not include the debug information

☑ Do not include underscores in dll exports

Entry point name (for dlls)

LibMain

Type of output

○ Windows appl

○ Console appl.

● DLL

○ Static library

Additional arguments to be passed to the linker

The entry point for Excel is named LibMain, no need to change it now.

The Debugger settings can be accepted as well - then wedit opens with the file vb_dll.c

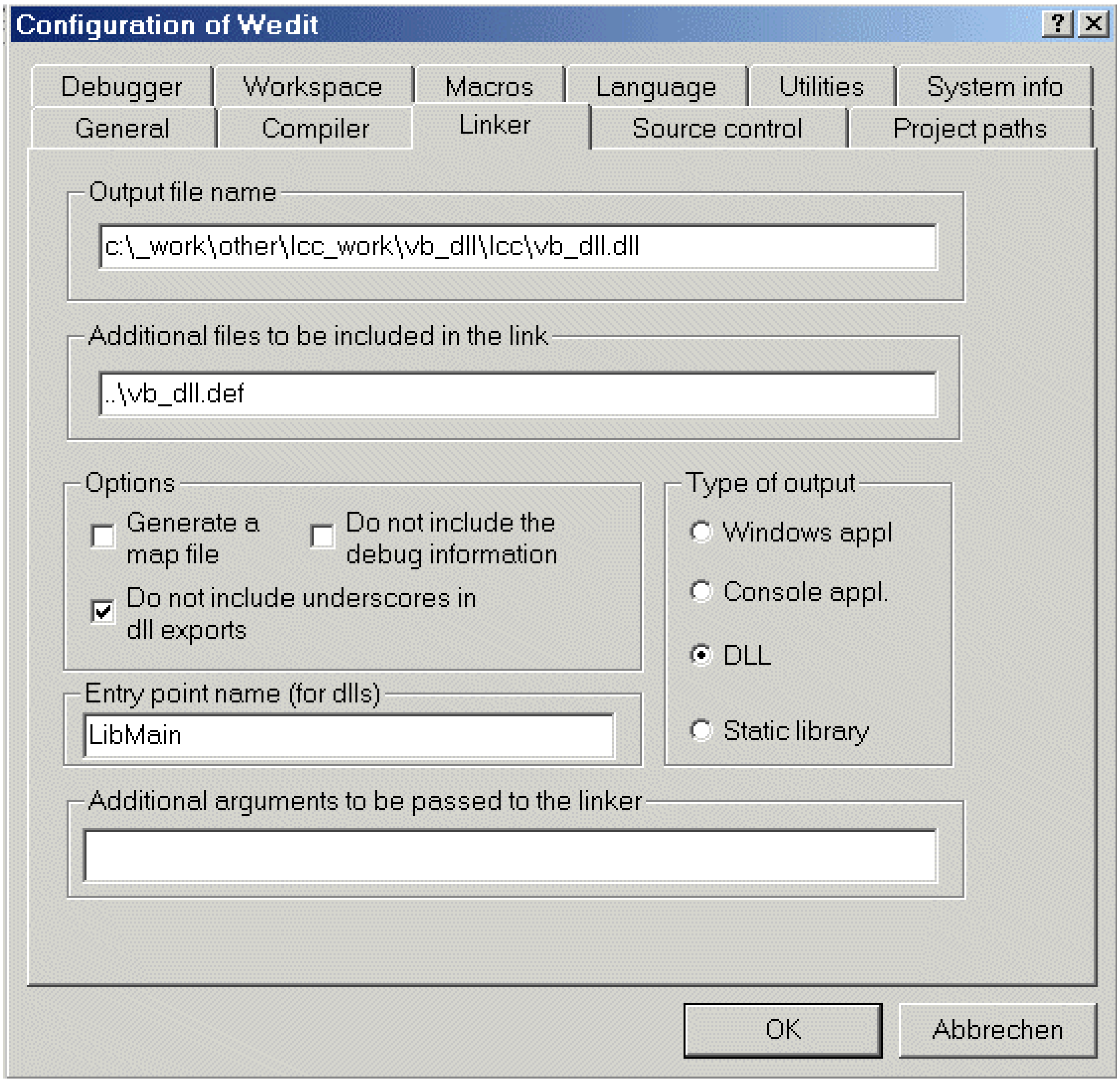
After that use a text editor to create a file vb_dll.txt and insert the following:

```
LIBRARY vb_dll
EXPORTS
      add
```

Save the file in your project directory (at the same level where you have the C source vb_dll.c).

Then rename the text file and call it vb_dll.**def** - even if Windows complains: take that extension.

Through the menu open the project configuration and change the settings for the Linker by adding the export file as "**..**\vb_dll.def" - observe the two dots and the back slash.



Output file name: c:_work\other\lcc_work\vb_dll\lcc\vb_dll.dll

Additional files to be included in the link: ..\vb_dll.def

Options

□ Generate a map file

□ Do not include the debug information

☑ Do not include underscores in dll exports

Type of output

○ Windows appl

○ Console appl.

● DLL

○ Static library

Entry point name (for dlls): LibMain

Additional arguments to be passed to the linker:

You will be asked whether the makefile should be rebuilt, agree that.

Likewise you can add the def-file through the menu: Project / Add/Delete files and then use the button "Add new file …" to select the def-file (showing all files).

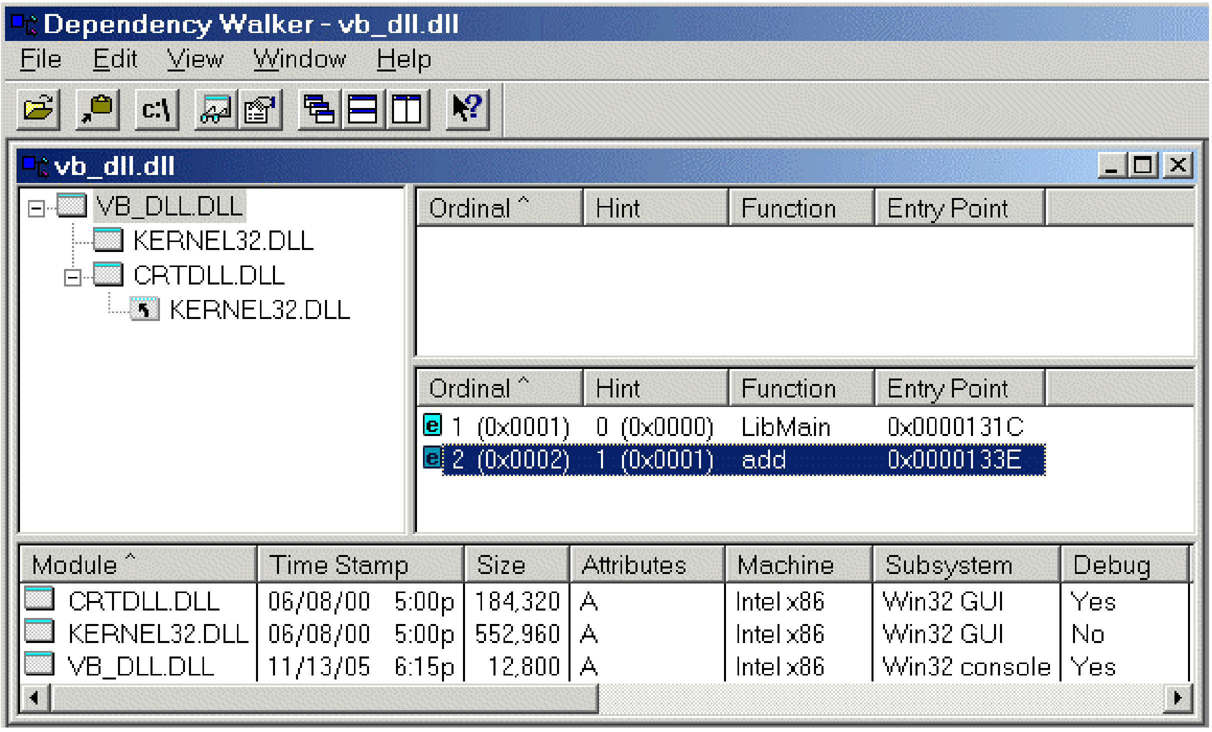
Do **not** use both ways to add a def-file to the project!

Then at the end of the C source insert your function (which is exported through the def file):

```
extern __declspec(dllexport) double __stdcall
add (double a, double b)
{
      return a + b;
}
```

Choose Compiler / Rebuild all and after saving you will find your DLL in the local subdirectory "lcc".

If you have MSVC you can see the functions in the DLL



Ordinal ^	Hint	Function	Entry Point

Ordinal ^	Hint	Function	Entry Point
1 (0x0001)	0 (0x0000)	LibMain	0x0000131C
2 (0x0002)	1 (0x0001)	add	0x0000133E

Module ^	Time Stamp	Size	Attributes	Machine	Subsystem	Debug
CRTDLL.DLL	06/08/00 5:00p	184,320	A	Intel x86	Win32 GUI	Yes
KERNEL32.DLL	06/08/00 5:00p	552,960	A	Intel x86	Win32 GUI	No
VB_DLL.DLL	11/13/05 6:15p	12,800	A	Intel x86	Win32 console	Yes

But this is not need … it is just to see, that there are no "underscores" (there wont be any).

All you need now is to create an Excel file and use the following code in its project:

```
Option Explicit

Declare Function add_DLL _
  Lib "C:\_Work\other\LCC_Work\VB_DLL\lcc\vb_dll.dll" _
  Alias "add" ( _
  ByVal a As Double, ByVal b As Double) As Double

Sub tst()
Dim a As Double, b As Double

' take some randomized test values
a = Rnd()
b = Rnd()

Debug.Print add_DLL(a, b); a + b
End Sub
```

Done.

Note that this DLL also works for Maple:

```
restart; kernelopts(version);
```

```
Maple 10.01, IBM INTEL NT, Aug 18 2005 Build ID 196612
```

```
Digits:=18;
```

```
Digits := 18
```

```
# the DLL used below; assumed to be in the folder of this sheet
currentdir(): theDLL:=cat(%,`\\vb_dll.dll`);
```

```
theDLL := "C:\_Work\other\LCC_Work\VB_DLL\lcc\vb_dll.dll"
```

```
fct := define_external(
  'add',
  'C',
  'x'::float[8],
  'y'::float[8],
  'RETURN'::float[8] ,
  LIB=theDLL);
```

```
fct := proc (x::numeric, y::numeric)
option  call_external, define_external( add, C, x::float[ 8 ], y::float[ 8 ],
RETURN::float[ 8 ], LIB = "C:\_Work\other\LCC_Work\VB_DLL\lcc\vb_dll.dll");
    call_external( Array(1..11, [...], datatype = integer[4], readonly) args)
end proc
```

```
x:= RandomTools[Generate](float(range=1..2));
y:= RandomTools[Generate](float(range=1..2));
``;
fct(x,y);
x+y;
```

```
x := 1.41019876016299766

y := 1.36650214228473823


2.77670090244773605

2.77670090244773589
```

AVt, Nov 2005