Tutorial : creating a Max/MSP external project for Windows using Visual Studio

Version 1.0 (17th July 2011) by Benoit Bouchez
Reviewed on 5th November 2013 for Max 6 SDK before publishing on Cycling'74 forum.

Introduction

This document is a short tutorial explaining how to create "from scratch" a Visual Studio project for a Max/MSP external for the Windows platform. It is not meant to explain how Max objects are working, there are enough good examples for that in Max SDK.

I wrote a similar tutorial for writing Max externals on Mac.

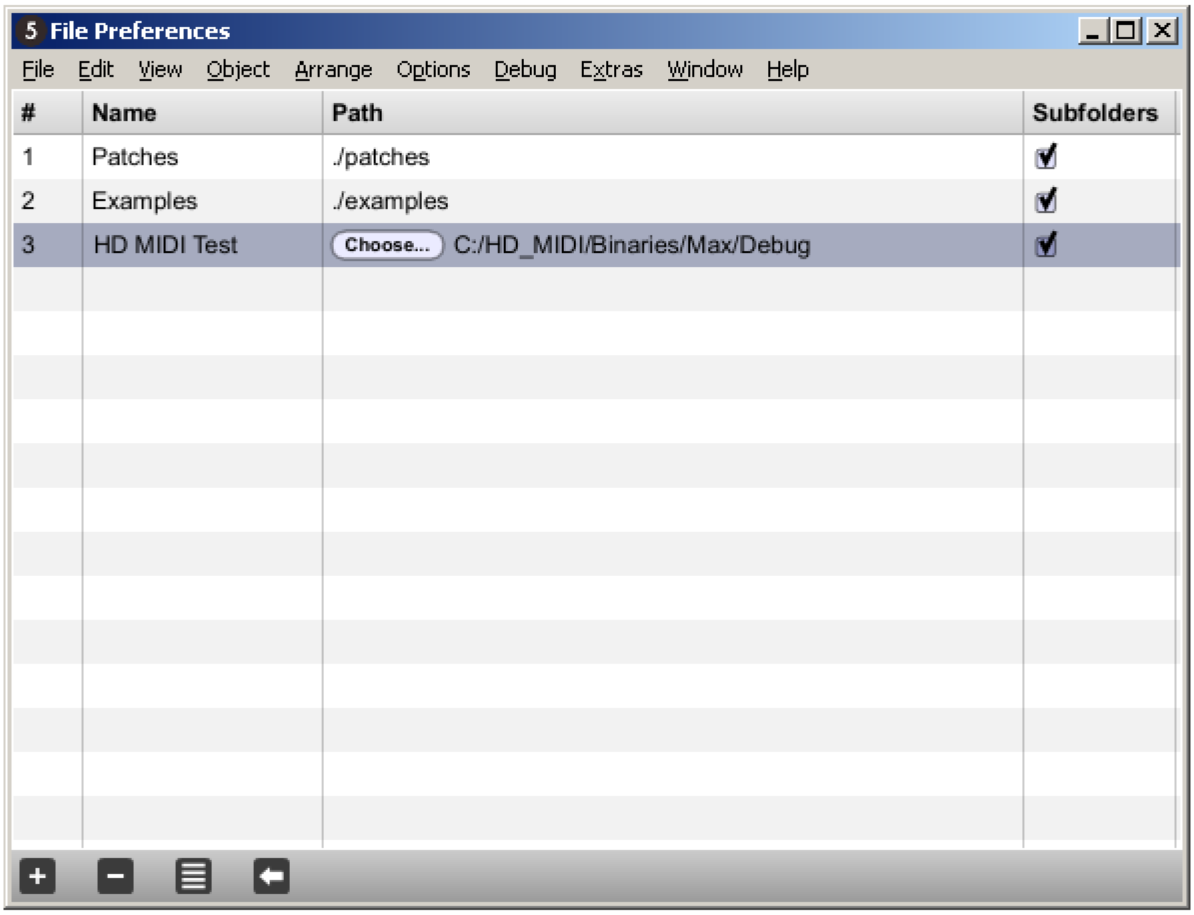
Note for release made in 2013: I wrote this document a when I was still using VS2005 and Max5. The steps to create a Max 6 external are exactly the same however.

Preparing the computer

I assume that Visual Studio (Express or Developper edition) has been installed on your computer and you know how to start and use it. I also assume that Max SDK has been downloaded and installed on your hard disk. Personnally, I have created a c:\Max\SDK directory where the SDK has been unzipped.

To avoid mixups with "original" Max externals, I have also created a specific directory where I put all the compiled files. Make sure that Max is pointing to this directory by setting file preferences, so it will be able to find the externals you have created.

Max / Options / File preferences


File Preferences
File Edit View Object Arrange Options Debug Extras Window Help
Choose...

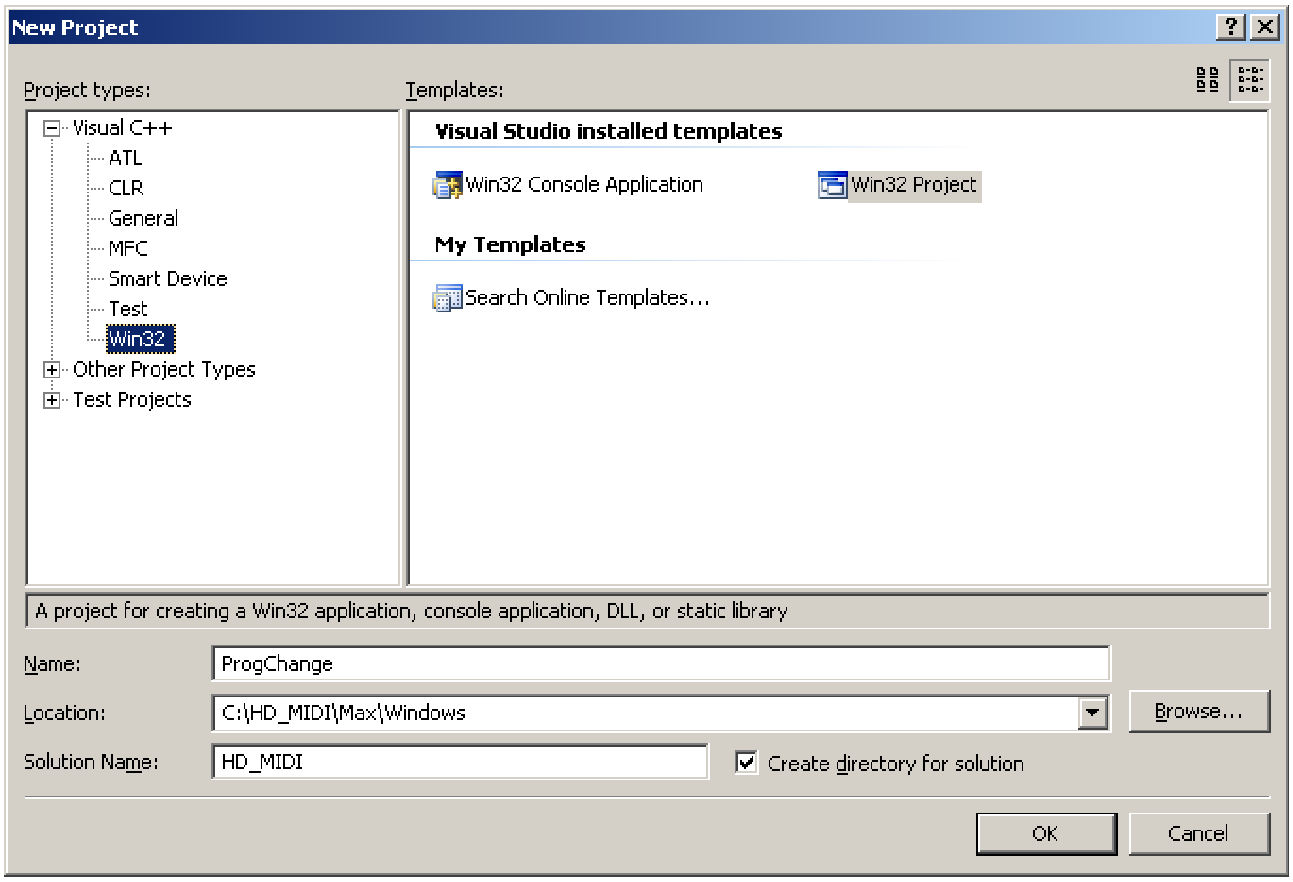
#	Name	Path	Subfolders
1	Patches	./patches	✓
2	Examples	./examples	✓
3	HD MIDI Test	C:/HD_MIDI/Binaries/Max/Debug	✓

Click on "+" to create a new entry, then click "Choose" to select the folder you want to use. If you check Subfolders, Max will recurse the selected directory to find all files of interest.

Creating the project in Visual Studio

Start Visual Studio
In File menu, select New then Project. Select Win32 project
Enter the project name (e.g : hd.progchange). You can use here the name which will appear as Max external, or use a more symbolic name and define the output filename (which will be the external name) later in the project properties.
Click OK when done.

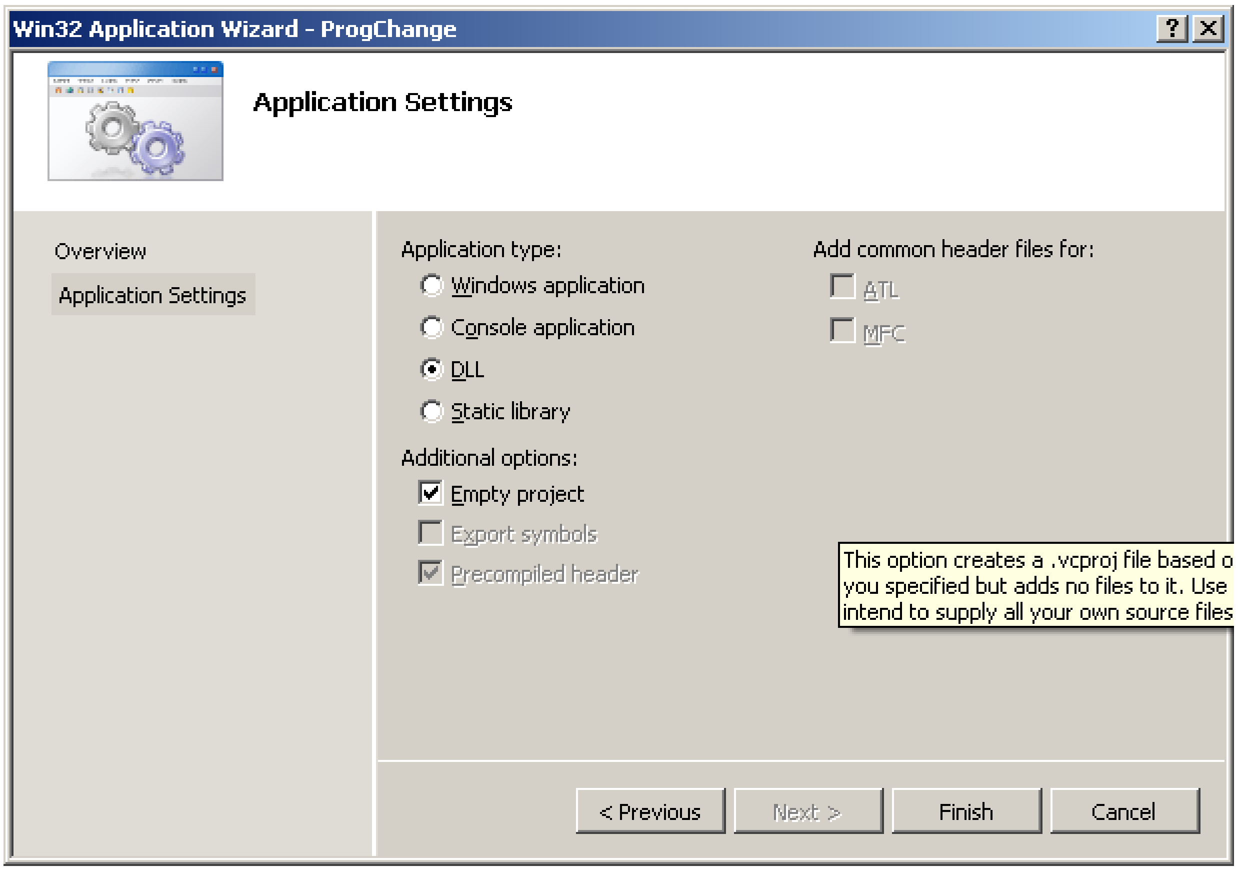
Solution is the name given by Microsoft to a project group. You can choose to create a new project group, or use an existing one (to add your project in an already existing solution)



A project for creating a Win32 application, console application, DLL, or static library

Name: ProgChange
Location: C:\HD_MIDI\Max\Windows
Solution Name: HD_MIDI
[✓] Create directory for solution

Click Next on the windows which appears then select "DLL" and check "Empty project"


Win32 Application Wizard - ProgChange

Application Settings

Application type:

- ( ) Windows application
- ( ) Console application
- (●) DLL
- ( ) Static library

Additional options:

- [✓] Empty project
- [ ] Export symbols
- [✓] Precompiled header

Add common header files for:

- [ ] ATL
- [ ] MFC

This option creates a .vcproj file based o
you specified but adds no files to it. Use
intend to supply all your own source files

The project (and eventually the solution) is then created. Visual Studio puts the files in the directory specified during project creation, which may not fit your personal needs or tastes. In that case, close Visual Studio and rearrange the project directories depending on your needs. When you reopen the solution file, Visual Studio may indicate that it does not find the project anymore. Delete the project you see in the solution explorer (which should be grayed in that case), then right click on the solution and choose Add / Existing project. Select the .vcproj file you moved manually to make it available in the solution.

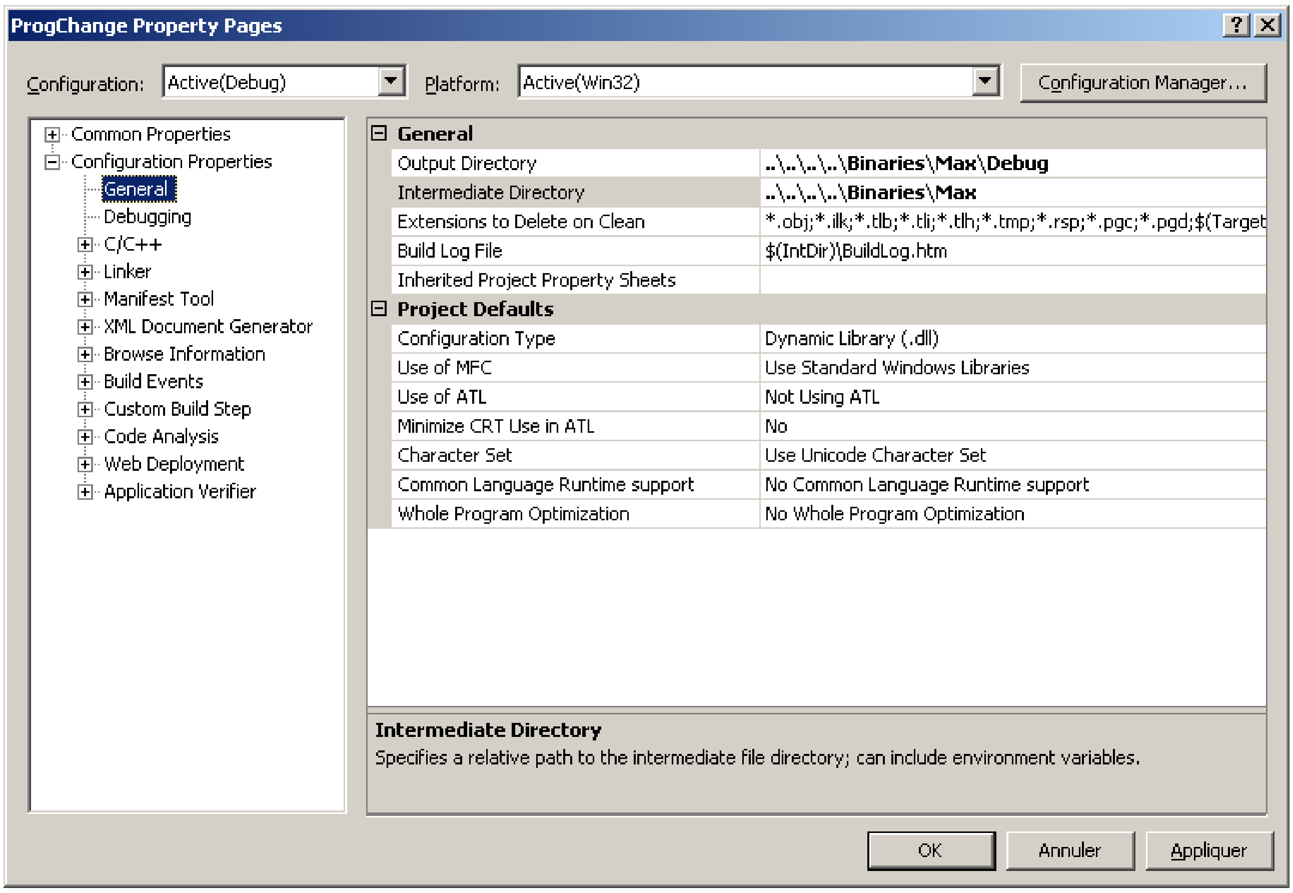
If you selected "Empty project" when the project has been created, it is not yet possible to define all project properties, since the project does not know what language is used (and is then unable to know the related options).

Add the different project source files (if you want to make a quick and dirty test, just pick up simplemax.c and simplemax.def files from SDK and put them in your project)

Open the project properties (Project / Properties or right click on the project in the solution explorer)

Go to General entry

Select the directory where output and intermediate files will be written by the compiler. I personnally use a specific directory, outside from the project, to avoid to copy the compiled files when I make a project backup.



ProgChange Property Pages

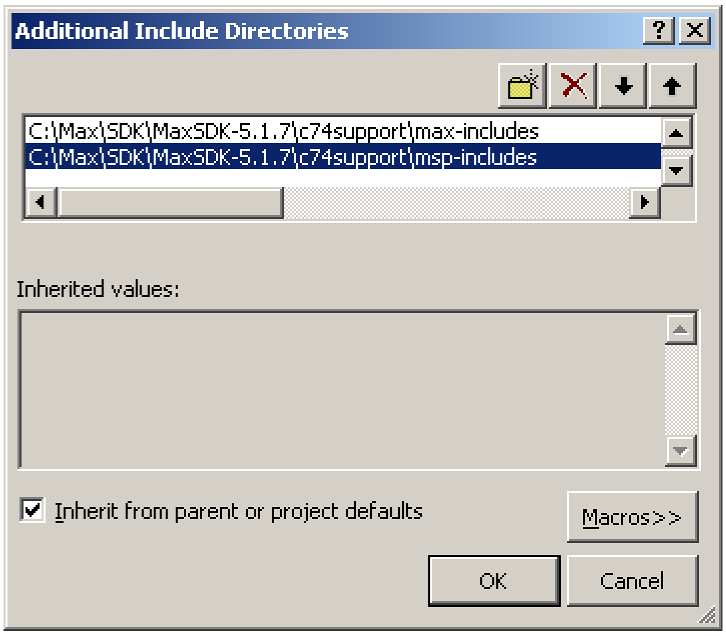
Configuration: Active(Debug) Platform: Active(Win32)

General	
Output Directory	**..\..\..\..\Binaries\Max\Debug**
Intermediate Directory	**..\..\..\..\Binaries\Max**
Extensions to Delete on Clean	*.obj;*.ilk;*.tlb;*.tli;*.tlh;*.tmp;*.rsp;*.pgc;*.pgd;$(Target
Build Log File	$(IntDir)\BuildLog.htm
Inherited Project Property Sheets	
Project Defaults	
Configuration Type	Dynamic Library (.dll)
Use of MFC	Use Standard Windows Libraries
Use of ATL	Not Using ATL
Minimize CRT Use in ATL	No
Character Set	Use Unicode Character Set
Common Language Runtime support	No Common Language Runtime support
Whole Program Optimization	No Whole Program Optimization

Intermediate Directory
Specifies a relative path to the intermediate file directory; can include environment variables.

Go in the C/C++ entry, then select "General" page
((SCREENSHOT))

Enter the Max SDK include directories in "Additional Include Directories"



Additional Include Directories

```
C:\Max\SDK\MaxSDK-5.1.7\c74support\max-includes
C:\Max\SDK\MaxSDK-5.1.7\c74support\msp-includes
```

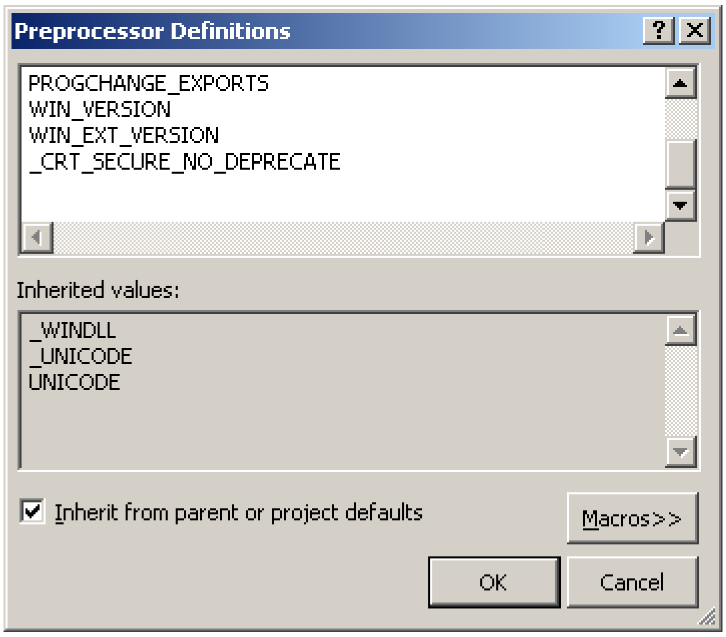
Inherited values:

☑ Inherit from parent or project defaults

Go into Preprocessor, and open Preprocessor Definitions

Define preprocessor options for Windows target
WIN_VERSION
WIN_EXT_VERSION

If your external uses the sprintf function, it is recommended to also add _CRT_SECURE_NO_DEPRECATE to avoid multiple warnings when compiling.



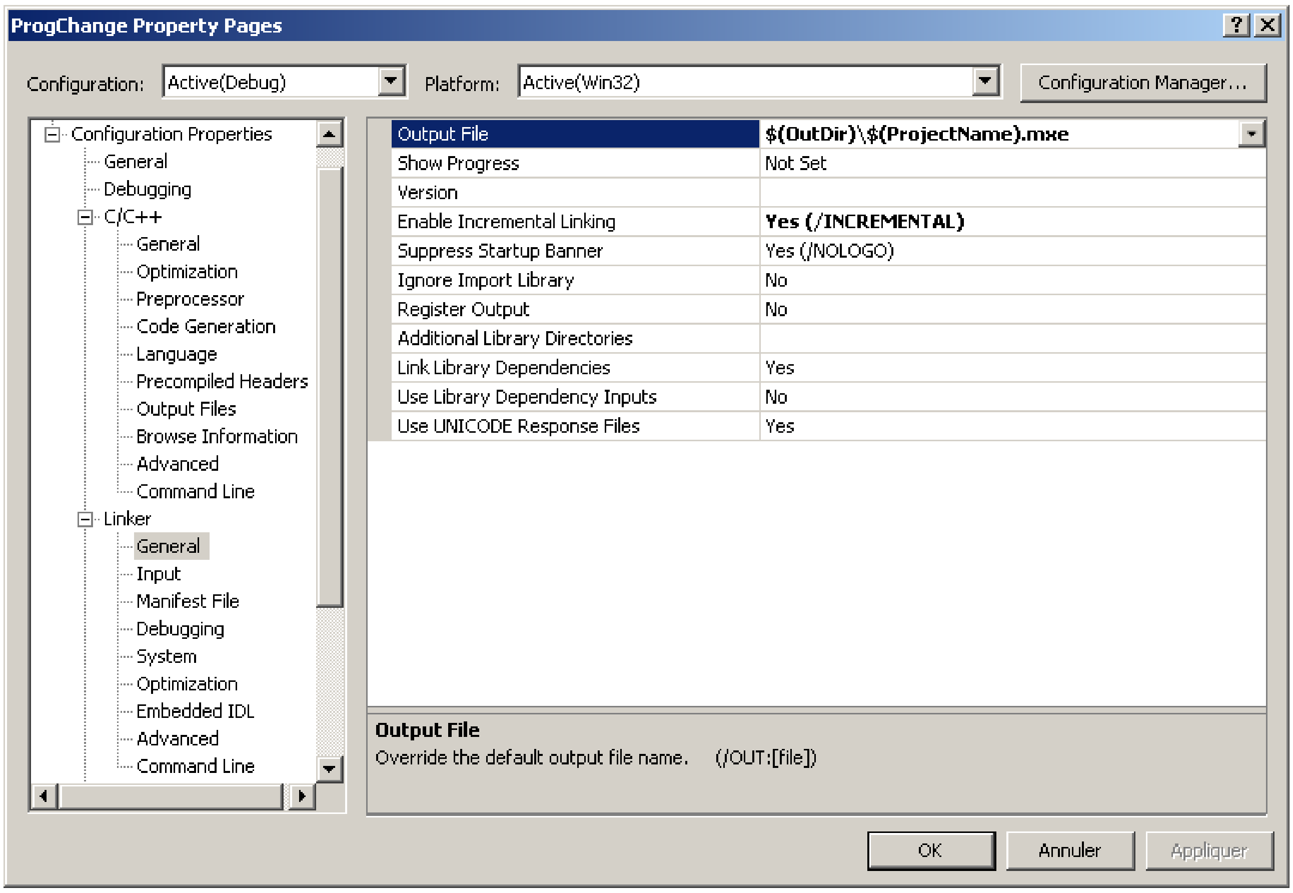
```
PROGCHANGE_EXPORTS
WIN_VERSION
WIN_EXT_VERSION
_CRT_SECURE_NO_DEPRECATE
```

Inherited values:

```
_WINDLL
_UNICODE
UNICODE
```

Go to Linker / General

Define the location and name of output file. **This is the name of the Max external, so it must be a .mxe!**

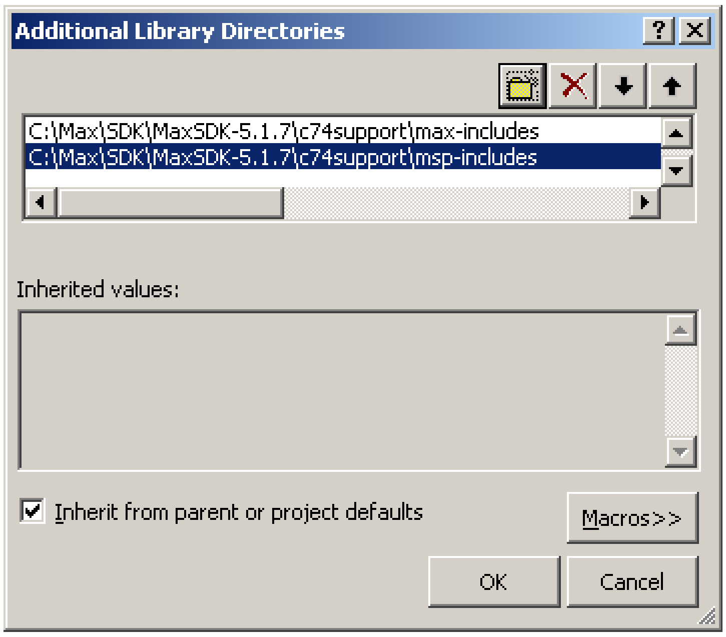


Configuration: Active(Debug) Platform: Active(Win32)

Output File	$(OutDir)\$(ProjectName).mxe
Show Progress	Not Set
Version	
Enable Incremental Linking	Yes (/INCREMENTAL)
Suppress Startup Banner	Yes (/NOLOGO)
Ignore Import Library	No
Register Output	No
Additional Library Directories	
Link Library Dependencies	Yes
Use Library Dependency Inputs	No
Use UNICODE Response Files	Yes

Output File
Override the default output file name. (/OUT:[file])

Define the path to Max external libraries (Linker / General / Additional Library Directories)


Additional Library Directories

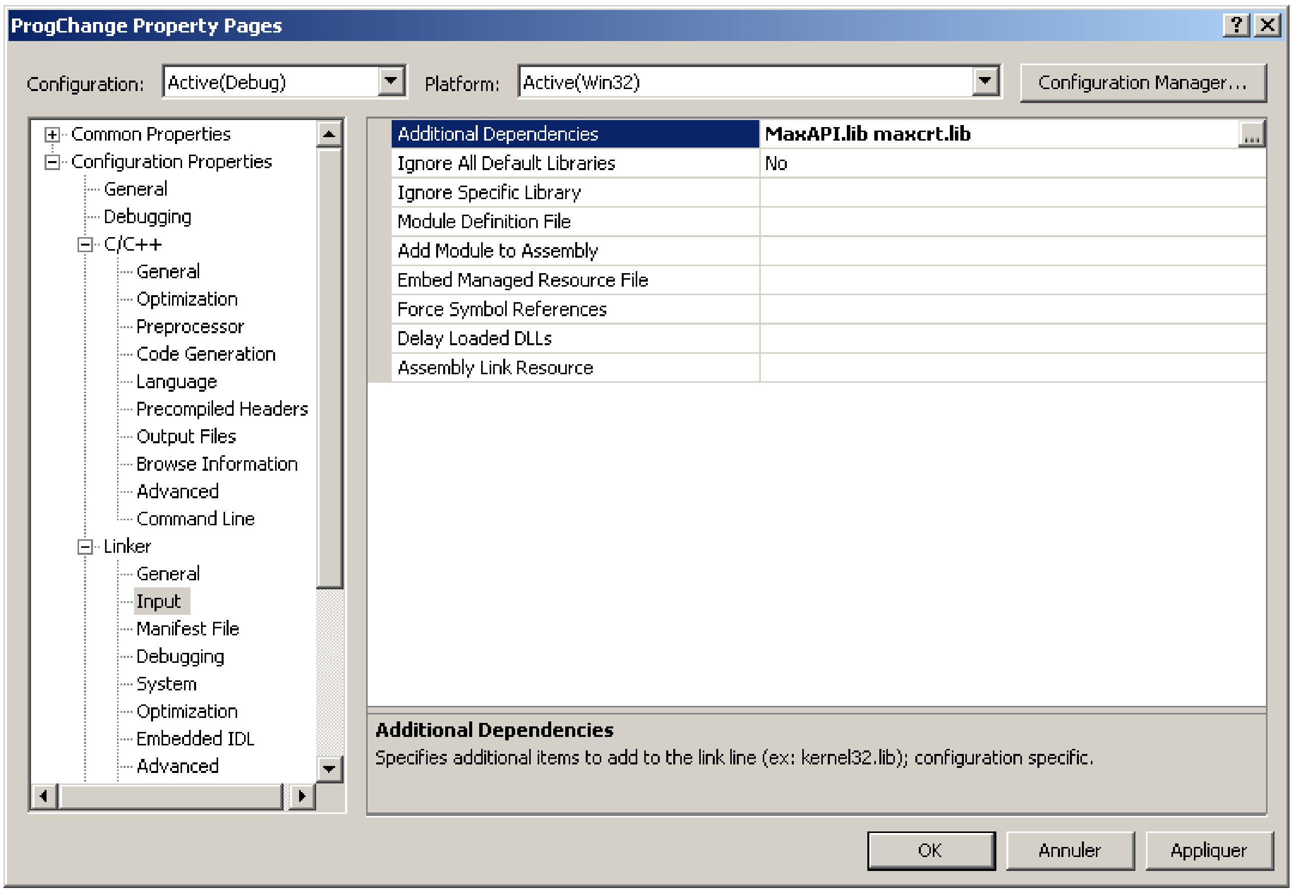
C:\Max\SDK\MaxSDK-5.1.7\c74support\max-includes
C:\Max\SDK\MaxSDK-5.1.7\c74support\msp-includes

Inherited values:

☑ Inherit from parent or project defaults

Link with Max external libraries (Linker / Input / Additional Dependencies)
MaxAPI.lib
maxcrt.lib

WARNING : I had to "Ignore all default libraries" otherwise I was getting a warning at compilation.

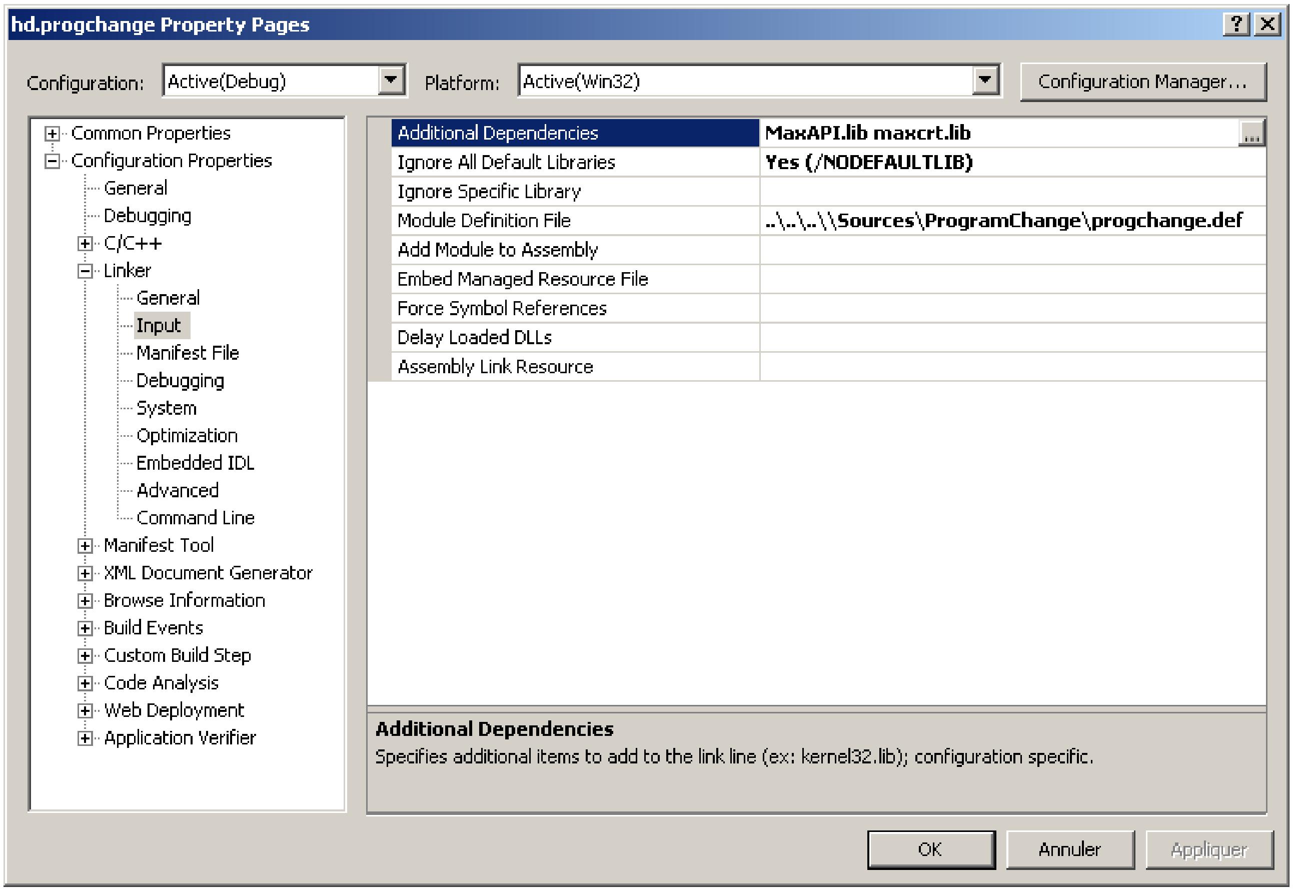

ProgChange Property Pages

Configuration: Active(Debug) Platform: Active(Win32)

Additional Dependencies	MaxAPI.lib maxcrt.lib
Ignore All Default Libraries	No
Ignore Specific Library	
Module Definition File	
Add Module to Assembly	
Embed Managed Resource File	
Force Symbol References	
Delay Loaded DLLs	
Assembly Link Resource	

Additional Dependencies
Specifies additional items to add to the link line (ex: kernel32.lib); configuration specific.

Add the module definition file (otherwise the DLL can not be loaded by Max).



hd.progchange Property Pages

Configuration: Active(Debug) Platform: Active(Win32)

Additional Dependencies	MaxAPI.lib maxcrt.lib
Ignore All Default Libraries	Yes (/NODEFAULTLIB)
Ignore Specific Library	
Module Definition File	..\..\..\\Sources\ProgramChange\progchange.def
Add Module to Assembly	
Embed Managed Resource File	
Force Symbol References	
Delay Loaded DLLs	
Assembly Link Resource	

Additional Dependencies
Specifies additional items to add to the link line (ex: kernel32.lib); configuration specific.

You can now compile your external and test it in Max.

And to finish...

Do not forget that two project configurations are created automatically by Visual Studio (Debug and Release). You can create yourself other configurations if needed.

You will need to perform the same configuration steps for each project configuration, otherwise you will experience compilation errors and warnings when you will want to create a Release version.