Building GDAL 1.4.2 for Windows using Visual Studio

V. 1.0

Daniele Romagnoli

Dott. Ing. Simone Giannecchini

1. Introduction

This simple guide will provide you some instructions about how to build GDAL 1.4.2 with support for the following formats:

- MrSID (v 6.0.7)
- ECW (v 3.3)
- Kakadu (v. 5.2.6)
- HDF4 (v. 4.2r1)

2. Preliminary steps

In order to add support for the listed formats, you need to achieve some preliminary steps for each format.

2.1 - ECW

Download the Image Compression SDK (source code) from ER Mapper site at this address:

http://www.ermapper.com (You need to be registered in order to download it).

From the main site, select the menu “products & downloads” -> “Image Compression SDK” from the Desktop section. Download the Image Compression SDK Source Code 3.3 file and extract this somewhere on your hard disk, as an instance on `C:\ExternalLibraries\libecwj2-3.3`.

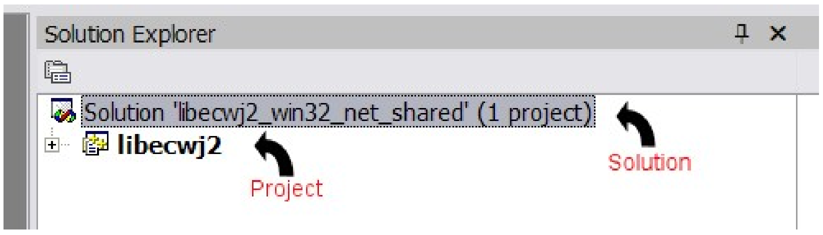
Be sure you have Microsoft Windows® Server 2003 R2 Platform SDK installed.

If not yet installed, download it from this location:

http://www.microsoft.com/downloads/details.aspx?FamilyID=484269e2-3b89-47e3-8eb7-1f2be6d7123a&DisplayLang=en

Then, open the ready-to-use `libecwj2_win32_net_shared.vcproj` Visual Studio Project available in `C:\ExternalLibraries\libecwj2-3.3\Source\NCSBuildQmake`.

You will be asked to create a new solution. Select a location where to create the solution and, when done, change the solution properties (right click on the solution -> Properties).



Select “Configuration Properties” and switch the Configuration value from “Debug” to “Release”.

Finally, select the libecwj2 project in your solution explorer and change its properties as follow:

Configuration Properties -> C/C++ -> General -> Additional Include Directories:

add an entry referring the include subfolder of the Microsoft Windows® Server 2003 R2 Platform SDK (As an instance: “`C:\ProgramFiles\Microsoft Platform SDK for Windows Server 2003 R2\Include`”).

Then, you are ready to build your solution.

2.2 - MrSID

As a first requirement, you need the LizardTech Decoding Software Development Kit (DSDK).

You can download it free of charge from this site:

`http://developer.lizardtech.com` (You need to be registered in order to download it).

After logged in, select “Download” -> “Software Development Kits” -> “Download SDK's”.

Select the proper version of SDK to be download.

When your download is completed, unzip the DSDK somewhere on your hard disk, as an instance, on `C:\ExternalLibraries\MrSid`

2.3 - KAKADU

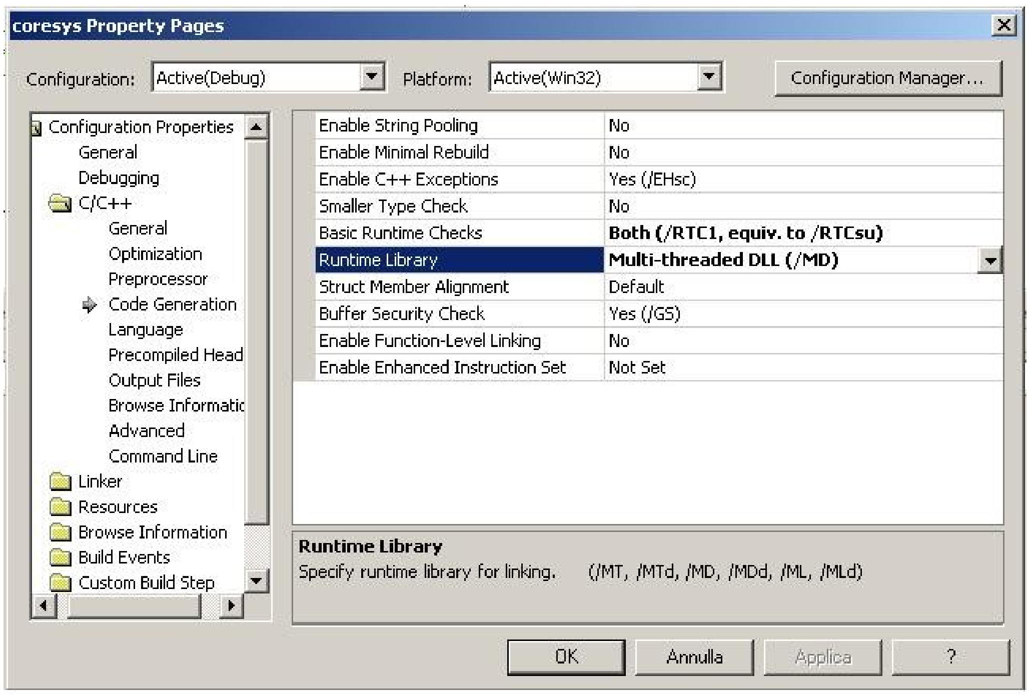
The visual studio solution for kakadu building allows to build a shared debug DLL.

We need to change some settings to build a shared Release DLL.

Otherwise, sometimes, memory allocations errors occurs since some libs use MSVCRT71D and some others MSVCR71.

First step is open the proper ready-to-use Visual Studio Solution[1] of kakadu coresys (located in `kakadu\VERSION\coresys`) and change the coresys configuration properties by changing the option C/C++->Code Generation -> RunTime Library from /MDd to /MD.

[1]As an instance, if you are using Visual Studio .Net 2003, open the `coresys_2003.sln` solution file.



Enable String Pooling	No
Enable Minimal Rebuild	No
Enable C++ Exceptions	Yes (/EHsc)
Smaller Type Check	No
Basic Runtime Checks	**Both (/RTC1, equiv. to /RTCsu)**
Runtime Library	**Multi-threaded DLL (/MD)**
Struct Member Alignment	Default
Buffer Security Check	Yes (/GS)
Enable Function-Level Linking	No
Enable Enhanced Instruction Set	Not Set

Runtime Library
Specify runtime library for linking. (/MT, /MTd, /MD, /MDd, /ML, /MLd)

Then, you are ready to build your solution.

After you done this, open the Kakadu apps solution (located in `kakadu\VERSION\apps`). Change the Code Generation runtime library of each project from /MDd to /MD and build them.

Then, copy the produced `.obj` files in the `kakadu\VERSION\apps\make` folder. The required `.obj` and the location from where to get them are listed in the following table.

File	Originating location
`args.obj`	`\v5_generated\v_compress\debug`
`2.obj`	`\v5_generated\v_compress\debug`
`mj2.obj`	`\v5_generated\v_compress\debug`
`image_in.obj`	`\v5_generated\compress\debug`
`palette.obj`	`\v5_generated\compress\debug`
`roi_sources.obj`	`\v5_generated\compress\debug`
`kdu_tiff.obj`	`\v5_generated\compress\debug`
`jpx.obj`	`\v5_generated\compress\debug`
`kdu_stripe_decompressor.obj`	`\v5_generated\buffer_expand\debug`
`image_out.obj`	`\v5_generated\expand\debug`

2.4 - HDF4

As a first requirement, you need to download the binary distribution of HDF4 release from this site:

http://hdf.ncsa.uiuc.edu/release4/obtain.html

Scroll this page until you find the link to the binary distribution file for Windows.

When your download is completed, unzip the binary somewhere on your hard disk, as an instance, on `C:\ExternalLibraries\HDF4`

Then, enter in the `release` subfolder and create a `libpath` folder where you need to copy 4 `*.lib` files contained in `release\lib` and `release\dll` subfolders. They are: `hd421.lib, hd421m.lib, hm421.lib, hm421m.lib`

This could seem a strange workaround but it avoid errors when building GDAL against HDF4.

It is finally worth to point out that HDF4 leverages on some external libs: JPEG, ZLIB, SZIP. Be sure you have them. http://hdf.ncsa.uiuc.edu/release4/obtain.html also contains 3 links where to download the required libraries. For each library, you need to select the “Pre-Compiled Binaries” link and select the proper version (for Windows).

3. Configuring GDAL

Firstly, you need to download GDAL 1.4.2 from OSGeo SVN at this location:
https://svn.osgeo.org/gdal/tags/1.4.2/gdal

You may use Tortoise SVN (available at: http://tortoisesvn.net/downloads) to download it.

Then, you need to apply the patch available at this location:

https://imageio-ext.dev.java.net/svn/imageio-ext/trunk/patches/1.4.2GDAL.patch

This patch contains several changes for:

- Kakadu support: Multithreading added; More Kakadu options supported. Makefile modified
- Java bindings: improved data access (read Dataset at once instead of read RasterBands at once; PixelSpace, LineSpace and BandSpace parameters now are allowed)

To apply the patch, supposing you already downloaded Tortoise SVN, you need to select the `GDAL` folder on your hard-disk, then right-button's click on it with your mouse and select TortoiseSVN ->Apply patch... At this point, you need to specify the previously downloaded `1.4.2GDAL.patch` file and apply.

Finally, you need to modify your `GDAL\NMAKE.opt` as explained in the following sections.

3.1 - KAKADU SETTINGS

Find the KAKADU Setting properties by looking for the following line:

```
# Uncommment if you have Kakadu 4.0 or newer
```

Supposing your KAKADU library has been placed in `C:\ExternalLibraries\kakadu`, edit the next line, like this:

```
KAKDIR = C:\ExternalLibraries\kakadu\v5_2_6-90032L
```

Be sure the proper version subfolder is set. (In this case: v5_2_6-90032L).

To enable kakadu support we need to change only another property. Go ahead in the `NMAKE.OPT` and look for the following line:

```
# Any extra libraries needed on this platform?
```

Then, edit the `ADD_LIBS` variable by adding the kakadu lib, like this:

```
ADD_LIBS = C:\ExternalLibraries\kakadu\lib\kdu_v52D.lib
```

3.2 - MrSID SETTINGS

Find the MrSID Setting properties by looking for the following line:

```
#Uncomment the following for MrSID support
```

Supposing your MrSID library has been placed in `C:\ExternalLibraries\MrSid`, edit the next lines, like this

```
MRSID_DIR = C:\ExternalLibraries\MrSid

MRSID_INCLUDE = -I$(MRSID_DIR)\include\base -I$(MRSID_DIR)\include\support \

      -I$(MRSID_DIR)\include\metadata \

      -I$(MRSID_DIR)\include\mrsid_readers \

      -I$(MRSID_DIR)\include\j2k_readers

MRSID_LIB = $(MRSID_DIR)\lib\Release_md\lti_dsdk_dll.lib advapi32.lib
user32.lib
```

3.3 - ECW SETTINGS

Find the ECW Setting properties by looking for the following line:

```
# Uncomment the following and update to enable ECW support.
```

Supposing your ECW library has been placed in `C:\ExternalLibraries\libecwj2-3.3`, edit the next 2 lines, like this:

```
ECWDIR  =  E:\work\libecwj2-3.3

ECWLIB  =  $(ECWDIR)\Source\NCSBuildQmake\Debug\libecwj2.lib
```

3.4 - HDF4 SETTINGS

Find the HDF4 Setting properties by looking for the following line:

```
# Uncomment the following and update to enable NCSA HDF Release 4 support.
```

Supposing your HDF4 library has been placed in `C:\ExternalLibraries\HDF4`, edit the next 2 lines, like this:

```
HDF4_DIR = c:\ExternalLibraries\HDF4\release

HDF4_LIB = /LIBPATH:$(HDF4_DIR)\libpath
```

4. Building GDAL

Now, you are ready to build GDAL. Open Visual Studio Command Prompt, and enter in your GDAL home folder. At this point, you are ready to start the build process by running the following command:

```
      nmake /f makefile.vc
```

When the build is terminated, you need to generate JAVA bindings.

5. Generating Java Bindings

5.1 - Requirements

Be sure you have properly downloaded SWIG, the Simplified Wrapper and Interface Generator which allow to produce JAVA bindings for GDAL. You can download it from: http://www.swig.org/

You also need ANT which will be used in order to build a JAR containing all JAVA classes generated by SWIG. You can download it from: http://ant.apache.org/

5.2 - Variable settings

Be sure your `GDAL\NMAKE.opt` has the `SWIG` variable, properly set. Check this variable by finding the following lines:

```
# Set the location of your SWIG installation

!IFNDEF SWIG

SWIG = C:\Programs\swig\swigwin-1.3.29\swig.exe

!ENDIF
```

Be sure `SWIG` variable refer to the proper `swig.exe` path.

Then, check your `GDAL\SWIG\JAVA\java.opt` is properly configured. (Basically, you need to check the `JAVA_HOME` and `ANT_HOME` variables)

5.3 - Running SWIG

Now, you are ready to generate java bindings. From the Command Line, enter in your `GDAL\SWIG` folder and run `nmake /f makefile.vc java`

This command will automatically generate wrappers and bindings.

6. Final Settings

At this point, you should have:

- some external DLLs (Kakadu: `kdu_v52D.dll`, ECW: `libecwj2.dll`, MrSID: `lti_dsdk_dll.dll`, HDF4: `hd421m.dll`, `hm421m.dll`)
- a GDAL DLL (`gdal14.dll`)
- 4 JNI DLL (`gdalconstjni.dll`, `gdaljni.dll`, `ogrjni.dll`, `osrjni.dll`)
- a jar file (`gdal.jar`)

You need to place all these DLLs in the folder where your application will look for libraries. A typical location where to place them is your `JAVA_HOME/JRE/BIN` if you are using JDK's JRE, or `JAVA_HOME/BIN` if you are using JRE.

Finally, `gdal.jar` needs to be used as dependency of your projects using GDAL Libraries and related java bindings.