How to build MPTK with CMake

SUMMARY

Read this document to learn how to build the Matching Pursuit Tool Kit on Win32 platform using CMake and Visual Studio

LAYOUT

1 Getting Started

1.1 Required tools

- Necessary:
  - CMake (tested with version 2.8.0) [for further informations see appendix 4.1]
  - Microsoft Visual Studio (tested with Visual C++ 2008 Express)
  - Microsoft Windows SDK
  - NSIS Nullsoft Scriptable install system (tested with version 2.46)
- Optional (if Matlab interface is needed):
  - Matlab at least version R2006
    - Caution : Matlab MUST be installed in the default directory
      - For Windows : C:\Program Files\MATLAB

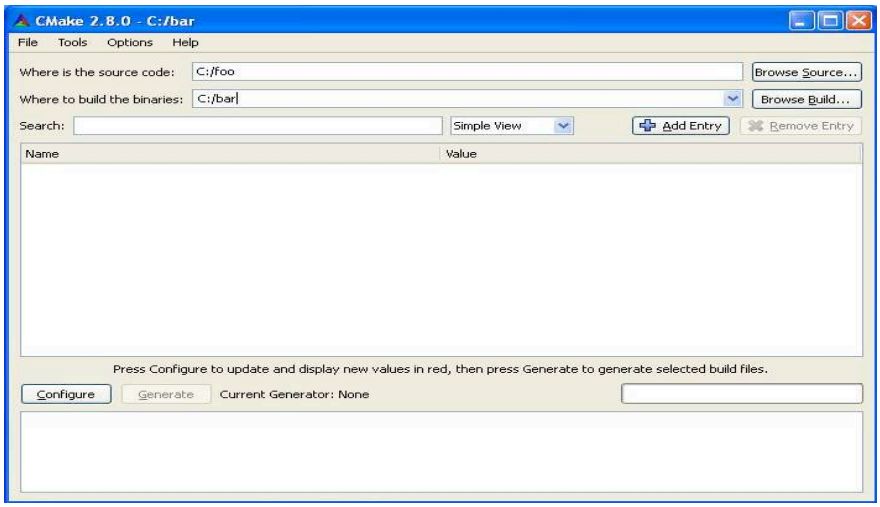
1.2 Required package

- MPTK: at least version 0.5.7
- Libsndfile: (tested with version 1.0.21) pre compiled dll [for further informations on how to get it and where to install the file, see appendix 4.2]
- FFTW: (tested with version 3.2.2) pre compiled dll [for further informations on how to get it and where to install the file, see appendix 4.3]
- Pthread-w32: (tested with version 2.8.0) pre compiled dll [for further informations on how to get it and where to install the file, see appendix 4.4]

1.3 Basic build

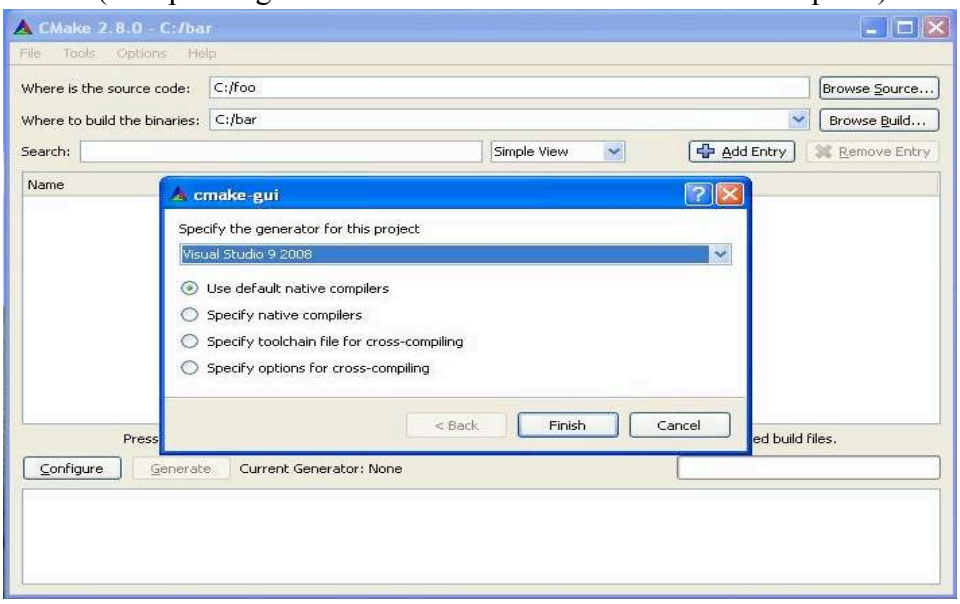
If the required libraries and include files are installed as the appendix 4.1 to 4.4 indicate it, you should be able to generate a build system using the basic build settings.

Suppose you unzip MPTK-v.v.v.tar.gz in directory c:\foo and you want to build MPTK files in c:\bar (you may choose to build for example in a temporary location which will differ from the place where you will eventually install MPTK). Use CMake to generate the build system, use the Make icons from the program menu to launch the CMake configuration application:

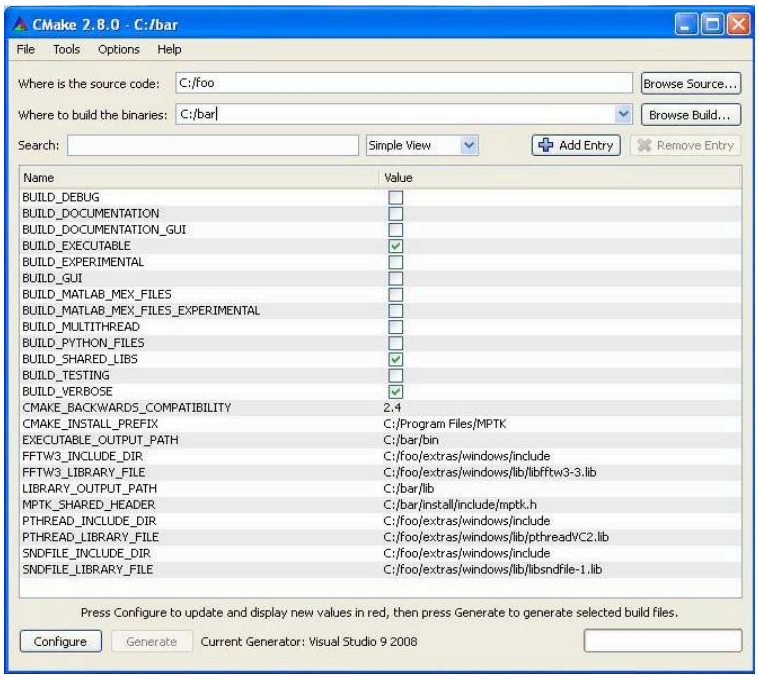


Set the « Where is the source code: » text box with the path of the directory where the source files are located (c:\foo) and the « Where to build the binaries: » with the path of the directory where you want to build the library and executable files (c:\bar).

When clicking for the first time on the [Configure] button, CMake will ask for the build tool you want to use. The build system type depends on the builder you want to use, in our case this is the Visual Studio X (X depending the version of Visual installed on the computer) chain tools:



When pressing again the [Configure] button to configure the build system, CMake performs a list of tests to determine the system configuration and manage the build system. If the configuration is correct then no pop-up will appears during the tests and CMake finally shows the various options of the build underlaid in grey. In case of a configuration issue, a pop up window warns you about this issue indicating which test has failed, in this case the build option in the CMake application software will be underlaid in red. We will discuss in Section 2.2.3 what to do in such a case, but let us for the moment assume that everything ran smoothly.



Once the build system configured then generated, you have to actually build MPTK, using Visual.

2 Custom build and install

If you want to change build and/or install options (for example to install MPTK in a different folder or to enable the parallel computing using multithread abilities provided by the OS).

2.1 Principle

In order to build the MPTK library and executable files, you have to generate a build system with CMake This build system can be parametrized using the CMake application software, which is also used to fix potential issues in the build system configuration.

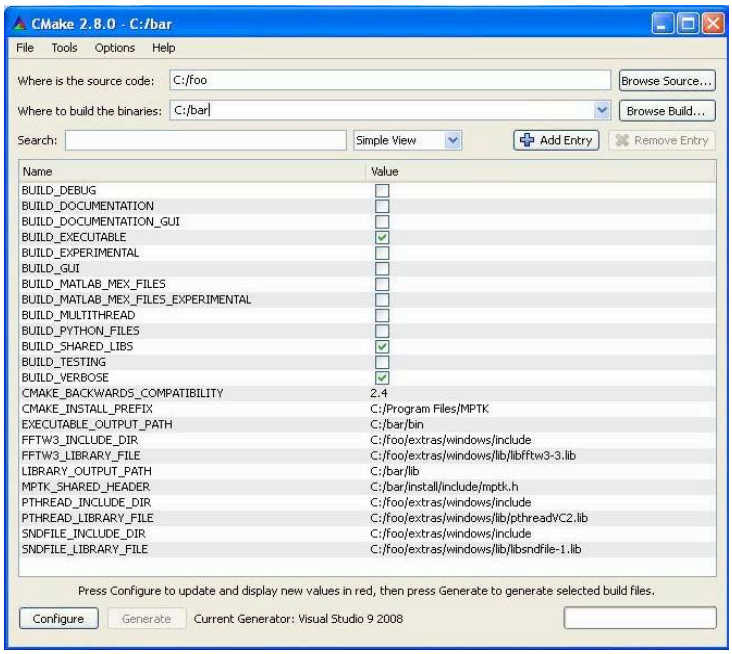
2.2 Pre-configuration using CMake application software

2.2.1 Setting build and install option

Using the Graphical User Interface with the CMake application software allows you to:

- Specify build system options, such as whether to enable the use of multiple threads to speed up the Matching Pursuit decomposition or whether to generate the doxygen documentation.
- Troubleshoot the various paths required to build MPTK
- Display advanced build options via the advanced mode

When using the CMake application software, the CMake GUI appears:



When scrolling on a value and clicking, this value can be edited, the grey underlaid row under the button displays some informations about the option and required path to create the build system. In the case of clicking on an option, a Combo Box appears to toggle the ON/OFF values. For a path field you can edit manually or use the file browser in order to find the correct path.

After choosing options for the build and setting the required fields, click on [Configure] to configure. The configuration of the build system is checked again by CMake application software, at the end of this check if the build settings are correct, you can press [Generate] in order to generate the build system.

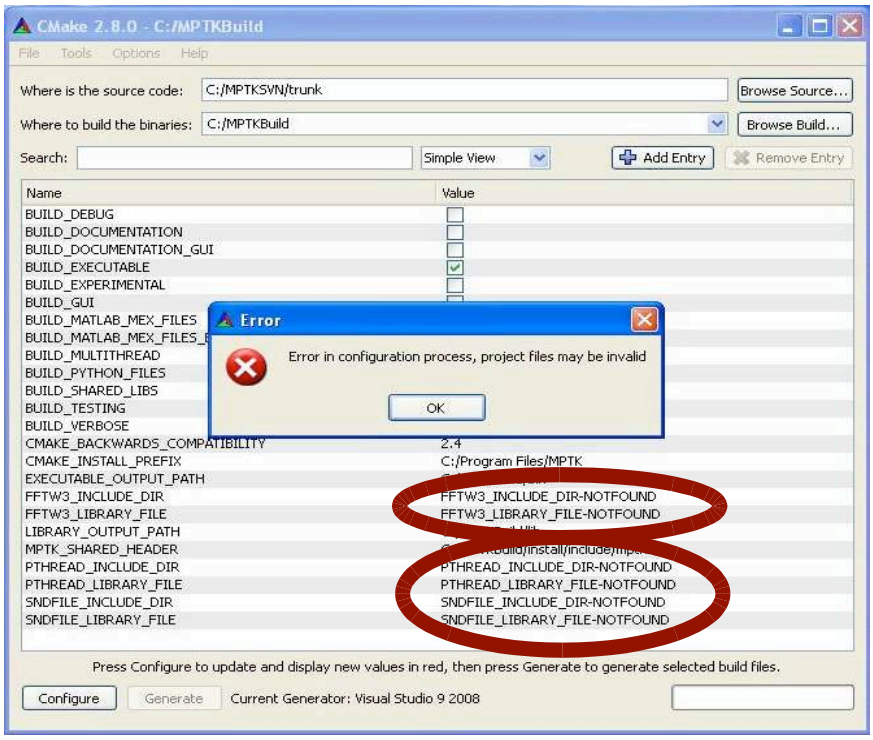
2.2.2 Main options

This Graphical User Interface allows the user to set the build options for MPTK, currently named BUILD_SOMETHING, if you want to:

- Install MPTK executable files in a specific directory: set CMAKE_INSTALL_PREFIX with the path of this directory. (by default: C:\Program Files\MPTK). In this directory, the bin directory contains the command line executable files and the needed dll (pthreadVC2.dll, libsndfile-1.dll and libfftw3-3.dll), the lib directory contains the libmptk.a and libdsp_windows.a library files and the include directory contains the mptk.h header file.
- Generate the doxygen documentation for MPTK library: set BUILD_DOCUMENTATION to ON. This operation can take a lot of time, be patient. (default OFF)
- Allow MPTK to use parallel computing, set BUILD_MULTITHREAD to ON, this option allows to increase performance on multi-processor hardware architectures

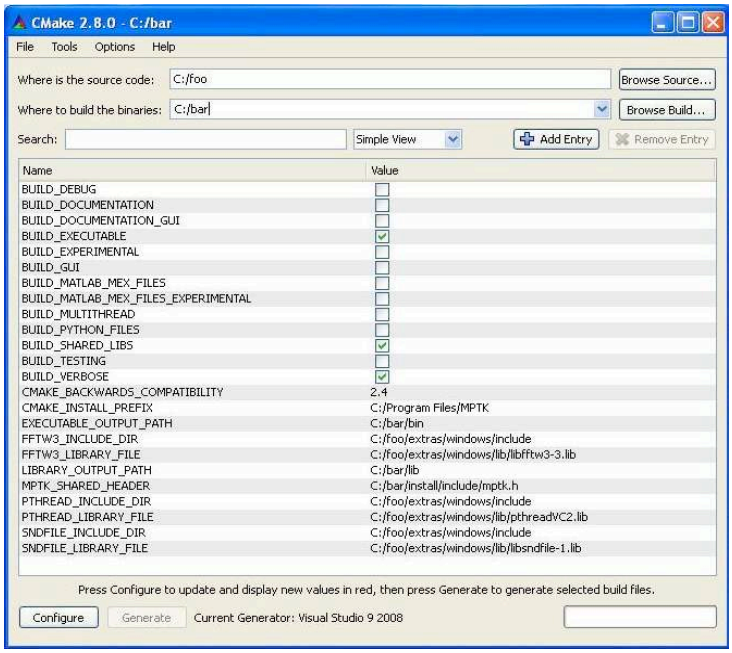
2.2.3 Troubleshooting

Let us see a more complex use of the CMake GUI with troubleshooting. Despite all the integrated tests when CMake manages to generate the build system of MPTK, sometimes CMake application software cannot determine some parameters of your configuration. When using the CMake application software, CMake performs a list of tests on the build system configuration in order to generate it, if there is some issue related to the configuration, the errors list is displayed on a pop up windows. Here we have 3 include directory and files that CMake cannot found:



In this case, `FFTW3_INCLUDE_DIR`, `FFTW3_LIBRARY_FILE`, `PTHREAD_INCLUDE_DIR`, `PTHREAD_LIBRARY_FILE`, `SNDFILE_INCLUDE_DIR`, `SNDFILE_LIBRARY_FILE` fields must be set with the path to the directory containing the include files for FFTW, pThread and sndfile library. To set the fields values simply click on the value and use the file explorer to set the field with the correct path.

Important note: In the case of path to include files `*_INCLUDE_DIR`, only the directory where this include files are located is required. But in the case of a library files `*_LIBRARY_FILE`, the entire path to the library including its name must be indicated in the corresponding field. As in example below:



After choosing the options for the build and setting the required fields, press [Configure] to configure. The configuration of the build system is checked again by CMake, at the end of this check if the build settings are correct, you can press [Generate] in order to generate the build system.

2.2.4 Advanced options

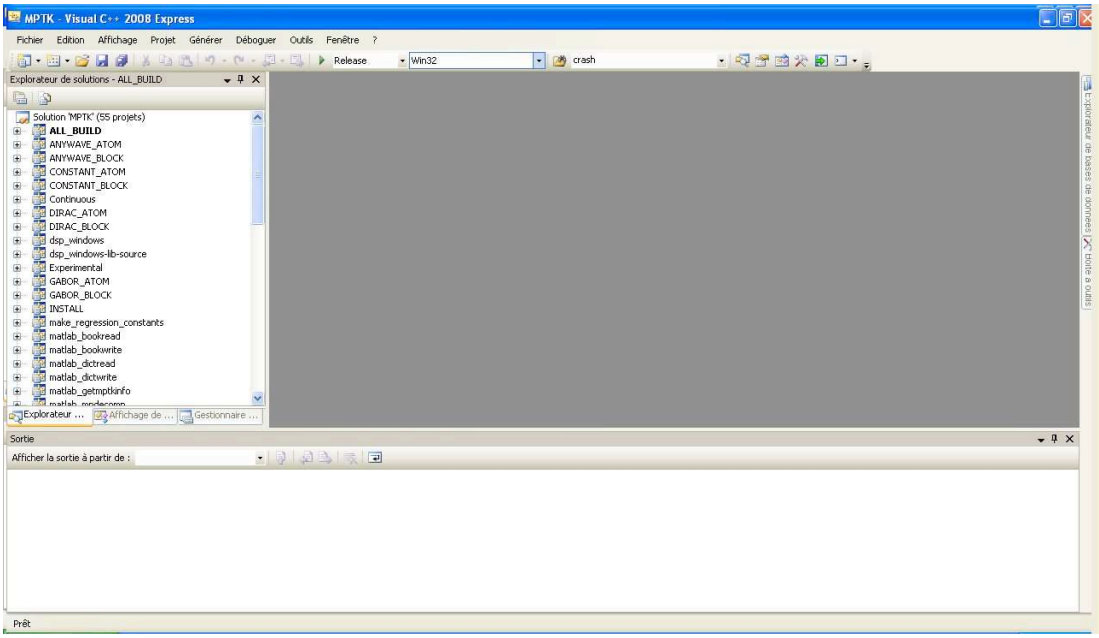
You can display more information and all the configuration options by switching to the advanced mode, which is set to off by default and can be toggled on by using the [Show Advanced Values] check box.

The list of the following options should be considered as advanced:

- BUILD_SHARED_LIBS: this option will install the MPTK library (libdsp_windows.a and libmptk.a) in a lib directory and the include file (mptk.h) in an include directory.
- BUILD_TESTING: this option allows to create tests for MPTK library and executable files using Dart server setting. (default value: OFF)
- BUILD_VERBOSE: this option displays a lot of various informations during the build.
- BUILD_EXECUTABLE: Build the executable files from MPTK (mpd, mpd_demix, mpf , mpview, mpr, mpcat) in a bin directory (default value: ON).

2.3 Build using Visual Studio application software

Once the project is generated using Cmake, the build directory (ex : C:/bar) contains a Visual solution (MPTK.sln). When you double click on this file, Visual Studio software starts and opens the differents projects associated with this solution.



2.3.1 Compilation of MPTK

Right clicking on the project « ALL_BUILD » located on the left tab of Visual Studio, then selecting « Generate » will compile all the MPTK binary files.

2.3.2 Installation of MPTK

Right clicking on the project « INSTALL » located on the left tab of Visual Studio, then selecting «This Project» → « Generate only INSTALL » will install MPTK to the directory associated with the « MPTK_INSTALL_PREFIX » variable of Cmake.

2.3.3 Creation of MPTK package

Right clicking on the project « PACKAGE » located on the left tab of Visual Studio, then selecting «This Project» → « Generate only PACKAGE » will create an archive of MPTK files.

2.3.4 Testing MPTK binary files

If the « BUILD_TESTING » variable of Cmake has been enabled, the project « RUN_TESTS » is available. Selecting «This Project» → « Generate only RUN_TESTS » will run all the tests of MPTK.

2.3.5 Debugging MPTK project

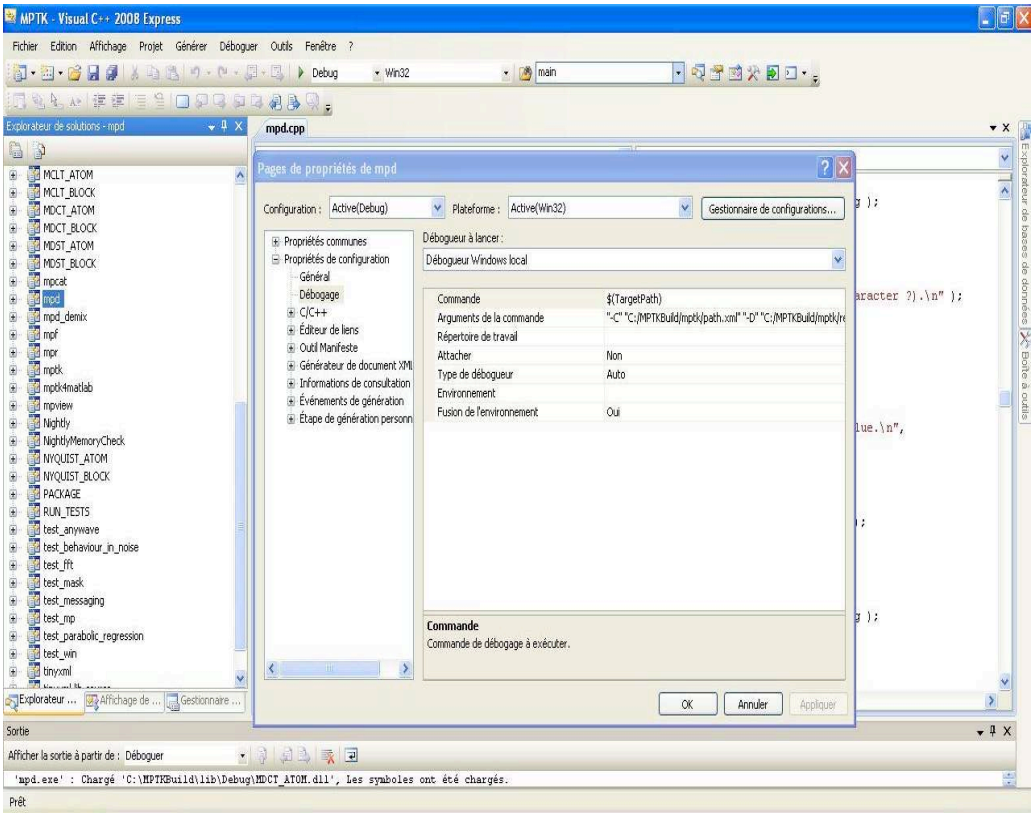
Debugging MPTK project with Visual is possible by two different manners :

- Debugging directly the MPTK executables
- Debugging MPTK through MEX files of Matlab

2.3.5.1 Debugging MPTK executables

Visual Studio development environment's integrated debugging functions allow to stop at procedure locations, inspect memory and register values, change variables, observe message traffic, and get a close look at what the code does.

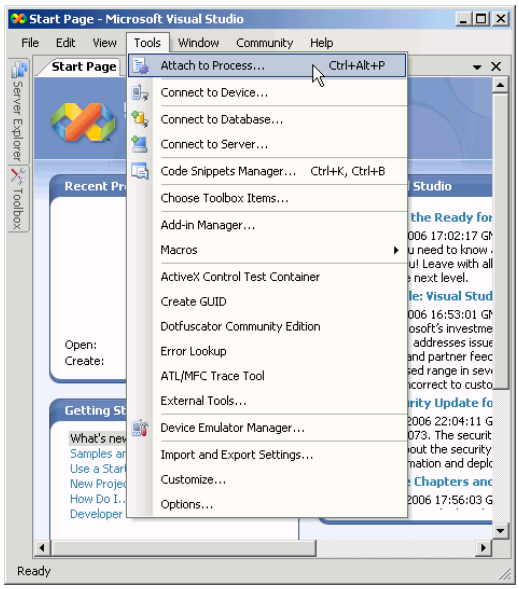
1. Right click on the project you want to debug and select “Properties”
   - It will open a new window containing the parameters of the project
2. On the left tab of this window, select “Configuration properties” → “Debugging”
3. On the right tab, add the command line argument of the executable you want to debug
   - Press Ok to validate the command line argument.
4. Add a break point to one of the source files where you want to Stop
5. Start the debug using right click on the project and “Debugging” → “Start a new instance”



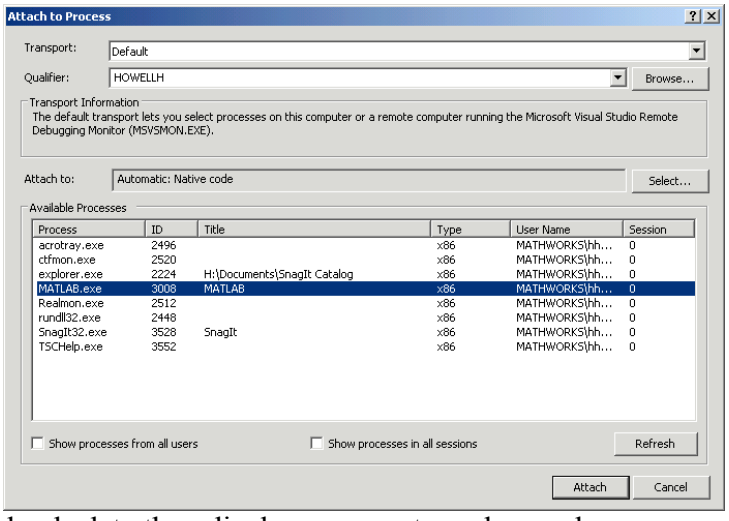
2.3.5.2 Debugging MEX files of Matlab

MEX-files are dynamically linked subroutines produced from C or C++ source code that, when compiled, can be run from within MATLAB in the same way as M-files or built-in functions.

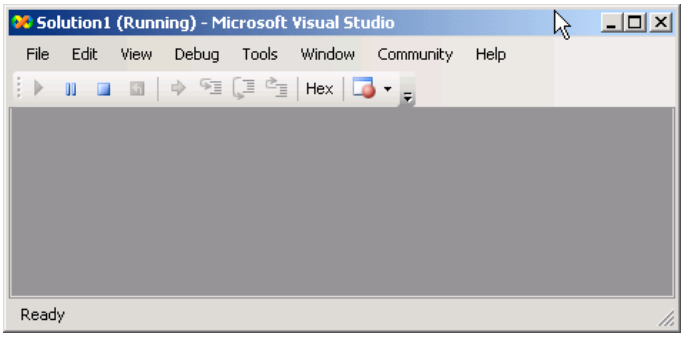
1. Launch Mathworks Matlab®.
2. Start Visual Studio®. Do not exit your MATLAB session.
3. From the Visual Studio "**Tools**" menu, select "**Attach to Process...**"



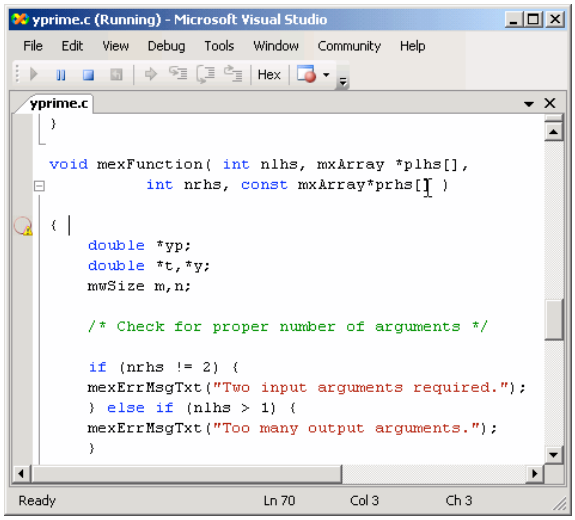
4. In the Attach to Process dialog box, select the MATLAB process and click "**Attach**".



5. Visual Studio loads data then displays an empty code panel.



6. Open the source file you want to debug by selecting **File → Open → File**.
7. Set a breakpoint on the desired line of code. It is often convenient to set a breakpoint at mexFunction to stop at the beginning of the gateway routine. If you have not yet run the executable file, ignore any "!" icon that appears with the breakpoint next to the line of code.



8. Once you hit one of your breakpoints, you can make full use of any commands the debugger provides to examine variables, display memory, or inspect registers.
9. Run the binary MEX-file in MATLAB. After typing a correct command (for example “GettingStarted”, the Visual Studio debugger is waiting at the first breakpoint.

3 Fixing FFTW used plan with wisdom in FFTW configuration

MPTK currently relies on FFTW to compute Fast Fourier Transforms. The principle of FFTW is to adaptively select the fastest running codelet to perform a given FFT, which is good for performance. However, this can lead to non deterministic behaviour of MPTK since two consecutive runs of the same command (e.g. 'mpd') with the same options on the same data can yield slightly different results. To avoid this issue we have implemented a mechanism which allows MPTK to fix which FFT codelet is used for a specific type of FFT computation. To do this, you just need to create and export a environment variable called MPTK_FFTW_WISDOM and set this environment variable with the path of the file you want to use in order to save the FFTW configuration. MPTK will create a « wisdom » file, that will be reloaded each time you use MPTK. It means that the codelets use to compute FFTW plan will be the same between two successive decompositions with the same parameters.

In order to set the environment variable used to store the FFTW wisdom file:

1. Right-click on "My Computer" and select "Properties".
2. Click Advanced -> Environment Variables.
3. In the box entitled "System Variables" and click on new
4. You will be presented with a dialogue box with two text boxes,the head text box has to be set with MPTK_FFTW_WISDOM, the bottom text box allows you to edit the Path of the FFTW wisdom file. Choose a path to store the wisdom file and a name, for example C:\Program Files\MPTK\FFTW_Wisdom.

Note that this file should always be reachable, other way the fftw plan will not be used. You may have to reopen your Windows session in order to set up this new environment variable.

4 Appendix

4.1 Getting CMake

CMake, is a cross-platform, open-source make system. CMake is used to control the software compilation process using simple platform and compiler independent configuration files. CMake generates native makefiles and workspaces that can be used in the compiler environment of your choice. CMake is quite sophisticated: it is possible to support complex environments requiring system configuration, pre-processor generation, code generation, and template instantiation. Please follow the next links in order to learn more about Cmake and download it:

- http://www.cmake.org/cmake/project/about.html
- http://www.cmake.org/cmake/resources/software.html

4.2 Getting libsndfile

Libsndfile is a C library for reading and writing files containing sampled sound (such as MS Windows WAV and the Apple/SGI AIFF format) through one standard library interface. It is released in source code format under the Gnu Lesser General Public License.

This library can be downloaded either through the web (for example via Mega-Nerd website) or in the "relased files" section of the MPTK Gforge:

- http://www.mega-nerd.com/libsndfile/
- http://mptk.gforge.inria.fr/

This package is needed two times by MPTK, the first time during the link of the build process (for both include file and library) and the second time during the execution of MPTK applications. If you don't feel familiar with Win32 dynamic link library management, just follow those instructions:

- "**sndfile.h"** include file in a \extras\windows\include directory of your MPTK source files repertory (for example C:\foo\extras\windows\include)
- unzip the "**libsndfile-1.dll"** and **"libsndfile-1.lib"** files in a \extras\windows\lib directory of your MPTK source files repertory (for example C:\foo\extras\wiunzip the ndows\lib)

4.3 Getting fftw

FFTW is a C subroutine library for computing the discrete Fourier transform (DFT) in one or more dimensions, of arbitrary input size, and of both real and complex data (as well as of even/odd data, i.e. the discrete cosine/sine transforms or DCT/DST).

This library can be downloaded either through the web or in the "relased files" section of the MPTK Gforge:

- http://www.fftw.org/install/windows.html
- http://mptk.gforge.inria.fr/

This package is needed two times by MPTK, the first time during the link of the build process (for both include file and library) and the second time during the execution of MPTK applications. If you don't feel familiar with Win32 dynamic link library management, just follow those instructions:

- unzip the "**fftw3.h"** include file in a \extras\windows\include directory of your MPTK

source files directory (for example C:\foo\extras\windows\include)
- unzip the "**libfftw3-3.dll"** and **"libfftw3-3.lib"** files in a \extras\windows\lib directory of your MPTK source files directory (for example C:\foo\extras\windows\lib)

Important : The static library (LIB) is not included in the package. In order to build it, follow the procedure described:

1. Open Microsoft Visual Studio
2. Go to "Tools" then "Visual Studio command prompt"
3. Configure the dos directory to the location of the fft library folder
   - cd "C:\FolderOf_libfft"
4. Write the following syntax:
   - lib /def:"PathOf_libfft3-3.def"

4.4 Getting Pthreads-w32

The POSIX 1003.1-2001 standard defines an application programming interface (API) for writing multi threaded applications. This interface is known more commonly as *pthreads*. A good number of modern operating systems include a threading library of some kind: Solaris (UI) threads, Win32 threads, DCE threads, DEC threads, or any of the draft revisions of the pthreads standard. The trend is that most of these systems are slowly adopting the pthreads standard API, with application developers following suit to reduce porting woes.

Win32 does not, and is unlikely to ever, support pthreads natively. This project seeks to provide a freely available and high-quality solution to this problem.

Various individuals have been working on independent implementations of this well-documented and standardised threading API, but most of them never see the light of day. The tendency is for people to only implement what they personally need, and that usually does not help others. This project attempts to consolidate these implementations into one implementation of pthreads for Win32.

The source tree, the precompiled dynamic (DLL) and static (LIB) librairies and necessary header files are included in the self-extracting zip file named "pthread-w32-v-v-v-release.exe" which can be downloaded either through the web or in the "relased files" section of the MPTK Gforge:

- http://sourceware.org/pthreads-win32/
- http://mptk.gforge.inria.fr/

This package is needed two times by MPTK, the first time during the link of the build process (for both include file and library) and the second time during the execution of MPTK applications. If you don't feel familiar with Win32 dynamic link library management, just follow those instructions:
Run the self-extracting zip, you should obtain three directory, in the "Pre-built.2" directory:

- copy **«pthread.h», «sched.h», «semaphore.h»** include files in a \extras\windows\include directory of your MPTK source files directory (for example C:\foo\extras\windows\include)
- copy "**pthreadVC2.dll"** and **"pthreadVC2.lib"** files in a \extras\windows\lib directory of your MPTK source files directory (for example C:\foo\extras\windows\lib)

4.5 Help, contact and forums

If you need help with the software:

1. check if a more recent **release** fixes your problem;
2. if not, use the **Help forum** to ask questions.

Other **Forums** are available for open discussions about the Matching Pursuit algorithm and its MPTK implementation.

Some articles exposing scientific results related to MPTK are available in PDF format through the **Released Files** page.

If you want specific informations, or if you want to communicate privately with us, please write to matching.pursuit@irisa.fr.

Request for help sent to this address won't be answered. Please use the **Help forum** instead.

Thank you for your interest in The Matching Pursuit Tool Kit !