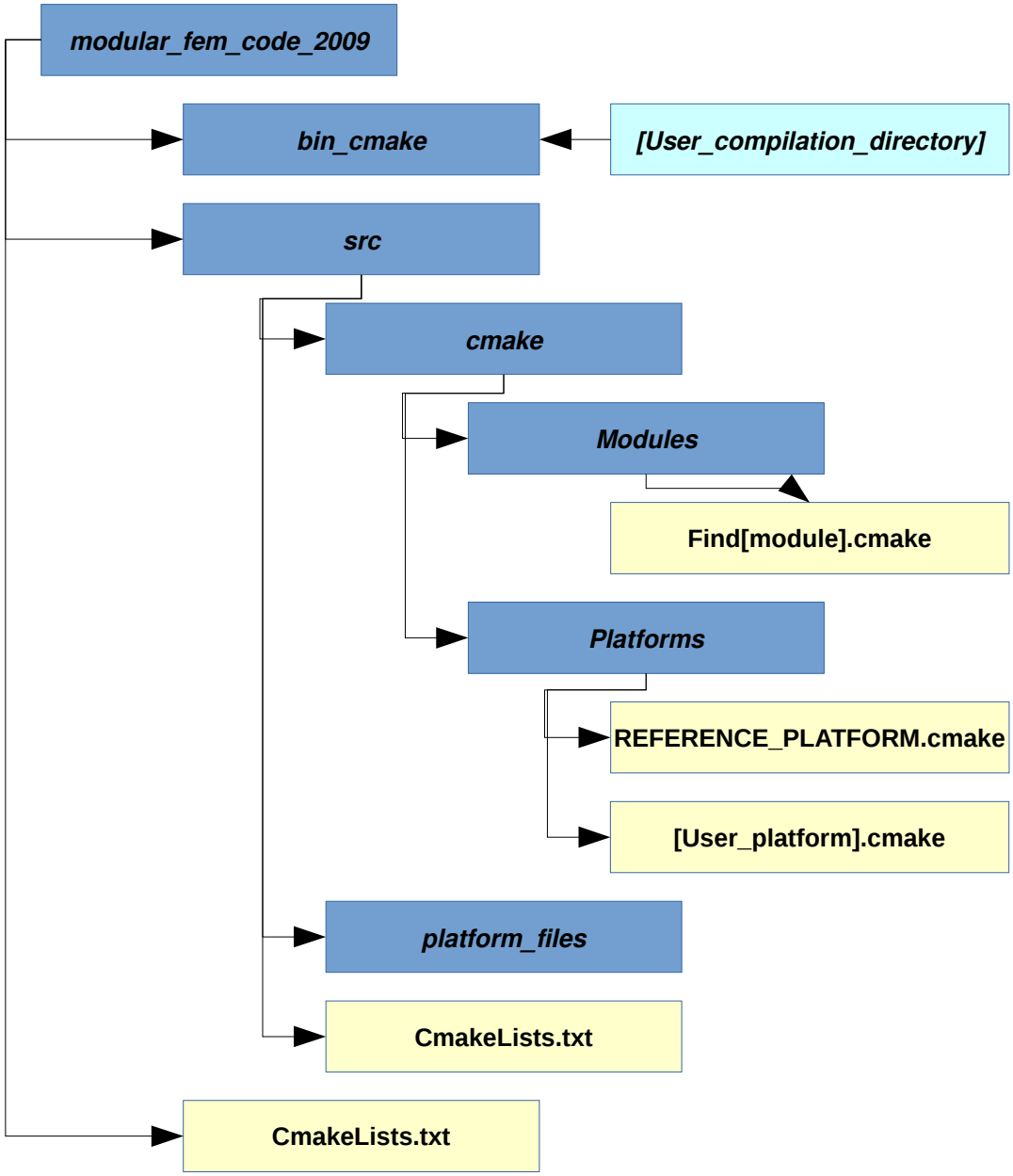
ModFEM build system (CMake)
User's Manual

1) CMAKE WORKING MODES

ModFEM code is built in **two stages**. The first stage is a **preparation level**, during this stage information is collected about platform configuration and the hierarchy of directories for executables files is created. This stage works at '***modular_fem_code_2009***' directory level. Second level is a **compilation stage** where source code is built. This stage works at '***modular_fem_code_2009/src***' directory level. The hierarchy of ModFEM directories related to *cmake* is presented in *Scheme 1*.



Scheme 1: cmake ModFEM hierarchy

1) The preparation stage offers 3 ways to build ModFEM project:
2) **ANCIENT_MAKE MODE** [*DEPRECATED*] – this mode uses 'Makefile_explicite' files and platforms configurations from 'platform_files' directory.

3) **SINGLE_PLATFORM MODE** – this mode uses 'CmakeLists.txt' file and platforms configurations from 'cmake/Platforms" directory. In this mode the code will be compiled for only one selected platform.
4) **MULTI_PLATFORM MODE** – similar to '*SINGLE_PLATFORM MODE*'. In this mode the code will be compiled for all '***modular_fem_code_2009/bin_cmake***' subdirectories linked with correct platforms from 'cmake/Platforms" directory.

2) ModFEM PROJECT COMPILATION

LIST OF OPTIONS

FLAG	VALUES	SUPPORTED MODES	DESCRIPTION	
-DMF_PROBLEM="problem"	***Problem name***	ANCIENT_MAKE	Problem name from list in 'Makefile_explicite' file	
-DCMAKE_BUILD_TYPE= "buil d type"	**Release** **RelWithDebInfo** **Debug**	SINGLE_PLATFORM MULTI_PLATFORM	Build type.	
			Mode	Postfix
			Release	
			RelWithDebInfo	**_i**
			Debug	**_d**
-DMF_CC="compiler"	***C compiler***	SINGLE_PLATFORM MULTI_PLATFORM	Used C compiler. Compiler name can contain version postfix: (ex. gcc-5.1.0)	
-DMF_CXX="compiler"	***C++ compiler***	SINGLE_PLATFORM MULTI_PLATFORM	Used C++ compiler. Compiler name can contain version postfix: (ex. g++-5.1.0)	
-DMF_MPI="value"	**mpi** **nompi**	SINGLE_PLATFORM MULTI_PLATFORM	MPI support flag, if mpi enabled then it requires correct mpi compiler such as mpicc / mpic++	
			Mode	Postfix
			mpi	**_mpi**
			nompi	
-DMF_ACCEL="value"	**none** **openmp** **opencl**	SINGLE_PLATFORM MULTI_PLATFORM	Multithreading and GPU acceleration flags.	
			Mode	Postfix
			none – single thread mode	
			openmp – multithread mode with openmp	
			opencl – multithread mode with openmp and gpu acceleration	**_ocl**

			witch opencl
-DTIME_TEST="switch"	**ON** / **OFF**	SINGLE_PLATFORM MULTI_PLATFORM	Enable/disable time test
-DTIME_TEST_2="switch"	**ON** / **OFF**	SINGLE_PLATFORM MULTI_PLATFORM	Enable/disable time test 2
-DTEST_SCALAR="switch"	**ON** / **OFF**	SINGLE_PLATFORM MULTI_PLATFORM	Enable/disable TEST_SCALAR (works with opencl - it has conflict with LAPLACE flag)
-DLAPLACE="switch"	**ON** / **OFF**	SINGLE_PLATFORM MULTI_PLATFORM	Enable/disable LAPLACE (works with opencl - it has conflict with TEST_SCALAR flag)
-DDEBUG_APM="switch"	**ON** / **OFF**	SINGLE_PLATFORM MULTI_PLATFORM	Enable/disable APM debug info flag.
-DDEBUG_LSM="switch"	**ON** / **OFF**	SINGLE_PLATFORM MULTI_PLATFORM	Enable/disable LSM debug info flag.
-DDEBUG_MMM="switch"	**ON** / **OFF**	SINGLE_PLATFORM MULTI_PLATFORM	Enable/disable MMM debug info flag.
-DDEBUG_PCM="switch"	**ON** / **OFF**	SINGLE_PLATFORM MULTI_PLATFORM	Enable/disable PCM debug info flag.
-DDEBUG_SIM="switch"	**ON** / **OFF**	SINGLE_PLATFORM MULTI_PLATFORM	Enable/disable SIM debug info flag.
-DDEBUG_TMM="switch"	**ON** / **OFF**	SINGLE_PLATFORM MULTI_PLATFORM	Enable/disable TMM debug info flag.

SINGLE_PLATFORM MODE

To begin with, unset environmental variables (in case they are e.g. set in shell initialization scripts)
Remark: All environmental variables handling is shown for bash, for other shells adapt to their specific requirements.

```
export -n MOD_FEM_ARCH
export -n MOD_FEM_ARCH_CMAKE
```

1. Go to „***modular_fem_code_2009***” directory
2. If you want to **create your own platform** you should **perform steps a – d**, otherwise go to **step 4**.
   a) Go to „***modular_fem_code_2009/src/cmake/Platforms***”
   b) Create your own platform file based on „***REFERENCE_PLATFORM.cmake***” file
   c) Modify your platform file, read comments in „***REFERENCE_PLATFORM.cmake***” to customize your platform file for your system configuration.
   d) Go back to „***modular_fem_code_2009***” directory.
3. Create system variable „MOD_FEM_ARCH_CMAKE” with your new or one of existing platform names (in ***modular_fem_code_2009/src/cmake/Platforms***).

```
export  MOD_FEM_ARCH_CMAKE="your_platform_name"
```

4. Run **cmake** with correct flags of:
   -DMF_CC="compiler" -DMF_CXX="compiler"
   -DMF_MPI="value"
   -DMF_ACCEL="value"

```
cmake . -DMF_CC="compiler" -DMF_CXX="compiler" -DMF_MPI="value"
-DMF_ACCEL="value"
```

   You can add other flags, from the options list above, that are compatible with this mode. During this process a correct subdirectory will be created.
5. The executable files will be created in a subdirectory of „***modular_fem_code_2009/bin_cmake***" with the name corresponding to your platform. The rule for folder name creation is presented below:

your_platform_name_mpi_flag_acceleration_flag_c_compiler_c++_compiler

7. Remove cmake files from preparation step:

```
make clean-upper
```

8. To recompile the code go to your platform directory "***modular_fem_code_2009/bin_cmake/your_platform_directory***" and run make command.
9. Running *make* is also sufficient after changes to your platform file (e.g. changing the targets)

MULTI_PLATFORM MODE

1. Go to „***modular_fem_code_2009***" directory
2. Remove useless system variables:

```
export -n MOD_FEM_ARCH
export -n MOD_FEM_ARCH_CMAKE
```

3. Run „**cmake .**" to get a list of available platforms
4. If you want to **create your own platform** you should **perform steps a – d**, otherwise go to **step 5**.
   a) Go to „***modular_fem_code_2009/src/cmake/Platforms***"
   b) Create your own platform from „***REFERENCE_PLATFORM.cmake***" file
   c) Modify your platforms file, read comments in „***REFERENCE_PLATFORM.cmake***" to customize your platform file for your system configuration.
   d) Go back to „***modular_fem_code_2009***" directory.
5. Go to „***modular_fem_code_2009/bin_cmake***" directory.
6. Create your platform subdirectories using the rule below:

your_platform_name_mpi_flag_acceleration_flag_c_compiler_c++_compiler

7. Go to „***modular_fem_code_2009***” directory.
8. Run cmake with selected additional flags compatible with this mode

```
cmake .
```

9. Remove cmake files from preparation step:

```
make clean-upper
```

10. For recompilation code go to your platform directory “***modular_fem_code_2009/bin_cmake/your_platform_directory***” and run make command.

ANCIENT_MAKE MODE [*DEPRECATED*]

1. Go to „***modular_fem_code_2009***” directory
2. Set „**MOD_FEM_ARCH**” flag, to get list of arch check „***modular_fem_code_2009/src/platform_files***” directory

```
export -n MOD_FEM_ARCH_CMAKE
export -n MOD_FEM_ARCH="your_platform_name"
```

3. Run **cmake** witch correct problem flag:
   -DMF_PROBLEM="problem"

```
cmake . -DMF_PROBLEM="problem"
```

4. Remove cmake files from preparation step:

```
make clean-upper
```

OTHER COMPILATION METHODS

1. Go to „***modular_fem_code_2009***” directory
2. Run „**cmake .**” to get a list of available platforms
3. If you want **create your own platform** you should **perform steps a – d**, otherwise go to **step 4**.
   a) Go to „***modular_fem_code_2009/src/cmake/Platforms***”
   b) Create your own platform from „***REFERENCE_PLATFORM.cmake***” file
   c) Modify your platforms file, read comments in „***REFERENCE_PLATFORM.cmake***” to customize your platform file for your system configuration.
   d) Go back to „***modular_fem_code_2009***” directory.
4. Go to „***modular_fem_code_2009/bin_cmake***” directory.
5. Create your platform subdirectories using the rule below:

your_platform_name_mpi_flag_acceleration_flag_c_compiler_c++_compiler

6. Go to create directory.
7. Run cmake from platform directory:

cmake ../../src

8. Run make from platform directory:

make

3) PLATFORM CONFIGURATION „*REFERENCE_PLATFORM.cmake*”

The platforms file stores a system configuration and a list of modules. The settings list is submitted below (the advanced settings are described inside the *REFERENCE PLATFORM.cmake* file in comments):

- Additional compilation flags
- Program linking configuration (**static** / **dynamic**)
- Selection of MPI module (*enable* ***new mpi module***)
- Selection of solver (***direct*** *and* ***iterative***)
- Selection of linear algebra libraries (***MKL / LAPACK***)
- Libraries ***libconfig / boost / MKL / LAPACK/ Voro++ / ViennaCL / PETSc / GLUT / wxWidgets***
- OpenCL configuration (**CPU/GPU/PHI/HSA** *and settings for it*)
- Selection of ModFEM modules

4) List of external libraries used by ModFEM

FREE

- **BLAS Basic Linear Algebra Subprograms (**http://www.netlib.org/blas/**)**
- **Boost C++ Libraries (**http://www.boost.org/**)**
- **GLUT - The OpenGL Utility Toolkit (**https://www.opengl.org/resources/libraries/glut/**),** **(**http://freeglut.sourceforge.net/**)**
- **LAPACK – Linear Algebra PACKage (**http://www.netlib.org/lapack/**)**
- **libconfig (**http://www.hyperrealm.com/libconfig/**)**
- **METIS - Serial Graph Partitioning and Fill-reducing Matrix Ordering (**http://glaros.dtc.umn.edu/gkhome/metis/metis/overview**)**
- **ParMETIS - Parallel Graph Partitioning and Fill-reducing Matrix Ordering (**http://glaros.dtc.umn.edu/gkhome/metis/parmetis/overview**)**
- **PETSc - Portable, Extensible Toolkit for Scientific Computation (**http://www.mcs.anl.gov/petsc/**)**
- **wxWidgets (**https://www.wxwidgets.org/**)**
- **ViennaCL (**http://viennacl.sourceforge.net/**)**
- **Voro++ (**http://math.lbl.gov/voro++/**)**

COMMERCIAL

- **Intel Math Kernel Library [Intel MKL] (https://software.intel.com/en-us/intel-mkl)**
- **Intel MKL PARDISO - Parallel Direct Sparse Solver Interface (**https://software.intel.com/en-us/articles/intel-mkl-pardiso**)**