# EMsoft: General User and Development Manual

Program Manual, Release 3.0.3, April 12, 2016

## Table of Contents

# 1 What is EMsoft?

EMsoft is an open source package for the simulation of electron microscopy imaging and diffraction modalities. It consists of a library with core routines for crystallography, symmetry, dynamical scattering, Monte Carlo simulations, and so on, plus a series of programs for the different modalities. The package is based on the older, and now discontinued, CTEMsoft package, for which the version 2.0 source code is still available at *http://www.github.com/marcdegraef/CTEMsoft*. This code base is no longer supported and is replaced by a new source code base, version 3.0, available from *http://www.github.com/marcdegraef/EMsoft*. The two versions are not compatible with each other (with one exception; see section 7). In the current release, only EBSD and ECP programs are available, along with a number of support programs; in a later release, we will make updated versions of the TEM programs available as well.

This manual describes a few aspects of the EMsoft package:

- how to install the software developer toolkit;
- where to get the executables, if you don't want to compile things yourself;
- the different ways of communicating with the programs (input files);
- and a list of programs available in this release (3.0).

At the time of writing of this manual, these programs have been successfully compiled and executed on the Mac OS X platform (Yosemite) and on Linux CentOS 7 using the public domain gfortran compiler. Starting in the next version, 3.0.4, Windows 10 systems will also be supported. Where appropriate, computations are carried out using OpenMP directives, so that multiple cores are used. In addition, several programs make use of a GPU (if available), using *OpenCL* code.

If you wish to compile EMsoft yourself, then please proceed with the next section; if you only plan on using the executables, please jump ahead to section 4.

# 2 The EMsoft Software Developer Kit

If you wish to compile the package code yourself, then you will first need to pull the code base from github; you can clone the repository from *http://www.github.com/marcdegraef/EMsoft*. Select a location on your hard drive to host the source code tree; we recommend that the foldername be EMsoft, but you are obviously free to change this.

Before you can start to compile the code, you must install the software developer kit (SDK). This is made very easy thanks to our friends at BlueQuartz.net; open a terminal window, navigate inside the EMsoft tree to the folder Support/SDK_Build_Scripts, and then into the correct subfolder (for Mac OS X or Linux). On the Mac OS X and Linux platforms, the SDK will be installed by default in a new /opt/EMsoft_SDK folder; this can be changed by editing the SDK_PARENT and SDK_INSTALL variables in the SDK_Configuration.conf file. Depending on the number of cores on your system, you can also set the PARALLEL_BUILD parameter; if you have $N$ cores, you can in theory put this variable at $2N$, in which case the compilation process will pretty much use all the available compute power on your system. Smaller values will mean a somewhat longer compile time, but your system will remain reasonably responsive.

The build script should be executed with

```
sudo ./Build_SDK.sh
```

This shell script will first download the EMsoft_SDK archive from one of the BlueQuartz web servers and install and unpack the archive. Then the script will install the following program and libraries:

- CMake: CMake is a package that manages the entire compilation process. To make sure that the correct version is used, it is installed first.
- jsonfortran: a small library to parse and create json-formatted files (JavaScript Object Notation);
- HDF5: a large library for Hierarchical Data Format support; note that this library will be compiled twice, once in debug mode and once in release mode. It is normal for there to be a large number of warnings during the compilation; just ignore them.
- clfortran: another small library with wrapper routines to call OpenCL from within f90 programs. Note that this represents a change with respect to older versions, which used the fortrancl library. If you have an existing SDK with fortrancl, you should rebuild the entire SDK by removing the folder and starting a fresh build.
- FFTW: used for fast Fourier transforms; not actually used in release 3.0.3, but will be used in later releases.

All five of these packages should compile and install without any issues, assuming that you have the correct compilers on your system. Note that the installation is sandboxed, i.e., it will not interfere with or overwrite any existing implementations of these libraries; you should only use the libraries inside the sandbox, though. Make sure that your PATH variable lists the SDK before any other collections of libraries that may already be on your system; in particular, make sure that the CMake bin folder is in your path. There is no guarantee that the compilation process will work with other versions of CMake. The compilation process should only take about five minutes. Once the SDK has been compiled, you will be ready to compile EMsoft, as described in the next section.

# 3 Compiling EMsoft

Add the location of the bin folder inside the SDK to your path variable; this will depend on your operating system and the type of shell you are using. For the bash shell on Mac OS X, the command is as follows (assuming the default SDK location):

```
export PATH=$PATH:/opt/EMsoft_SDK/cmake-3.4.3-Darwin-x86_64/CMake.app/Contents/bin/
```

For bash on Linux, use

```
export PATH=$PATH:/opt/EMsoft_SDK/cmake-3.4.3-Linux-x86_64/bin/
```

If you installed the SDK elsewhere, please adjust the actual paths above. Typically, you would add this line, or a similar one for other shells, to your .bashrc or .cshrc file.

In a terminal window, cd to your EMsoft top folder, create a Build folder, cd inside it and enter the following command (if you changed the location of the SDK, make sure you select the correct path instead of /opt):

```
cmake -DEMsoft_SDK=/opt/EMsoft_SDK -DCMAKE_BUILD_TYPE=Debug ../
```

This will spit out a number of variables and path names.[1] This is the first cmake run, which will set up all necessary variables for future runs. The next time you call the cmake program, you can omit the arguments (except for ../); you can also use the interactive ccmake version. To compile the EMsoft package, simply type

```
make -j
```

and the entire library and all the executables will be compiled. From here on, you can start editing the source code and recompile things as needed. All executables will be located in the Build/Bin folder, so you may want to add this folder to your path variable.

# 4 Installation Instructions

If you chose not to compile the code, and you only wish to execute the programs, or you did perform the steps in the last two sections, you will now need to set up your environment so that the executables will be found. The latest package file with executables can be downloaded from the following location:

```
http://muri.materials.cmu.edu/?p=858
```

In the older version of CTEMsoft, the programs required several environment variables to be set, so that all the resource files and such could be located. In the present release, this has been changed to a more flexible system that should work on all platforms (OS X, Linux and Windows). There are only a few variables that need to be set in order for the programs to function properly. They are defined via the EMsoftConfig.json file, located inside the hidden .config/EMsoft folder in the user's home directory. A default version of this file can be generated automatically by executing the EMsoftinit program as the first program that you execute.

1. Place the EMsoft.zip archive somewhere inside your home folder; for instance, if your username is "me," you could create a folder named packages inside the /Users/me folder, and download the archive inside /Users/me/packages. Then unpack the archive, which will create an EMsoft folder containing several other folders.

2. [CHANGED IN RELEASE 3.0.3] Run the EMsoftinit program to create the configuration file inside the [home].config/EMsoft folder. Use a text editor to open the EMsoftConfig.json file and verify that the information is correct:

```
{
        "EMsoftpathname" : "/full/path/to/the/top/EMsoft/installation/folder/",
        "EMdatapathname" : "/full/path/to/the/EMsoft/data/folder/",
        "EMsoftLibraryLocation" : "/full/path/to/the/library/folder/",
        "Release" : "No",
        "UserName" : "Your full name",
        "UserLocation" : "Your location or affiliation",
        "UserEmail" : "Your email address"
}
```

[1] If you get any error messages or notice things that do not look right, please contact us with a screen dump as well as a description of your system, including the OS version.

Note that the white space at the start of lines 2 through 8 is a tab. Typically, the installation folder would be somewhere below your home folder, so on Mac OS X for instance, this could be /Users/me/packages/EMsoft/; note that the path must be terminated with a /. The second variable indicates where you will keep your data files, i.e., all the files generated by the EMsoft programs. This folder must be outside of the main EMsoft-tree, so that no data will be overwritten when future updates or releases become available. Create a top data folder, for instance /Users/me/Data/EMplay/. Then, inside this folder, create an additional folder with the name XtalFolder; this is where all the crystal structure descriptor files will be kept. Whenever the user specifies the location of a data file, this location is always a relative pathname with respect to the EMdatapathname location; so, if we have EMdatapathname set to /Users/me/Data/EMplay/, and we wish to create a data file with absolute pathname /Users/me/Data/EMplay/Silicon/EBSDmaster.h5, then the relative pathname will be Silicon/EBSDmaster.h5. This makes it possible to exchange data files between users or transfer files from one system to another. The variable Release must be set to Yes if you are using the Release version of this package (in other words, if you installed the executables). If instead you have installed the Software Developer Kit, and you are editing source code and compiling things yourself, then you should set this variable to No. Note that this variable, along with the EMsoftLibraryLocation variable, are only used by the IDL visualization interface. Finally, set the variables that start with User to the correct values; these variables will appear in all of the HDF5 files created by the EMsoft programs (added in Release 3.0.1).

3. Adjust the PATH variable so that your shell will find the executables in the EMsoft/bin folder. You can either do this via your shell setup files (e.g., .cshrc or .bashrc), or via the system wide /etc/paths file (which will require sudo access). The syntax will depend a little on which shell you are using (for UNIX-based systems). Note that the UNIX flavor on Mac OS X is not case-sensitive for file and folder names; on all other UNIX-flavored platforms, filenames and folders *are* case-sensitive.

4. To view the HDF5 formatted data files, you may wish to install the free HDF5 viewer. First you need to have Java installed (if it isn't already): for Mac OS X, go to

   `https://java.com/en/download/faq/java_mac.xml`

   to install Java. Then download the HDF5 viewer app from

   `http://www.hdfgroup.org/products/java/release/download.html;`

   scroll down to Max OS X. The HDF5 files can also be read by any other program that supports this format, e.g., IDL, Matlab, etc...

# 5 The general structure of EMsoft programs

Each EMsoft program has the same high level source code structure, and is called in the same way. If EMprogram represents a program name (for instance EMECP or EMlacbed, then the following command line options can be used (all of them optional):

```
EMprogram [-h] [-t] [file.nml]
```

The arguments between square brackets are optional, and are defined as follows:

- -h: (h=help) this argument causes the program to display a list of all command line arguments and their meaning. The program will quit after printing the help message.
- -t: (t=template) this argument causes the program to create template files for all the input files used by the program. For each name-value entry in the template file (see section 6 for more details), a comment line is inserted with a brief explanation of the variable and its units, if appropriate. The user can then copy the template file to a new file with the namelist (.nml) extension and edit the file with a regular text editor. The program will quit after generating the template files. Each template file contains the default values for each parameter. If this option produces an error, then this is likely due to an incorrect setting of the shell variables in the previous section.
- file.nml: the main namelist file to be used by the program. If no name is present, then the default filename EMprogram.nml will be used. If the provided namelist file does not exist, the program will report an error and abort.

Note that the programs will also accept .json files instead of .nml files.

The programs in the top half of the table on the last page of this manual function in the way described above; all other programs have a simple command line interface, with user prompts for all relevant variables. Eventually, in a future release, all programs will make use of namelist or json input files and will likely be callable from a single user interface.

# 6 Program input files

All user input for the EMsoft programs is performed either via namelist files or via json files; the namelist is a special f90 IO format that allows one to list *name-value* pairs in an ordinary text file, and then this file will be read and interpreted by the program. The basic format of a namelist file is as follows:[2]

```
&nmlname
 var1 = val1,
 var2 = val2,
 ...
/
```

The first line must begin with the ampersand character "&" and is followed (without spaces) by the internal name used by the program to identify the set of variables (this is similar to a *common block* in the older f77 standard). After the mandatory first line, which should not be altered, a list of name-value pairs follows. These can appear in any order, and variables may be omitted from the file, in which case the program will use a default value (this value can be found in the template file; see section 5). The final line of the file must contain a single forward slash character, "/", to end the namelist.

The json format (JavaScript Object Notation) is an xml-like version of the name-value pairs, and is very similar to the namelist format, with the exception of the presence of curly braces. For the example namelist file shown above, the equivalent json file would look as follows:

```
{
        "nmlname": {
                 "var1": val1,
                 "var2": val2,
                ...
        }
}
```

The EMsoft programs automatically detect which format of input file is used, so either one is fine. The json formatted input files will in a future release be generated by a number of filters in the DREAM.3D software package, whereas the namelist formatted files must be edited by hand, starting from the template files (created with option -t).

As a concrete example, consider the namelist file used to define a dislocation:

```
&dislocationdata
 id = 0.501,
 jd = 0.501,
 u = 1.0,0.0,1.0,
 bv = 0.5,0.0,0.5
/
```

[2] Note that the .nml format has been changed slightly from the one in the previous release. The main reason is compatibility with the ifort compiler on Windows, which expects each line in the .nml file to start with a space and end with a comma. On non-Windows platforms, the older .nml format should continue to work.

Internally, the program identifies this set of variables with the namelist identifier "dislocationdata," as shown on the first line of the file. Then there are four name-value pairs: id and jd are real numbers in the range $[-1, 1]$ and define the position of the defect; u represents the line direction, which is declared as a triplet of floating point numbers; and the burgers vector is represented by another triplet of floating point numbers and the variable name bv. The forward slash closes the file. Note that these entries can appear in any order. Comment lines are allowed in the namelist file; simply start the line with the f90 comment character (the exclamation mark "!") and then type your comment. For variables that have components, such as the two vectors above, one may also define each component separately, in which case the file would look as follows:

```
&dislocationdata
 id = 0.501,
 jd = 0.501,
! these vectors are defined one component at a time
 u(1) = 1.0,
 u(2) = 0.0,
 u(3) = 1.0,
 bv(1) = 0.5,
 bv(2) = 0.0,
 bv(3) = 0.5
/
```

It is not necessary to leave spaces before and after the "=" symbol, but it does improve the readability of the namelist file to do so. Since one can always recover the template file(s) by using the [-t] option when calling the program, there is no need to list all the name-value pairs in the namelist file; pairs which are not listed will take on their default value(s), which are the ones listed in the template file.

One important thing to note: when entering file names and other strings in name list files, one must use single quotes both at the start and at the end of the string. Some word processing programs have so-called "smart quotes" settings, which will introduce a different kind of quote, and the fortran programs will not recognize smart quotes as being the same as single quotes. Therefore, turn off the "smart quotes" option in your editor before you start editing namelist or json files.

# 7 Generating a crystal structure file

With very few exceptions, all programs in the EMsoft suite of programs require a crystallographic input file that defines the crystal system, crystal lattice parameters, space group number, and all unique positions in the asymmetric unit along with site occupations and Debye-Waller parameters. To keep things simple and portable across platforms, structure description files (*.xtal) are small HDF5-formatted files that contain the minimum amount of information needed to unambiguously define the complete unit cell.

To create a crystal structure description file, one uses the EMmkxtal program, which will prompt the user for the crystal system number, the lattice parameters (in nm and degrees), the space group number, and the atom positions in the asymmetric unit. Each atom entry is defined by five numbers: the fractional coordinates (which can be entered as real numbers or as fraction, e.g., 0.5 or 1/2), the site occupation parameter (a real number between 0 and 1), and the isotropic Debye-Waller factor in $nm^2$.

Once all atoms in the asymmetric unit have been entered, the program will ask for a structure file name; we recommend that all such files have the extension .xtal, to make them easily recognizable. All .xtal files will be automatically placed in the XtalFolder inside the main EMsoft data folder.

In the next major release, 3.1, we will make available a DREAM.3D filter that can generate the *.xtal file interactively.

# 8 List of programs

The list below shows all the available programs in the current Release (3.0) of the EMsoft suite, along with whether or not there is a separate manual for each program. Programs for which no manual is available are generally relatively easy to figure out. In the current release, many programs still have a command-line interaction with the user; this will be modified in the next release so that all programs will eventually make use of namelist files. The top half of the table lists those programs for which an extensive manual is available, as well as IDL visualization routines. The EMmkxtal program is explained in the present document. The other programs do not yet have manual pages, but are generally simple to figure out. Many of the programs in the bottom half of the table are simple illustrations of how to perform certain computations.

| Program | Short description | Manual? |
|---|---|---|
| EMMCOpenCL | Monte Carlo simulation (SEM) | ✓ |
| EMEBSDmaster | EBSD master pattern simulation (SEM) | ✓ |
| EMEBSD | EBSD pattern simulation (SEM) | ✓ |
| EMECPmaster | electron channeling master pattern (SEM) | ✓ |
| EMECP | electron channeling pattern (SEM) | ✓ |
| EMKosselMaster | electron Kossel master patterns (SEM) | ✓ |
| EMmkxtal | make a crystal structure input file | ✓ |
| EMOpenCLinfo | list the available compute devices on your computer | |
| EMlistSG | list the equivalent positions of any space group | |
| EMqg | list Fourier coefficient information | |
| EMfamily | draw a family of planes (stereographic) | |
| EMstar | list the star of any reciprocal lattice point | |
| EMorbit | list the orbit of a point | |
| EMZAgeom | print zone axis geometry and symmetry information | |
| EMlatgeom | perform some simple lattice geometry calculations | |
| EMstereo | basic stereographic prpjection tool | |
| EMorient | stereographic projection for orientation relation | |
| EMxtalinfo | makes postscript file with lots of information | |
| EMzap | create postscript file with zone axis patterns | |
| EMdrawcell | draw a unit cell (very primitive[3]) | |

# 9 Release Updates

3.0.3 (04/13/16)

- added option to turn off requirement for LaTeX installation to be present in order to compile package;
- modified variable definitions in MBmodule and EMdymod to eliminate compilation errors when using the Intel fortran compiler in Visual Studio [S. Wright];
- updated HDFsupport module; converted HDF read operations to subroutines instead of functions to please the Intel fortran compiler;
- added DLLexport comment lines to all module files;
- changed OpenCL support from the fortrancl library to the more complete clfortran library;
- added EMOpenCLinfo program to display the computing environment;
- added CMake support for compilation on Linux (tested on CentOS 7);
- automatic generation of configuration file upon first execution of any EMsoft program;
- removed dispfield.f90, family.f90, and hdftest.f90 from src folder;
- started preparation of source code for Windows 10 support in the next version.

3.0.2 (12/11/15)

- added EMsoftLibraryLocation to EMsoftConfig.json file;
- corrected several of the example .nml files;
- added EMZAgeom program to bin folder; was missing in 3.0 and 3.0.1;

- moved VMapps folder into IDL folder and added symbolic link so that the virtual apps can find the EMsoft libraries.

3.0.1 (11/20/15) added ability to set UserName, UserLocation, and UserEmail in EMsoftConfig.json file.

# 10 Future releases

The following future releases are currently planned:

- 3.1, Mid Summer 2016: addition of dictionary indexing codes for EBSD and ECP; ECCI defect imaging code; integration with DREAM.3D filters.
- 3.2, Winter 2016: addition of updated TEM codes (currently still available in release 2.0); further integration with DREAM.3D.