FindMatlab

Finds Matlab or Matlab Compiler Runtime (MCR) and provides Matlab tools, libraries and compilers to CMake.

This package primary purpose is to find the libraries associated with Matlab or the MCR in order to be able to build Matlab extensions (mex files). It can also be used:

- to run specific commands in Matlab in case Matlab is available
- for declaring Matlab unit test
- to retrieve various information from Matlab (mex extensions, versions and release queries, …)

The module supports the following components:

- `ENG_LIBRARY` and `MAT_LIBRARY`: respectively the `ENG` and `MAT` libraries of Matlab
- `MAIN_PROGRAM` the Matlab binary program. Note that this component is not available on the MCR version, and will yield an error if the MCR is found instead of the regular Matlab installation.
- `MEX_COMPILER` the MEX compiler.
- `MCC_COMPILER` the MCC compiler, included with the Matlab Compiler add-on.
- `SIMULINK` the Simulink environment.

Note: The version given to the **`find_package()`** directive is the Matlab **version**, which should not be confused with the Matlab *release* name (eg. *R2014*). The **`matlab_get_version_from_release_name()`** and **`matlab_get_release_name_from_version()`** provide a mapping between the release name and the version.

The variable **`Matlab_ROOT_DIR`** may be specified in order to give the path of the desired Matlab version. Otherwise, the behaviour is platform specific:

- Windows: The installed versions of Matlab/MCR are retrieved from the Windows registry
- OS X: The installed versions of Matlab/MCR are given by the MATLAB default installation paths in `/Application`. If no such application is found, it falls back to the one that might be accessible from the `PATH`.
- Unix: The desired Matlab should be accessible from the `PATH`. This does not work for MCR installation and **`Matlab_ROOT_DIR`** should be specified on this platform.

Additional information is provided when **`MATLAB_FIND_DEBUG`** is set. When a Matlab/MCR installation is found automatically and the `MATLAB_VERSION` is not given, the version is queried from Matlab directly (on Windows this may pop up a Matlab window) or from the MCR installation.

The mapping of the release names and the version of Matlab is performed by defining pairs (name, version). The variable **`MATLAB_ADDITIONAL_VERSIONS`** may be provided before the call to the **`find_package()`** in order to handle additional versions.

A Matlab scripts can be added to the set of tests using the **`matlab_add_unit_test()`**. By default, the Matlab unit test framework will be used (>= 2013a) to run this script, but regular `.m` files returning an exit code can be used as well (0 indicating a success).

Module Input Variables

Users or projects may set the following variables to configure the module behaviour:

`Matlab_ROOT_DIR`
the root of the Matlab installation.

`MATLAB_FIND_DEBUG`
outputs debug information

`MATLAB_ADDITIONAL_VERSIONS`
additional versions of Matlab for the automatic retrieval of the installed versions.

Variables defined by the module

Result variables

`Matlab_FOUND`
`TRUE` if the Matlab installation is found, `FALSE` otherwise. All variable below are defined if Matlab is found.

`Matlab_ROOT_DIR`
the final root of the Matlab installation determined by the FindMatlab module.

`Matlab_MAIN_PROGRAM`

the Matlab binary program. Available only if the component `MAIN_PROGRAM` is given in the **`find_package()`** directive.

`Matlab_INCLUDE_DIRS`
the path of the Matlab libraries headers

`Matlab_MEX_LIBRARY`
library for mex, always available.

`Matlab_MX_LIBRARY`
mx library of Matlab (arrays), always available.

`Matlab_ENG_LIBRARY`
Matlab engine library. Available only if the component `ENG_LIBRARY` is requested.

`Matlab_MAT_LIBRARY`
Matlab matrix library. Available only if the component `MAT_LIBRARY` is requested.

`Matlab_ENGINE_LIBRARY`
Matlab C++ engine library, always available for R2018a and newer.

`Matlab_DATAARRAY_LIBRARY`
Matlab C++ data array library, always available for R2018a and newer.

`Matlab_LIBRARIES`
the whole set of libraries of Matlab

`Matlab_MEX_COMPILER`
the mex compiler of Matlab. Currently not used. Available only if the component `MEX_COMPILER` is requested.

`Matlab_MCC_COMPILER`
the mcc compiler of Matlab. Included with the Matlab Compiler add-on. Available only if the component `MCC_COMPILER` is requested.

Cached variables

`Matlab_MEX_EXTENSION`
the extension of the mex files for the current platform (given by Matlab).

`Matlab_ROOT_DIR`
the location of the root of the Matlab installation found. If this value is changed by the user, the result variables are recomputed.

Provided macros

`matlab_get_version_from_release_name()`
returns the version from the release name

`matlab_get_release_name_from_version()`
returns the release name from the Matlab version

Provided functions

`matlab_add_mex()`
adds a target compiling a MEX file.

`matlab_add_unit_test()`
adds a Matlab unit test file as a test to the project.

`matlab_extract_all_installed_versions_from_registry()`
parses the registry for all Matlab versions. Available on Windows only. The part of the registry parsed is dependent on the host processor

`matlab_get_all_valid_matlab_roots_from_registry()`
returns all the possible Matlab or MCR paths, according to a previously given list. Only the existing/accessible paths are kept. This is mainly useful for the searching all possible Matlab installation.

`matlab_get_mex_suffix()`
returns the suffix to be used for the mex files (platform/architecture dependent)

`matlab_get_version_from_matlab_run()`
returns the version of Matlab/MCR, given the full directory of the Matlab/MCR installation path.

Known issues

Symbol clash in a MEX target
By default, every symbols inside a MEX file defined with the command **`matlab_add_mex()`** have hidden visibility, except for the entry point. This is the default behaviour of the MEX compiler, which lowers the risk of

symbol collision between the libraries shipped with Matlab, and the libraries to which the MEX file is linking to. This is also the default on Windows platforms.

However, this is not sufficient in certain case, where for instance your MEX file is linking against libraries that are already loaded by Matlab, even if those libraries have different SONAMES. A possible solution is to hide the symbols of the libraries to which the MEX target is linking to. This can be achieved in GNU GCC compilers with the linker option `-Wl,--exclude-libs,ALL`.

Tests using GPU resources

in case your MEX file is using the GPU and in order to be able to run unit tests on this MEX file, the GPU resources should be properly released by Matlab. A possible solution is to make Matlab aware of the use of the GPU resources in the session, which can be performed by a command such as `D = gpuDevice()` at the beginning of the test script (or via a fixture).

Reference

`Matlab_ROOT_DIR`

The root folder of the Matlab installation. If set before the call to **`find_package()`**, the module will look for the components in that path. If not set, then an automatic search of Matlab will be performed. If set, it should point to a valid version of Matlab.

`MATLAB_FIND_DEBUG`

If set, the lookup of Matlab and the intermediate configuration steps are outputted to the console.

`MATLAB_ADDITIONAL_VERSIONS`

If set, specifies additional versions of Matlab that may be looked for. The variable should be a list of strings, organised by pairs of release name and versions, such as follows:

```
set(MATLAB_ADDITIONAL_VERSIONS
    "release_name1=corresponding_version1"
    "release_name2=corresponding_version2"
    ...
    )
```

Example:

```
set(MATLAB_ADDITIONAL_VERSIONS
    "R2013b=8.2"
    "R2013a=8.1"
    "R2012b=8.0")
```

The order of entries in this list matters when several versions of Matlab are installed. The priority is set according to the ordering in this list.

`matlab_get_version_from_release_name`

Returns the version of Matlab (17.58) from a release name (R2017k)

`matlab_get_release_name_from_version`

Returns the release name (R2017k) from the version of Matlab (17.58)

`matlab_extract_all_installed_versions_from_registry`

This function parses the registry and founds the Matlab versions that are installed. The found versions are returned in *matlab_versions*. Set *win64* to *TRUE* if the 64 bit version of Matlab should be looked for The returned list contains all versions under `HKLM\\SOFTWARE\\Mathworks\\MATLAB` and `HKLM\\SOFTWARE\\Mathworks\\MATLAB Runtime` or an empty list in case an error occurred (or nothing found).

> **Note:** Only the versions are provided. No check is made over the existence of the installation referenced in the registry,

`matlab_get_all_valid_matlab_roots_from_registry`

Populates the Matlab root with valid versions of Matlab or Matlab Runtime (MCR). The returned matlab_roots is organized in triplets `(type,version_number,matlab_root_path)`, where `type` indicates either `MATLAB` or `MCR`.

```
matlab_get_all_valid_matlab_roots_from_registry(
    matlab_versions
    matlab_roots)
```

`matlab_versions`

the versions of each of the Matlab or MCR installations

`matlab_roots`

the location of each of the Matlab or MCR installations

matlab_get_mex_suffix

Returns the extension of the mex files (the suffixes). This function should not be called before the appropriate Matlab root has been found.

```
matlab_get_mex_suffix(
    matlab_root
    mex_suffix)
```

`matlab_root`

the root of the Matlab/MCR installation

`mex_suffix`

the variable name in which the suffix will be returned.

matlab_get_version_from_matlab_run

This function runs Matlab program specified on arguments and extracts its version. If the path provided for the Matlab installation points to an MCR installation, the version is extracted from the installed files.

```
matlab_get_version_from_matlab_run(
    matlab_binary_path
    matlab_list_versions)
```

`matlab_binary_path`

the location of the *matlab* binary executable

`matlab_list_versions`

the version extracted from Matlab

matlab_add_unit_test

Adds a Matlab unit test to the test set of cmake/ctest. This command requires the component `MAIN_PROGRAM` and hence is not available for an MCR installation.

The unit test uses the Matlab unittest framework (default, available starting Matlab 2013b+) except if the option `NO_UNITTEST_FRAMEWORK` is given.

The function expects one Matlab test script file to be given. In the case `NO_UNITTEST_FRAMEWORK` is given, the unittest script file should contain the script to be run, plus an exit command with the exit value. This exit value will be passed to the ctest framework (0 success, non 0 failure). Additional arguments accepted by **add_test()** can be passed through `TEST_ARGS` (eg. `CONFIGURATION <config> ...`).

```
matlab_add_unit_test(
    NAME <name>
    UNITTEST_FILE matlab_file_containing_unittest.m
    [CUSTOM_TEST_COMMAND matlab_command_to_run_as_test]
    [UNITTEST_PRECOMMAND matlab_command_to_run]
    [TIMEOUT timeout]
    [ADDITIONAL_PATH path1 [path2 ...]]
    [MATLAB_ADDITIONAL_STARTUP_OPTIONS option1 [option2 ...]]
    [TEST_ARGS arg1 [arg2 ...]]
    [NO_UNITTEST_FRAMEWORK]
    )
```

The function arguments are:

`NAME`

name of the unittest in ctest.

`UNITTEST_FILE`

the matlab unittest file. Its path will be automatically added to the Matlab path.

`CUSTOM_TEST_COMMAND`

Matlab script command to run as the test. If this is not set, then the following is run: `runtests('matlab_file_name'), exit(max([ans(1,:).Failed]))` where `matlab_file_name` is the `UNITTEST_FILE` without the extension.

`UNITTEST_PRECOMMAND`

Matlab script command to be ran before the file containing the test (eg. GPU device initialisation based on CMake variables).

`TIMEOUT`

the test timeout in seconds. Defaults to 180 seconds as the Matlab unit test may hang.

`ADDITIONAL_PATH`

a list of paths to add to the Matlab path prior to running the unit test.

`MATLAB_ADDITIONAL_STARTUP_OPTIONS`

a list of additional option in order to run Matlab from the command line. `-nosplash -nodesktop -nodisplay` are always added.

`TEST_ARGS`

Additional options provided to the add_test command. These options are added to the default options (eg. “CONFIGURATIONS Release”)

`NO_UNITTEST_FRAMEWORK`

when set, indicates that the test should not use the unittest framework of Matlab (available for versions >= R2013a).

`WORKING_DIRECTORY`

This will be the working directory for the test. If specified it will also be the output directory used for the log file of the test run. If not specified the temporary directory `${CMAKE_BINARY_DIR}/Matlab` will be used as the working directory and the log location.

matlab_add_mex

Adds a Matlab MEX target. This commands compiles the given sources with the current tool-chain in order to produce a MEX file. The final name of the produced output may be specified, as well as additional link libraries, and a documentation entry for the MEX file. Remaining arguments of the call are passed to the **`add_library()`** or **`add_executable()`** command.

```
matlab_add_mex(
    NAME <name>
    [EXECUTABLE | MODULE | SHARED]
    SRC src1 [src2 ...]
    [OUTPUT_NAME output_name]
    [DOCUMENTATION file.txt]
    [LINK_TO target1 target2 ...]
    [R2017b | R2018a]
    [...]
)
```

`NAME`

name of the target.

`SRC`

list of source files.

`LINK_TO`

a list of additional link dependencies. The target links to `libmex` and `libmx` by default.

`OUTPUT_NAME`

if given, overrides the default name. The default name is the name of the target without any prefix and with `Matlab_MEX_EXTENSION` suffix.

`DOCUMENTATION`

if given, the file `file.txt` will be considered as being the documentation file for the MEX file. This file is copied into the same folder without any processing, with the same name as the final mex file, and with extension *.m*. In that case, typing `help <name>` in Matlab prints the documentation contained in this file.

`R2017b` or `R2018a` may be given to specify the version of the C API

to use: `R2017b` specifies the traditional (separate complex) C API, and corresponds to the `-R2017b` flag for the *mex* command. `R2018a` specifies the new interleaved complex C API, and corresponds to the `-R2018a` flag for the *mex* command. Ignored if MATLAB version prior to R2018a. Defaults to `R2017b`.

`MODULE` or `SHARED` may be given to specify the type of library to be

created. `EXECUTABLE` may be given to create an executable instead of a library. If no type is given explicitly, the type is `SHARED`.

The documentation file is not processed and should be in the following format:

```
% This is the documentation
function ret = mex_target_output_name(input1)
```