Generic Makefile

v1.0.0

Generated by Doxygen 1.8.8

Mon Oct 1 2018 14:27:24

Contents

Chapter 1

Generic Makefile

1.1 Introduction

This Makefile compiles the C / C ++ sources of a project that respects a specific folders architecture. It adapts according to the binary to be generated through the different variables available.

1.2 Folder architecture

This architecture must match the one shown below. Only the directories used by this makefile are indicated in this diagram.

```
Workspace

     Project 1
                 build

                           Debug_<arch>

                           Release_<arch>

                           Test_<arch>

                 include

                 log

                 package

                             deb

                             rpm

                             tar

                             ...

                             zip


                 src

                 test
```

```
 Project 2
:
:

 Project n
```

1.2.1 Directories contents

- Workspace ($WORKSPACE_DIR) : Root of workspace containing all associated projects.
- Project x ($PROJECT_DIR) : Roots of projects.
- build ($BIN_DIR) : Root directory containing the generated binaries. These binaries are sorted by build configuration and by hardware architecture type.
- build/$CONFIG_$TARGET_ARCH ($OBJ_DIR) : Binary specific to the build configuration.
- include ($INC_DIR) : Directory for public header files to be deployed with the project binary (usually for a library project)
- log ($LOG_DIR) : Directory intended to receive the files generated by the build and control tools.
- package ($PACKAGE_DIR) : Root directory of subdirectories intended to receive the delivery packages.
- src ($SRC_DIR) : Root directory of the source files. This directory can contain sub-directories.
- test ($TEST_DIR) : Root directory of the unit testing source files.

Chapter 2

Module Index

2.1 Modules

Here is a list of all modules:

Chapter 3

File Index

3.1 File List

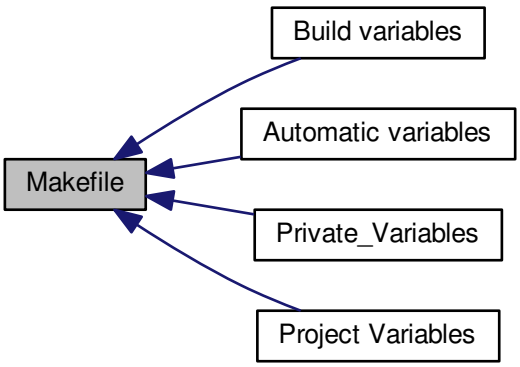
Here is a list of all documented files with brief descriptions:

Chapter 4

Module Documentation

4.1 Makefile

Collaboration diagram for Makefile:



Modules

- Project Variables

  These variables must be declared in this file according to the project.
- Build variables

  These variables can be redefined before the execution of 'make'.
- Private_Variables

  These variables.
- Automatic variables

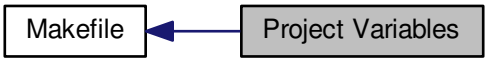
  These variables can be redefined during the execution of 'make'.

4.1.1 Detailed Description

4.2 Project Variables

These variables must be declared in this file according to the project.

Collaboration diagram for Project Variables:



Macros

- #define PROJECT_NAME

  Defines the project name. This name must match the name of the project's parent directory in the workspace.
- #define DEPENDENCIES

  Sets the name of libraries whose binary depends. These libraries must be in a path of the LD_LIBRARY_PATH variable, in the system default directories, or in workspace.
- #define TEST_DEPENDENCIES

  Sets the name of libraries whose unit testing depends. These libraries must be in a path of the LD_LIBRARY_PATH variable, in the system default directories, or in workspace.
- #define LIB_DIR

  Defines directory of system libraries. This variable can be set before calling 'make'.
- #define TEST_LIB_DIR

  Defines the library directory required for unit testing. This variable can be set before calling 'make'.
- #define BINARY_TYPE

  Defines the type of binary generated. Authorized values are 'exe' (by default), 'lib', or 'shared'.

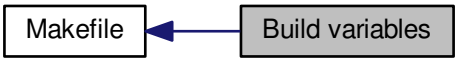
4.2.1 Detailed Description

These variables must be declared in this file according to the project.

4.3 Build variables

These variables can be redefined before the execution of 'make'.

Collaboration diagram for Build variables:



Macros

- #define CONFIG

  Defines build configuration. Authorized values are 'Release' (by default), 'Debug', or 'Test'.
- #define PRE_DEFINED

  Stores predefined variables. These variables are passed to the preprocessor.
- #define TARGET_ARCH

  Defines target architecture. Authorized values are 'x86', 'x86_64' (by default), 'arm', 'armhf'.
- #define OS_TYPE

  Defines target Operating System. Authorized values are 'Linux' (by default), 'Linux64', 'Win32', 'Win64'.
- #define MAJ_VERSION

  Defines the major version of the binary. The default value is '1'.
- #define MIN_VERSION

  Defines the minor version of the binary. The default value is '0'.
- #define BUILD_VERSION

  Defines the build version of the binary. The default value is '0'.

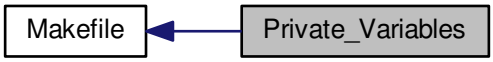
4.3.1 Detailed Description

These variables can be redefined before the execution of 'make'.

4.4 Private_Variables

These variables.

Collaboration diagram for Private_Variables:



Macros

- #define REV_NUMBER_FILE

  Sets the filename containing the revision number. This filename is 'rev-number.txt' by default.
- #define VERS_NUMBER_FILE

  Sets the filename containing the version number. This filename is 'vers-number.txt' by default. It contains the version number in the form x.y.z where x is the major version, y the minor version and z the build version.
- #define TEST_LIB

  Sets the name of the unit test library.

4.4.1 Detailed Description

These variables.

4.5 Automatic variables

These variables can be redefined during the execution of 'make'.

Collaboration diagram for Automatic variables:

Macros

- #define VERSION

  Defines the version in the format x.y.z.
- #define BINARY_PREFIX

  Sets the prefix of the binary file according to the configuration.
- #define BINARY_EXT

  Sets the extension of the binary file according to the configuration.
- #define **BIN_DIR_NAME**
- #define **SRC_DIR_NAME**
- #define **INC_DIR_NAME**
- #define **LOG_DIR_NAME**
- #define **TEST_DIR_NAME**
- #define PROJECT_DIR

  Reads actual directory corresponding to project name.
- #define WORKSPACE_DIR

  Defines Workspace directory.
- #define SRC_DIR

  Defines the parent directory of sources.
- #define SRC_SUBDIR

  Defines the sub-directories of sources.
- #define TEST_DIR

  Defines the parent directory of unit testing sources.
- #define BIN_DIR

  Defines the parent directory of binaries.
- #define INC_DIR

  Defines directory for header files to be deployed with the project binary.
- #define TEST_INC_DIR

  Defines directory for header files required for unit testing.
- #define OBJ_DIR

  Defines directory where the object files will be stored during compilation.
- #define LOG_DIR

  Defines directory where the report files will be stored.
- #define REV_NUMBER

  Contains the revision number. This number is read from REV_NUMBER_FILE.
- #define C_SOURCES

Defines the list of c source files.

- #define CXX_SOURCES

  Defines the list of c++ source files.
- #define C_TEST_SOURCES

  Defines the list of unit tests c source files.
- #define CXX_TEST_SOURCES

  Defines the list of unit tests c++ source files.
- #define SOURCES

  Defines the list of c and c++ source files.
- #define OBJ_LIST

  Defines the list of object files.
- #define OBJECTS

  Defines the list of object files.
- #define DEP_FILES

  Establish the list of dependency files from the list of object files.
- #define LOCAL_ARCH

  Stores host architecture.
- #define ARM_FAMILY

  Stores the ARM family.
- #define CC

  Stores C compilator executable.
- #define CXX

  Stores C++ compilator executable.
- #define AR

  Stores archiver command.
- #define RM

  Stores remove command.
- #define ARM_FLAGS

  Stores specific ARM flags for compilation and linking.
- #define COMMON_FLAGS

  Defines the C and C++ common flags (compilation and linking)
- #define COMPIL_FLAGS

  Defines the C and C++ common compilation flags.
- #define CFLAGS

  Defines the C compilation flags.
- #define CXXFLAGS

  Defines the C++ compilation flags.
- #define LDFLAGS

  Defines the linker flags.

4.5.1 Detailed Description

These variables can be redefined during the execution of 'make'.

4.5.2 Macro Definition Documentation

4.5.2.1 #define ARM_FAMILY

Stores the ARM family.

The main values allowed are : cortex-a8, cortex-a9, ...

Definition at line 462 of file Makefile.

Chapter 5

File Documentation

5.1 Makefile File Reference

Generic project build file.

Macros

- #define COLOR

  Sets the escape command to display messages in color.
- #define END_COLOR

  Sets the escape command to complete the colorization.
- #define PROJECT_NAME

  Defines the project name. This name must match the name of the project's parent directory in the workspace.
- #define DEPENDENCIES

  Sets the name of libraries whose binary depends. These libraries must be in a path of the LD_LIBRARY_PATH variable, in the system default directories, or in workspace.
- #define TEST_DEPENDENCIES

  Sets the name of libraries whose unit testing depends. These libraries must be in a path of the LD_LIBRARY_PATH variable, in the system default directories, or in workspace.
- #define LIB_DIR

  Defines directory of system libraries. This variable can be set before calling 'make'.
- #define TEST_LIB_DIR

  Defines the library directory required for unit testing. This variable can be set before calling 'make'.
- #define BINARY_TYPE

  Defines the type of binary generated. Authorized values are 'exe' (by default), 'lib', or 'shared'.
- #define CONFIG

  Defines build configuration. Authorized values are 'Release' (by default), 'Debug', or 'Test'.
- #define PRE_DEFINED

  Stores predefined variables. These variables are passed to the preprocessor.
- #define TARGET_ARCH

  Defines target architecture. Authorized values are 'x86', 'x86_64' (by default), 'arm', 'armhf'.
- #define OS_TYPE

  Defines target Operating System. Authorized values are 'Linux' (by default), 'Linux64', 'Win32', 'Win64'.
- #define MAJ_VERSION

  Defines the major version of the binary. The default value is '1'.
- #define MIN_VERSION

  Defines the minor version of the binary. The default value is '0'.

- #define BUILD_VERSION

  Defines the build version of the binary. The default value is '0'.
- #define REV_NUMBER_FILE

  Sets the filename containing the revision number. This filename is 'rev-number.txt' by default.
- #define VERS_NUMBER_FILE

  Sets the filename containing the version number. This filename is 'vers-number.txt' by default. It contains the version number in the form x.y.z where x is the major version, y the minor version and z the build version.
- #define TEST_LIB

  Sets the name of the unit test library.
- #define VERSION

  Defines the version in the format x.y.z.
- #define BINARY_PREFIX

  Sets the prefix of the binary file according to the configuration.
- #define BINARY_EXT

  Sets the extension of the binary file according to the configuration.
- #define **BIN_DIR_NAME**
- #define **SRC_DIR_NAME**
- #define **INC_DIR_NAME**
- #define **LOG_DIR_NAME**
- #define **TEST_DIR_NAME**
- #define PROJECT_DIR

  Reads actual directory corresponding to project name.
- #define WORKSPACE_DIR

  Defines Workspace directory.
- #define SRC_DIR

  Defines the parent directory of sources.
- #define SRC_SUBDIR

  Defines the sub-directories of sources.
- #define TEST_DIR

  Defines the parent directory of unit testing sources.
- #define BIN_DIR

  Defines the parent directory of binaries.
- #define INC_DIR

  Defines directory for header files to be deployed with the project binary.
- #define TEST_INC_DIR

  Defines directory for header files required for unit testing.
- #define OBJ_DIR

  Defines directory where the object files will be stored during compilation.
- #define LOG_DIR

  Defines directory where the report files will be stored.
- #define REV_NUMBER

  Contains the revision number. This number is read from REV_NUMBER_FILE.
- #define C_SOURCES

  Defines the list of c source files.
- #define CXX_SOURCES

  Defines the list of c++ source files.
- #define C_TEST_SOURCES

  Defines the list of unit tests c source files.
- #define CXX_TEST_SOURCES

  Defines the list of unit tests c++ source files.
- #define SOURCES

Defines the list of c and c++ source files.

- #define OBJ_LIST

  Defines the list of object files.
- #define OBJECTS

  Defines the list of object files.
- #define DEP_FILES

  Establish the list of dependency files from the list of object files.
- #define LOCAL_ARCH

  Stores host architecture.
- #define ARM_FAMILY

  Stores the ARM family.
- #define CC

  Stores C compilator executable.
- #define CXX

  Stores C++ compilator executable.
- #define AR

  Stores archiver command.
- #define RM

  Stores remove command.
- #define ARM_FLAGS

  Stores specific ARM flags for compilation and linking.
- #define COMMON_FLAGS

  Defines the C and C++ common flags (compilation and linking)
- #define COMPIL_FLAGS

  Defines the C and C++ common compilation flags.
- #define CFLAGS

  Defines the C compilation flags.
- #define CXXFLAGS

  Defines the C++ compilation flags.
- #define LDFLAGS

  Defines the linker flags.

5.1.1 Detailed Description

Generic project build file.

** Copyright (c) 2018 by Tuxin ** ** All Rights Reserved **

** Version 1.0.0 April 25, 2018 **

```
This makefile can generate a binary by automatically detecting
the files of a project, provided that it respects a pre-defined
directory architecture.
```

Author

Tuxin (Jean-Pierre)

Version

1.0.0

Since

Created 04/25/2018 (JPB)

Date

April 25, 2018

Definition in file Makefile.