

# CMake FrameWork User Manual

## Document Revision Details

| sl no | Change Details  | Document Revisions      |
|-------|---|-------------------------|
| 1     | Initial Documentation of CMake Framework              | rtss_cmake_scripts_v0.1 |
| 2     | Cmake support for GNSS-M55 core                       | rtss_cmake_scripts_v0.2 |
| 3     | Contents of scripts directory moved to base directory | rtss_cmake_scripts_v0.3 |
| 4     | Additional Compiler Arguments support                 | rtss_cmake_scripts_v0.4 |
| 5     | GCC compiler Support                                  | rtss_cmake_scripts_v0.5 |
| 6     | Support for Windows Host machine                      | rtss_cmake_scripts_v0.6 |
| 7     | Multi test applications build support                 | rtss_cmake_scripts_v0.7 |
| 8     | Optimizing running steps and feature enhancement      | rtss_cmake_scripts_v0.8 |
| 9     | Update running steps                                  | rtss_cmake_scripts_v0.9 |
| 10    | Adding support to build from pack                     | rtss_cmake_scripts_v1.0 |
| 11    | Device Selection support added                        | rtss_cmake_scripts_v1.1 |
| 12    | Help Option is added                                  | rtss_cmake_scripts_v1.2 |

# Table of Contents

## Contents

|   |    |
|---|----|
| System Requirements to run the Scripts .....                    | 4  |
| Ubuntu .....  | 4  |
| Windows .....   | 4  |
| Contents of the 'rtss_cmake_scripts' repo .....                 | 5  |
| cmake_rtss directory .....                                      | 5  |
| Shell Script files .....  | 6  |
| Steps to run the Shell Scripts .....                            | 7  |
| git_clone.sh .....  | 7  |
| setup_user_env.sh .....   | 8  |
| run.sh .....  | 10 |
| Note Regarding passing Arguments to shell scripts .....         | 11 |
| Modifying CMake scripts .....                                   | 12 |
| Adding New Driver Files .....                                   | 12 |
| scripts/cmake_rtss/dirvers_cmake. ....                          | 12 |
| Editing/Adding Test Applications in cmake for the Build .....   | 12 |
| scripts/cmake_rtss/os_cmake .....                               | 12 |
| Editing/Adding RTE_Components.h header file for the Build ..... | 13 |

# rtss\_cmake\_scripts

This document contains detailed information on how to use the CMake scripts and helper shell scripts, to clone git repos or pack setup, compile test applications and generate binary files with respect to **Cortex-M55**.

## System Requirements to run the Scripts

### Ubuntu

1. CMake version on the Linux Host system should be having the latest version **3.25** and above.
2. Ubuntu OS version on the Host system should be at least **18.4 and above**.

### Steps to update the CMake version on the Host

Run the below commands on the terminal to update the Cmake version to the latest version.

1. `sudo apt purge --auto-remove cmake`
2. `wget -O - https://apt.kitware.com/keys/kitware-archive-latest.asc 2>/dev/null | gpg --dearmor - | sudo tee /etc/apt/trusted.gpg.d/kitware.gpg >/dev/null`

### For Ubuntu 18.04

1. `sudo apt-add-repository 'deb https://apt.kitware.com/ubuntu/ bionic main'`
2. `sudo apt update`
3. `sudo apt install cmake`

### Windows

1. CMake version on the Windows Host system should be having the latest version **3.25** and above.
2. MingW64 should be installed, which provides the application named "GitBash".
3. Git should be installed.
4. GitBash application should be used as Terminal console while using this CMake Framework.

## Contents of the 'rtss\_cmake\_scripts' repo

1. **cmake\_rtss** → This directory contains all the CMake files, related to tool-chain and projects source files with respect to rtss.
2. **Include\_RTE\_Comp** → This directory contains "RTE\_Components.h" header file specifically for Crescendo or Ensemble packs.
3. **Shell Scripts** → These shell scripts are required to clone the repo, setup compiler environment and generate the binaries using CMake files. *'The Shell Scripts should be accessed first to run "git\_clone.sh"/ "packs\_setup.sh" and then "run.sh", which does the above- mentioned tasks one by one'.*

### cmake\_rtss directory

1. **device\_cmake** → This contains cmake file which has the details related to Devices bolt\_extsys\_driver or extsys0 and extsys1 for HP and HE).
2. **drivers\_cmake** → This contains cmake file which has the details of source files and header files of all the supporting drivers for cortex\_m55.
3. **ensemble\_cmake** → This contains cmake file which has the details of source files and header files with respect to Ensemble package.
4. **modem\_ss\_cmake** → This contains cmake file which has the details of Middleware source files and header files with respect to modem core.
5. **netxduo\_cmake** → This contains cmake file which has the details of source files and header files with respect to netowrk module.
6. **os\_cmake** → This contains cmake files which have the details of source files and header files to include with respect to different Operating Systems. It also has details of test applications to be built w.r.t each OS.
7. **toolchains** → This contains cmake file which has the compiler toolchain related details with respect to Cortex-M55.

8. **usbx\_cmake** → This contains cmake file which has the details of source files and header files of USBX driver.
9. **CMakeLists.txt** → This is the main cmake file, which links all the above cmake files based on the configuration and requirement.
10. **utilities\_func.cmake** → This cmake file has helper functions and macro definitions required for the build.
11. **rtss\_compiler\_config.cmake** → This cmake contains default configurable compiler argument variables for *Cortex-M55*, which can be modified as per developer requirement only.

### Include\_RTE\_Comp

1. **crescendo** → This directory consists of **RTE\_Components.h** header file specifically for **Crescendo pack**. It is modified during the run time, to match with Device type **M55\_HE** or **M55\_HP**.
2. **ensemble** → This directory consists of **RTE\_Components.h** header file specifically for **Ensemble pack**. It is modified during the run time, to match with Device type **M55\_HE** or **M55\_HP**.

### Shell Script files

1. **setup\_user\_env.sh** → This script basically setups the environment variables for the compiler configuration and the license file path specifically for Host.
2. **git\_clone.sh** → This shell script is used to clone the git repos from Alif's internal git server to get required source files to compile and generate binaries for the CM-55.

**OR**

**packs\_setup.sh** → This shell script is used to extract Alif's release pack to get required source files to compile and generate binaries for the CM-55.

3. **run.sh** → This script takes in arguments required for the compilation to generate Makefile and run the makefile internally to generate binaries for each test applications respectively.

## Steps to run the Shell Scripts

### git\_clone.sh

1. Command syntax -->

```
./git_clone.sh GIT_USER_NAME=gerrit_username GIT_REPO_PATH=path_to_git_repo_files
```

2. Here **gerrit\_username** will be the username provided or registered with one's Gerrit account on their NUC.
3. If the user does not have a User account with the Gerrit, they can use **NONE** for the **GIT\_USER\_NAME**.
4. And **path\_to\_git\_repo\_files**, is the path to a directory provided by the user where all the required git repo source codes will be cloned into.
5. This script has git repo's links to all the required source codes of the Middleware, OS files and Test applications.
6. ***"This script has to be updated in case any new repo needs to be considered while generating the binaries"***.
7. This script also updates the path of required Header files in the Scatter Linker files of both **M55\_HP** and **M55\_HE** devices.
8. Command example -->

```
./git_clone.sh GIT_USER_NAME=NONE GIT_REPO_PATH=$HOME/temp/source
```

```
rajranjan@ALIFSEMI:/mnt/d/platform/PMU/repo/b0cmakeLin/rtss_cmake_scripts$  
rajranjan@ALIFSEMI:/mnt/d/platform/PMU/repo/b0cmakeLin/rtss_cmake_scripts$  
rajranjan@ALIFSEMI:/mnt/d/platform/PMU/repo/b0cmakeLin/rtss_cmake_scripts$ ./git_clone.sh GIT_USER_NAME=ranjan GIT_REPO_PATH=source
```

OR

```
./packs_setup.sh PACKS_PATH=/mnt/d/platform/PMU/repo/pack/rajranjan
```

```
rajranjan@ALIFSEMI:/mnt/d/platform/PMU/repo/pack/rtss_cmake_scripts$  
rajranjan@ALIFSEMI:/mnt/d/platform/PMU/repo/pack/rtss_cmake_scripts$  
rajranjan@ALIFSEMI:/mnt/d/platform/PMU/repo/pack/rtss_cmake_scripts$  
rajranjan@ALIFSEMI:/mnt/d/platform/PMU/repo/pack/rtss_cmake_scripts$  
rajranjan@ALIFSEMI:/mnt/d/platform/PMU/repo/pack/rtss_cmake_scripts$ ./packs_setup.sh PACKS_PATH=/mnt/d/platform/PMU/repo/pack/rajranjan
```

## setup\_user\_env.sh

1. This script contains user required environment settings and tools installation's path which may differ system-to-system.
2. All these details are exported to specific environment variables.
3. User must configure following parameters as per host system.

```
#!/bin/bash

#.Configure:CMSIS.Version
export CMSIS_VERSION=5.9.0

#.Configure:CMSIS.Compiler.Version
export CMSIS_COMPILER_VERSION=1.0.0

#.Configure:compiler.(ARMCLANG.or.GCC.or.CLANG)
export COMPILER=GCC

#.Configure:ARM.License.Type
export ARM_PRODUCT_DEF=/opt/arm/developmentstudio-2022.0/sw/mappings/gold.elmap

#.Configure:Complete.Compiler.Path
export COMPILER_BIN_PATH=/mnt/d/Programs/arm-gnu-toolchain-12.3.rel1-x86_64-arm-none-eabi/bin/

#.Configure:Device/Part.Name.(default)
export DEVICE=AE722F80F55D5XX

#.Configure:Source.Code.Location.(default).[.ALIF_GIT|.PACK.]
export REPO_SRC=ALIF_GIT

#.Configure:CPU.Name.(default)
export CPU=M55

#.Configure:Board.Type.(default).[.DEVKIT_E7|.APPKIT.]
export BOARD=DEVKIT_E7

#.Configure:Sub-System.Type.(default).[.HP|.HE.]
export RTSS=HP

#.Configure:Board.Revision.(default).[.A0|.B0|.B1|.B2|.B3|.B4.]
export REV=B4

#.Configure:Operating.System.(default).[.THREADX|.FREERTOS|.CMSISRTOS|.NONE.]
export OS=NONE

#.Configure:Boot.Type.(default).[.MRAM|.TCM.]
export BOOT=TCM

#.Configure:Test.Apps.(default).[.UART4_Baremetal|.ALL.]
export TEST_APP=ALL

#.Configure:Clean.(default).[.NO|.YES|.FORCE.]
export CLEAN=YES
```



4. Make sure to update the variable “**CMSIS\_VERSION**”, “**CMSIS\_COMPILER\_VERSION**” and “**COMPILER**” with **ARMCLANG** or **GCC** based on the compiler requirement.
5. And based on the compiler, user must update “**ARM\_PRODUCT\_DEF**” and “**COMPILER\_BIN\_PATH**”.
6. Run “source setup\_user\_env.sh” command.

**Note:** User may get some warning related to **CMSIS\_COMPILER**, **LICENSE** path, if user does not need CMSIS\_COMPILER (i.e. retargeting), warning can be ignored. License path can vary based on system, so if user has proper license setting, license related warning can be ignored.

After running this script use can see default device which will be used. If user will press ‘y’, script will show all device list

```

rajranjan@ALIFSEMI:/mnt/d/platform/cmake/rev_support/rtss_cmake_scripts$ source setup_user_env.sh

Current Device AE722F80F55D5XX ; Do you want to change device? <y/N> y

ALIF DEVICES :

-----
|      E1 Series      |      E3 Series      |      E5 Series      |      E7 Series      |
|-----|-----|-----|-----|
| AE101F1061517LH    | AE302F80F55D5AE     | AE512F80F55D5AS     | AE722F80F55D5AS     |
| AE101F4071542LH    | AE302F80F5582AE     | AE512F80F5582AS     | AE722F80F55D5LS     |
|                     | AE302F80F55D5LE     | AE512F80F55D5LS     | AE722F80F55D5XX     |
|                     | AE302F80F5582LE     | AE512F80F5582LS     |                     |
|                     | AE302F80C1557LE     | AE512F80F55D5XX     |                     |
|                     | AE302F40C1537LE     | AE512F80F5582XX     |                     |
|                     | AE302F80F55D5XX     |                     |                     |
|                     | AE302F80F5582XX     |                     |                     |
|-----|-----|-----|-----|
|      E1C Series     |      B1C Series     |      XX Series      |      XX Series      |
|-----|-----|-----|-----|
| AE1C1F4051920PH    |                     |                     |                     |
| AE1C1F4051920HH    |                     |                     |                     |
| AE1C1F40319205H    |                     |                     |                     |
| AE1C1F1041010XX    |                     |                     |                     |
| AE1C1F1040505XX    |                     |                     |                     |
|-----|-----|-----|-----|

Select Desire Device : AE722F80F55D5XX

```

**Note:** All default values will be selected in this file. User can change the default values, further run time also user can override the build variables (only applicable for specific run session).

## run.sh

1. **Command syntax -->**  
*./run.sh RTSS=HP or HE DEVICE=AE722F80F55D5XX OS=NONE or THREADX or FREERTOS or CMSISRTOS BOOT=MRAM or TCM REPO\_SRC=ALIF\_GIT or PACK BOARD=DEVKIT\_E7 TEST\_APP=UART4\_Baremetal/ALL CLEAN=NO/YES/FORCE [JOB=32 DEVELOPER=YES]*
2. For **RTSS** variable, **HE** or **HP** should be used, to define the macro value in the CMake with respect to the Core or Proc used.
3. For the **DEVICE** variable, **AE722F80F55D5XX** (default) or list of device (given in setup\_user\_env.sh) should be used to define a macro for the device name used for the project.
4. For the **OS** variable, different OS types **THREADX**, **FREERTOS**, **CMSISRTOS** and **NONE** (*bare-metal*) should be used to define a macro for the OS type used during the build.
5. For **BOOT** variable, **MRAM** or **TCM** should be used to define a macro in cmake to give details on the specific scatter linker file to be selected during Linking stage of the compilation.
6. For the **REPO\_SRC** variable, **ALIF\_GIT** or **PACK** should be used to define a macro repo source i.e. location source code.
7. For the **BOARD** variable, **DEVKIT\_E7** should be used to define a macro for board name.
8. For **TEST\_APP** variable, any specific test application name to be built can be given or by giving **ALL** as the value to the variable, the script will compile and generate binaries for all the test application for the given OS type.
  - **TEST\_APP=ALL** → **Compile all available test applications.**
  - **TEST\_APP=demo\_appl,demoApp2,demo\_app3** → **Compile all 3 demo\_appl, demoApp2, demo\_app3 applications.**
  - **TEST\_APP=ALL,demo\_appl,demo\_app2** → **Compile all test applications except demo\_appl,demo\_app2.**
9. Clean build of the project can be handled by passing **YES/NO/FORCE** for **CLEAN** argument. The default argument value used should be **NO**. If user has selected **FORCE** option, it will delete build folder to clean cmake cache. *It is an optional argument; it can be neglected if not required.*
10. User can select **DEVELOPER=YES/NO**. If the user has selected **YES** git branch check will be removed, build can break but it is expected user will know correct combinations. *It is an optional argument; it can be neglected if not required.*
11. **C\_COMPILER\_ARG** and **ASM\_COMPILER\_ARG** are also optional arguments. They can

be used when the user wants to pass different compiler arguments other than default values. *It is an optional argument; it can be neglected if not required.*

12. The user should make sure to pass all the required compiler arguments, because the default arguments from the config file will not be considered when these two arguments are used.
13. The compiler arguments when used, should always be passed within the **Double Quotes**.

**Example for compiler arguments: -**

```
C_COMPILER_ARG    -->    "-std=c99 -O0 -g -Wall -fshort-enums -fshort-wchar"
ASM_COMPILER_ARG  -->    "-g3 -Wall -masm=auto"
```

**RTSS command example -->**

```
./run.sh    OS=THREADX    RTSS=HP    DEVICE=    AE722F80F55D5XX    BOOT=TCM
TEST_APP=UART4_Baremetal    or    ALL    CLEAN=NO    [    JOB=16    DEVELOPER=YES
C_COMPILER_ARG="-std=c99    -O0    -g    -Wall    -fshort-    enums    -fshort-wchar"
ASM_COMPILER_ARG="-g3 -Wall -masm=auto"]
```

14. The run.sh script internally calls the ‘**cmake**’ command by passing all the above given details as arguments. This intern generates a **Makefile** in the ‘build’ directory as configured.
15. Once the Makefile is generated, the run.sh script runs the ‘**make**’ command from the build directory to compile and generate binaries in the ‘**rtss\_cmake\_scripts\build\_rtss\PACK\ARMCLANG\ENSEMBLE\DEVKIT\_E7\BAREMETAL**’ directory, in the specific folders with respect to the given arguments for the run.sh script.

**Note :** User can check all available option by running

```
./run.sh -help
```

## Note Regarding passing Arguments to shell scripts

1. Make sure to give complete path of the directories while passing arguments to **git\_clone.sh** and in **setup\_user\_env.sh** script files.
2. Make sure **not to give extra slash** at the end of the **source or build directory path**.

## Modifying CMake scripts

When it comes to modifying cmake script files, care should be taken as wrong changes would result in compilation error or even make file generation might fail. There are different steps involved in modifying any cmake scripts.

*Please follow the below steps, if cmake script modification is necessary.*

### Adding New Driver Files

1. The **bolt\_drivers.cmake** script file is present in the directory path **scripts/cmake\_rtss/dirvers\_cmake**.
2. This cmake file has each driver source files and headers files included with respect to each driver available.
3. Each driver related files are included to the build, based on respective macros which are enabled in the **RTE\_Components.h** header file.
4. When a new driver is added to the git repo, please make sure to add the same in this cmake file, so that it will be compiled for the new build.

```
# Directories and source files for MIPI Driver
CHECK_DEF ("#define RTE_MIPI_DSI 1" "${RTE_DEVICE_FILE}" ret_val1)
CHECK_DEF ("#define RTE_Drivers_MIPI_DSI 1" "${RTE_COMPONENTS_FILE}" ret_val2)

if (ret_val1 AND ret_val2)
    set (MIPI_DRIVER_DIR "${DRIVER_DIR}/MIPI")
    file (GLOB_RECURSE MIPI_DRIVER_SRC "${MIPI_DRIVER_DIR}/*.c")
    SUBDIRLIST (" " ${MIPI_DRIVER_DIR})
else ()
    message (STATUS "RTE_MIPI_DSI or RTE_Drivers_MIPI_DSI not ENABLED")
endif ()
```

### Editing/Adding Test Applications in cmake for the Build

1. While cmake is building, the test applications are considered according to the OS name passed as an argument for the run.sh script.
2. The cmake files for each OS type are available in the directory **scripts/cmake\_rtss/os\_cmake**
3. Each cmake for particular OS type, has the commands in them to consider the test applications.
4. If **ALL** is passed as the test application name, all the test applications under that given OS type will be considered.

5. If a particular test application name is given, only that test application of the given OS will be considered while building.

```
30
31 # Collection all Test Applications wrt to Free-RTOS under one variable
32 file (GLOB_RECURSE TEST_APP_SRCS "${SRC_DIRECTORY}/bolt_apps/FreeRTOS/**/*.c")
33
```

## Editing/Adding RTE\_Components.h header file for the Build

1. There are two **RTE\_Components.h** header files in the directory **scripts/Include\_RTE\_Comp**, one for **Crescendo**, one for **Ensemble** pack and one for **GNSS-M55**.
2. Based on the values passed with **CORE** and **DEVICE** as arguments to **run.sh** script, the header file will be modified by the cmake script on the run time, with respect to the given configuration.
3. If there are any new Drivers added in the git repo source, then driver cmake is updated with the same. And make sure to add the required macro definitions in **RTE\_Components.h** to enable the particular drivers for the compilation.

```
49 /* AlifSemiconductor::CMSIS Driver.SOC Peripherals.Modem Network */
50 #define RTE_Drivers_MODEM_NETX 1 /* Driver Modem-netx */
51 /* AlifSemiconductor::CMSIS Driver.SOC Peripherals.Modem Communication */
52 #define RTE_Drivers_IPC_MODEM_SS 1 /* Modem IPC drivers */
53 /* AlifSemiconductor::SDK Middleware.MPSS.ATCMD Middleware */
54 #define RTE_SDK_ATCMD_MIDDLEWARE 1
```