I-CUBE-LRWAN – Cmake
(Summary)

Table of contents:

I. Introduction:

1) Project's goal:

The purpose of this document is to summarize the changes made to the I-CUBE-LRWAN project [1]. This project is the LoRa software expansion for STM32Cube; a set of embedded software bricks designed to ease development for STM32 platforms.

I-CUBE-LRWAN features source code of LoRa applications for the following boards:

- B-L072Z-LRWAN1
- STM32L053R8-Nucleo
- STM32L073RZ-Nucleo
- STM32L152RE-Nucleo
- STM32L476RG-Nucleo

The goal is to be able to generate makefiles for each mode of said boards easily; therefore the cmake scripting language [2] was used. It is a build system that uses a single script to generate appropriate platform-specific build scripts.

The project is an evolving programme open to subsequent adjustments and additions of other boards and platforms.

2) Prerequisites:

In order to run the project on linux environment, cmake must be installed as well as the Arm development environmet for eclipse which include: node-js, xpm. To install the arm compiler:

```
xpm install --global @gnu-mcu-eclipse/arm-none-eabi-gcc
xpm install --global @gnu-mcu-eclipse/openocd
```

N.B.: To run the cmake command, we must specify the location of the installed toolcahin, Example:

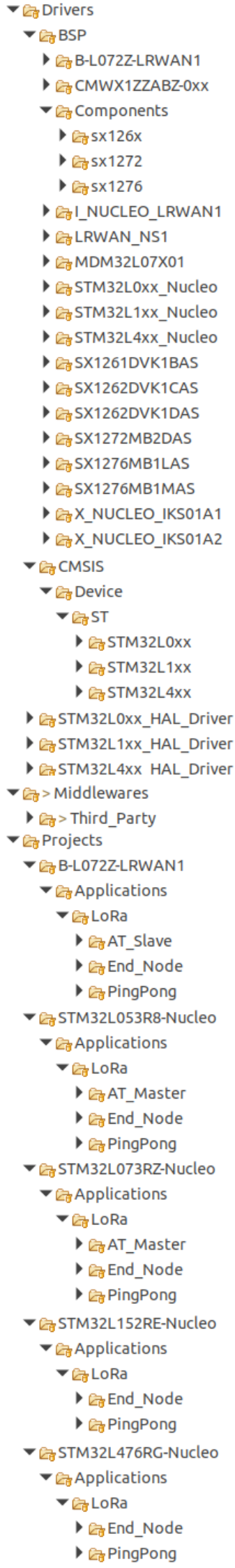
```
cmake -DCMAKE_TOOLCHAIN_FILE="cmake/toolchain-arm-none-eabi.cmake" -
DTOOLCHAIN_PREFIX="/home/abdallah/opt/xPacks/@gnu-mcu-eclipse/arm-none-eabi-gcc/
8.2.1-1.4.1/.content/" ..
```

The cache, located in the CmakeCache.txt in the build directory, should be cleared when importing the project from another PC.

On eclipse IDE, the project must be converted to a C++ project, with the ARM file GCC option. In the build options (Properties/Builds/C++), the build's directory should be specified and the build variable should be set to 'make'.

II. CMakeLists.txt files tree:

Two types of files were used to add the cmake features: CMakeLists.txt and stm32*$BoardSerie*.cmake.
CmakeLists.txt files are files containing a set of instructions describing the project's source files and targets (executable or library).
stm32*$BoardSerie*.cmake files contain function definitions, macros which can then be reused across multiple CmakeLists.txt files for the specified board serie.
The following figure lists all the directories in which a CmakeLists.txt file was created:


Drivers
BSP
B-L072Z-LRWAN1
CMWX1ZZABZ-0xx
Components
sx126x
sx1272
sx1276
I_NUCLEO_LRWAN1
LRWAN_NS1
MDM32L07X01
STM32L0xx_Nucleo
STM32L1xx_Nucleo
STM32L4xx_Nucleo
SX1261DVK1BAS
SX1262DVK1CAS
SX1262DVK1DAS
SX1272MB2DAS
SX1276MB1LAS
SX1276MB1MAS
X_NUCLEO_IKS01A1
X_NUCLEO_IKS01A2
CMSIS
Device
ST
STM32L0xx
STM32L1xx
STM32L4xx
STM32L0xx_HAL_Driver
STM32L1xx_HAL_Driver
STM32L4xx HAL_Driver
> Middlewares
> Third_Party
Projects
B-L072Z-LRWAN1
Applications
LoRa
AT_Slave
End_Node
PingPong
STM32L053R8-Nucleo
Applications
LoRa
AT_Master
End_Node
PingPong
STM32L073RZ-Nucleo
Applications
LoRa
AT_Master
End_Node
PingPong
STM32L152RE-Nucleo
Applications
LoRa
End_Node
PingPong
STM32L476RG-Nucleo
Applications
LoRa
End_Node
PingPong

III. Different categories of CMakeLists.txt:

All of the CMakeLists.txt created in the previously mentioned locations can be divided into four categories.

1) Global CMakeLists.txt

The CmakeFiles.txt file at the root of the project's directory defines the options that generate the specefic build for the chosen board and its extensions.
The following are the options listed so far in our project:

a) APPLICATION: The mode used by the board's communication
- End Node
- AT
- PingPong

b) CRYPTO: The cryptography method used by the board
- PearlIoT
- soft-se

c) PROJECT: The board used
- B-L072Z-LRWAN1
- STM32L053R8-Nucleo
- STM32L073RZ-Nucleo
- STM32L152RE-Nucleo
- STM32L476RG-Nucleo

d) RADIO: Radio module
- sx126x
- sx1272
- sx1276

e) SENSOR_BOARD: The sensor board attached to the board
- NO_SENSOR_BOARD
- X_NUCLEO_IKS01A1
- X_NUCLEO_IKS01A2

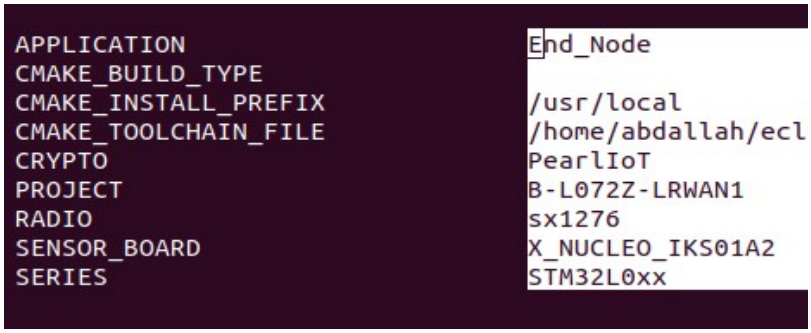
f) SERIE: The board's serie
- STM32L0xx
- STM32L1xx
- STM32L4xx

N.B.: It is important to specify in this file the following line: '`enable_language(ASM)`' in order to be able to read the startup files (.s).

---

2) Option's subdirectory:

When using the command 'ccmake', the following figure is shown in the terminal:



The options chosen by the user set the according option variables (APPLICATION, CRYPTO,…) to the chosen values. Multiple CmakeFiles.txt files were added to include the directories needed for the build with the specified options using the '`add_subdirectory()`' command.

Example: File that adds the CMSIS drivers for the specified serie located at:
`/Drivers/CMSIS/CMakeLists.txt`

```
##-------------Serie choice-------------##
#
#
cmake_minimum_required(VERSION 3.6)

#Series
if(SERIES STREQUAL STM32L0xx)
      add_subdirectory(${CMAKE_CURRENT_SOURCE_DIR}/Device/ST/STM32L0xx)
elseif(SERIES STREQUAL STM32L1xx)
      add_subdirectory(${CMAKE_CURRENT_SOURCE_DIR}/Device/ST/STM32L1xx)
elseif(SERIES STREQUAL STM32L4xx)
      add_subdirectory(${CMAKE_CURRENT_SOURCE_DIR}/Device/ST/STM32L4xx)
endif()
```

This type of CMakeLists.txt file is located in the following directories:

/Projects/CMakeLists.txt
/Projects/B-L072Z-LRWAN1/CMakeLists.txt
/Projects/STM32L152RE-Nucleo/CMakeLists.txt
/Projects/STM32L073RZ-Nucleo/CMakeLists.txt
/Projects/STM32L053R8-Nucleo/CMakeLists.txt
./Drivers/CMakeLists.txt
./Drivers/BSP/CMakeLists.txt
./Drivers/BSP/Components/CMakeLists.txt
./Drivers/CMSIS/CMakeLists.txt

N.B.: For files located in the /Projects directory, the corresponding linker script is specified according to the board chosen in the options. An include of a .cmake file is also specified when setting the choice for the boards, each board includes its corresponding .cmake file. The .cmake files are explained in detail in paragraph IV.

---

3) Driver's library:

This file category is used to add source files to a library with a specified name, which will then be linked to the executables.

Example: File that adds the source files of the HAL drivers for the specified STM32L0xx serie located at:
`/Drivers/STM32L0xx_HAL_Driver/CMakeLists.txt`

```
##-------------STM32L0xx_HAL_Driver-------------##
#
#The name of the created project will be called when linking this library to its
executable
project(STM32L0xx_HAL_Driver)
#The cmake minimum required version must be specified in each CmakeLists.txt
file
```

```
cmake_minimum_required(VERSION 3.6)

#Specify the project's source files; adding all .c files located at said
directory to the variable 'STM32L0xx_HAL_Driver_SOURCES'
file(GLOB ${PROJECT_NAME}_SOURCES
    "${CMAKE_CURRENT_SOURCE_DIR}/Src/*.c"
)


#All files added to the variable 'STM32L0xx_HAL_Driver_SOURCES' are added to the
library called  'STM32L0xx_HAL_Driver' which can then be called in the
executables
add_library(${PROJECT_NAME} OBJECT EXCLUDE_FROM_ALL ${${PROJECT_NAME}_SOURCES})

#This command specifies that the 'CMSIS-STM32L0xx' library should be built
before the 'STM32L0xx_HAL_Driver'
add_dependencies(${PROJECT_NAME} CMSIS-STM32L0xx)


#Add all the header files. When this library is linked to its executable, this
command acts as if we added #include to all .h files found at the specified
directory
#We also added header files from other libraries which are needed by this
library
#The variable ${PROJECT}-${APPLICATION} refers to the chosen board and
application by the user, hence by using this command we include the header files
of the executable
target_include_directories( ${PROJECT_NAME} PUBLIC
     ${CMAKE_CURRENT_SOURCE_DIR}/Inc
     $<TARGET_PROPERTY:middlewares,INTERFACE_INCLUDE_DIRECTORIES>
     $<TARGET_PROPERTY:CMSIS-STM32L0xx,INTERFACE_INCLUDE_DIRECTORIES>
     $<TARGET_PROPERTY:${PROJECT}-${APPLICATION},INTERFACE_INCLUDE_DIRECTORIES>
)
#Specify the C standard whose features are requested to build this target
set_property(TARGET ${PROJECT_NAME} PROPERTY C_STANDARD 11)
```

The code is practically the same in all files of this category, only the library's name is changed (using the '`project()`' command). For practical reasons, the library's name is always the name of the folder containing it.

This type of files is located in the following directories:
./Middlewares/CMakeLists.txt
./Middlewares/Third_Party/LoRaWAN/Utilities/CMakeLists.txt
./Drivers/BSP/B-L072Z-LRWAN1/CMakeLists.txt
./Drivers/BSP/CMWX1ZZABZ-0xx/CMakeLists.txt
./Drivers/BSP/Components/sx1272/CMakeLists.txt
./Drivers/BSP/Components/CMakeLists.txt
./Drivers/BSP/Components/sx126x/CMakeLists.txt
./Drivers/BSP/Components/sx1276/CMakeLists.txt
./Drivers/BSP/I_NUCLEO_LRWAN1/CMakeLists.txt
./Drivers/BSP/LRWAN_NS1/CMakeLists.txt
./Drivers/BSP/MDM32L07X01/CMakeLists.txt
./Drivers/BSP/STM32L0xx_Nucleo/CMakeLists.txt
./Drivers/BSP/STM32L1xx_Nucleo/CMakeLists.txt
./Drivers/BSP/STM32L4xx_Nucleo/CMakeLists.txt
./Drivers/BSP/SX1261DVK1BAS/CMakeLists.txt

./Drivers/BSP/SX1262DVK1CAS/CMakeLists.txt
./Drivers/BSP/SX1262DVK1DAS/CMakeLists.txt
./Drivers/BSP/SX1272MB2DAS/CMakeLists.txt
./Drivers/BSP/SX1276MB1LAS/CMakeLists.txt
./Drivers/BSP/SX1276MB1MAS/CMakeLists.txt
./Drivers/BSP/X_NUCLEO_IKS01A1/CMakeLists.txt
./Drivers/BSP/X_NUCLEO_IKS01A2/CMakeLists.txt
./Drivers/STM32L0xx_HAL_Driver/CMakeLists.txt
./Drivers/STM32L1xx_HAL_Driver/CMakeLists.txt
./Drivers/STM32L4xx_HAL_Driver/CMakeLists.txt
./Drivers/CMSIS/Device/ST/STM32L0xx/CMakeLists.txt
./Drivers/CMSIS/Device/ST/STM32L1xx/CMakeLists.txt
./Drivers/CMSIS/Device/ST/STM32L4x/CMakeLists.txt

N.B.: Some CmakeFiles.txt files are a combination of the last two categories which means that certain source and header files are added to the library depending on the user's choice of the options (APPLICATION, CRYPTO, ...etc)

---

4) Executables:

For each application of each board, the CMakeLists.txt adds the source files for the executable. In the following example, the different sections of the code and the instructions used will be explained in the comments.

Example: File for the End Node application of the B-L072Z-LRWAN1 board located at: `/Projects/B-L072Z-LRWAN1/Applications/LoRa/End_Node/CMakeLists.txt`

```
##-------------B-L072Z-LRWAN1-End_Node-------------##
#
#
#The name of the created project will be called when linking this executable to
its libraries
project(B-L072Z-LRWAN1-End_Node)
cmake_minimum_required(VERSION 3.6)

#Include the .cmake file of the toolchain
include (binutils-arm-none-eabi)

#Specify the project's source files; adding all .c files located at said
directory to the variable 'B-L072Z-LRWAN1-End_Node_SOURCES'
file(GLOB ${PROJECT_NAME}_SOURCES
    "${CMAKE_CURRENT_SOURCE_DIR}/Core/src/*.c"
    "${CMAKE_CURRENT_SOURCE_DIR}/LoRaWAN/App/src/*.c"
    "${CMAKE_CURRENT_SOURCE_DIR}/SW4STM32/startup_stm32l072xx.s"

)
#For the build to be able to take the startup file into account
set_property(SOURCE startup_stm32l072xx.s PROPERTY LANGUAGE C)

#-------------------------------------------------------------------------------
--------
# Libraries - Common libraries independent from choices
#Due to the fact that a board can have different extensions added onto it, we
separated the linking to the libraries into two parts.
```

```
#The first being the libraries that need to be linked regardless of the
extensions added to the board
#-------------------------------------------------------------------------
--------

#All files added to the variable 'B-L072Z-LRWAN1-End_Node_SOURCES' are added to
the executable called  'B-L072Z-LRWAN1-End_Node' which will then be linked in
the libraries
#We use add executable because this is not a library
add_executable(${PROJECT_NAME} ${${PROJECT_NAME}_SOURCES}
      $<TARGET_OBJECTS:middlewares>
      $<TARGET_OBJECTS:BSP-B-L072Z-LRWAN1>
      $<TARGET_OBJECTS:BSP-CMWX1ZZABZ-0xx>
      $<TARGET_OBJECTS:CMSIS-STM32L0xx>
      $<TARGET_OBJECTS:STM32L0xx_HAL_Driver>
      $<TARGET_OBJECTS:Components>
      )

#All libraries should be built before the executable
add_dependencies(${PROJECT_NAME} middlewares)
add_dependencies(${PROJECT_NAME} BSP-B-L072Z-LRWAN1)
add_dependencies(${PROJECT_NAME} BSP-CMWX1ZZABZ-0xx)
add_dependencies(${PROJECT_NAME} CMSIS-STM32L0xx)
add_dependencies(${PROJECT_NAME} STM32L0xx_HAL_Driver)
add_dependencies(${PROJECT_NAME} Components)


#Add all the header files
target_include_directories( ${PROJECT_NAME} PUBLIC
    ${CMAKE_CURRENT_SOURCE_DIR}/Core/inc
    ${CMAKE_CURRENT_SOURCE_DIR}/LoRaWAN/App/inc
      $<TARGET_PROPERTY:middlewares,INTERFACE_INCLUDE_DIRECTORIES>
    $<TARGET_PROPERTY:BSP-B-L072Z-LRWAN1,INTERFACE_INCLUDE_DIRECTORIES>
    $<TARGET_PROPERTY:BSP-CMWX1ZZABZ-0xx,INTERFACE_INCLUDE_DIRECTORIES>
    $<TARGET_PROPERTY:CMSIS-STM32L0xx,INTERFACE_INCLUDE_DIRECTORIES>
      $<TARGET_PROPERTY:STM32L0xx_HAL_Driver,INTERFACE_INCLUDE_DIRECTORIES>
      $<TARGET_PROPERTY:Components,INTERFACE_INCLUDE_DIRECTORIES>
)

#-------------------------------------------------------------------------
--------
#The second part are the libraries that should be linked only according the
choices in the options
#-------------------------------------------------------------------------
--------

#-------------------------------------------------------------------------
--------
# Libraries - Sensor boards
#-------------------------------------------------------------------------
--------

#The 'SENSOR_ENABLED' is defined in this file instead of in the stm32l072.cmake
file because it should not be defined for every use of the stm32l072 boards but
only when a sensor board is added.
if(SENSOR_BOARD STREQUAL X_NUCLEO_IKS01A1)
      target_compile_definitions(${PROJECT_NAME} PUBLIC -DSENSOR_ENABLED)
      target_link_libraries(${PROJECT_NAME} $<TARGET_OBJECTS:BSP-
X_NUCLEO_IKS01A1>)
      add_dependencies(${PROJECT_NAME} BSP-X_NUCLEO_IKS01A1)
```

```
      target_include_directories( ${PROJECT_NAME} PUBLIC
            $<TARGET_PROPERTY:BSP-
X_NUCLEO_IKS01A1,INTERFACE_INCLUDE_DIRECTORIES>
      )
elseif(SENSOR_BOARD STREQUAL X_NUCLEO_IKS01A2)
      target_compile_definitions(${PROJECT_NAME} PUBLIC -DSENSOR_ENABLED)
      target_link_libraries(${PROJECT_NAME} $<TARGET_OBJECTS:BSP-
X_NUCLEO_IKS01A2>)
      add_dependencies(${PROJECT_NAME} BSP-X_NUCLEO_IKS01A2)
      target_include_directories( ${PROJECT_NAME} PUBLIC
            $<TARGET_PROPERTY:BSP-
X_NUCLEO_IKS01A2,INTERFACE_INCLUDE_DIRECTORIES>
      )
endif()

#-------------------------------------------------------------------------------
--------
# Libraries - Radio
#-------------------------------------------------------------------------------
--------
if(RADIO STREQUAL sx1276)
      target_link_libraries(${PROJECT_NAME} $<TARGET_OBJECTS:sx1276>)
      add_dependencies(${PROJECT_NAME} sx1276)
      target_include_directories( ${PROJECT_NAME} PUBLIC
            $<TARGET_PROPERTY:sx1276,INTERFACE_INCLUDE_DIRECTORIES>
      )
else()
      message(FATAL_ERROR "You can only choose the sx1276 for the BL072Z-
LRWAN1")
endif()


# Add define if debbuger support is enabled
target_compile_definitions(${PROJECT_NAME} PUBLIC $<$<BOOL:$
{USE_DEBUGGER}>:USE_DEBUGGER>)

#Specify the C standard whose features are requested to build this target
set_property(TARGET ${PROJECT_NAME} PROPERTY C_STANDARD 11)

# Create output in hex and binary format
create_bin_output(${PROJECT_NAME})
create_hex_output(${PROJECT_NAME})
```

This type of files is located in the following directories:

./Projects/STM32L152RE-Nucleo/Applications/LoRa/PingPong/CMakeLists.txt
./Projects/STM32L152RE-Nucleo/Applications/LoRa/End_Node/CMakeLists.txt
./Projects/STM32L073RZ-Nucleo/Applications/LoRa/AT_Master/CMakeLists.txt
./Projects/STM32L073RZ-Nucleo/Applications/LoRa/PingPong/CMakeLists.txt
./Projects/STM32L073RZ-Nucleo/Applications/LoRa/End_Node/CMakeLists.txt
./Projects/STM32L053R8-Nucleo/Applications/LoRa/AT_Master/CMakeLists.txt
./Projects/STM32L053R8-Nucleo/Applications/LoRa/PingPong/CMakeLists.txt
./Projects/STM32L053R8-Nucleo/Applications/LoRa/End_Node/CMakeLists.txt
./Projects/STM32L476RG-Nucleo/Applications/LoRa/PingPong/CMakeLists.txt
./Projects/STM32L476RG-Nucleo/Applications/LoRa/End_Node/CMakeLists.txt
./Projects/B-L072Z-LRWAN1/Applications/LoRa/AT_Slave/CMakeLists.txt

./Projects/B-L072Z-LRWAN1/Applications/LoRa/PingPong/CMakeLists.txt
./Projects/B-L072Z-LRWAN1/Applications/LoRa/End_Node/CMakeLists.txt

IV. Cmake files:

Cmake files are located in the /cmake directory which was copied from the LoRaMacNode project [3]. These files allow us to load and run CMake code from a file instead of listing all the options in the terminal.
We have chosen to create a cmake file for each board, that file is included when the corresponding board is chosen.

Example: File for the B-L072Z-LRWAN1 board:

```
## STM32L072 target specific CMake file
##

#Linker script verification
if(NOT DEFINED LINKER_SCRIPT)
message(FATAL_ERROR "No linker script defined")
endif(NOT DEFINED LINKER_SCRIPT)

#-------------------------------------------------------------------------------
--------
# Set compiler/linker flags
#-------------------------------------------------------------------------------
--------

# Object build options
set(OBJECT_GEN_FLAGS "-Og -g -mthumb -g2 -fno-builtin -mcpu=cortex-m0plus -Wall
-Wextra -pedantic -Wno-unused-parameter
#--
#All the necessary defines for the BL072Z-LRWAN1 board
#--
-DUSE_LRWAN_NS1
-DUSE_BAND_868
-DUSE_MODEM_LORA
-DUSE_MDM32L07X01
-DUSE_HAL_DRIVER
-DSTM32L072xx
-DDEBUG
-DLOW_POWER_DISABLE
-DUSE_B_L072Z_LRWAN1
-DREGION_EU868
-ffunction-sections -fdata-sections -fomit-frame-pointer -mabi=aapcs -fno-
unroll-loops -ffast-math -ftree-vectorize")

set(CMAKE_C_FLAGS "${OBJECT_GEN_FLAGS} -std=gnu99 " CACHE INTERNAL "C Compiler
options")
set(CMAKE_CXX_FLAGS "${OBJECT_GEN_FLAGS} -std=c++11 " CACHE INTERNAL "C++
Compiler options")
set(CMAKE_ASM_FLAGS "${OBJECT_GEN_FLAGS} -x assembler-with-cpp " CACHE INTERNAL
"ASM Compiler options")

# Linker flags
set(CMAKE_EXE_LINKER_FLAGS "-Wl,--gc-sections --specs=nano.specs --
specs=nosys.specs -mthumb -g2 -mcpu=cortex-m0plus -mabi=aapcs -T${LINKER_SCRIPT}
-Wl,-Map=${CMAKE_PROJECT_NAME}.map" CACHE INTERNAL "Linker options")
```

These definitions (-D) are called throughout the program in '#if defined' conditions to include header files if a certain variable is defined.

V. Adding a new element to the project:

This paragraph lists the different steps to be followed when importing a new board's source code into the project.

1) The source code:

The board's source code should be placed in the ./Projects/ directory, the board's serie as well as its compatibility with the different extensions should be analysed before implementing all of the necessarry options in the CMakeLists.txt files.

2) CMakeLists.txt:

Additional CMakeLists.txt files must be implemented in the following directories:

- ./Projects/*$Board_name* → Specifies the different Application options for the board
- ./Projects/*$Board_name/$Each_application* → Should be identical to CMakeLists.txt form other boards except for changing the executable's name using the '`project()`' command.

The CMakeLists.txt files that should be modified are the following:

- ./*$Root_Directory*/ → The first CMakeLists.txt file should be modified to include the new board in the 'PROJECT' option.
- ./Projects/ → The CMakeLists.txt file located in this directory should now include an additional 'elseif()' condition for the board where it specifies the Linker script file's directory and the include of the new board's .cmake file.

An additional .cmake file should be created in the ./cmake directory if the board is from a different serie.

VI. Changes made in the original project's source code:

To integrate Idemia's Secure Element in the I-CUBE-LRWAN porject, we added the LoRaMacCrypto-SE.c file in the ./Middlewares/Third_Party/LoRaWAN/Mac/ directory.

This file required changements to be made to the project:

- `void HAL_Delay(__IO uint32_t Delay)` function in the 'stm32l0xx_hal.c' was set to '__weak' because another function with the same name uses the RTC instead of the Systick variable.

VII. References:

[1] : User manual for the I-CUBE-LRWAN project
https://www.st.com/content/ccc/resource/technical/document/user_manual/group0/31/96/2f/3b/df/c1/40/2e/DM00300436/files/DM00300436.pdf/jcr:content/translations/en.DM00300436.pdf

[2] : Documentation for cmake scripting language
https://cmake.org/cmake/help/v3.10/manual/cmake.1.html

[3] : Github repository for the LoRaMacNode project
https://github.com/Lora-net/LoRaMac-node