Сборка проекта для cortex-m

Первоначально необходимо скачать и установить [cmake](https://cmake.org/download/), после чего в директории проекта создать два файла CMakeList.txt и toolchain.cmake.

В файле CMakeLists.txt происходит все настройка проекта, подключение директорий с заголовочными файлами, библиотек, ld файлов, ASM файлов и т.д. В файле toolchain.cmake прописывается какой компилятор использовать, флаги для компиляции, линковки и т.д.

Так же необходимо установить [make](https://gnuwin32.sourceforge.net/packages/make.htm).

Ниже будет приведен пример кода с аннотацией за что какая функция отвечает.

CMake: Основы

cmake\_minimum\_required(VERSION 3.26)

set(CMAKE\_TOOLCHAIN\_FILE ${CMAKE\_SOURCE\_DIR}/toolchain.cmake)

project(MyApp C ASM)

file(GLOB MAIN\_SRC "${CMAKE\_SOURCE\_DIR}/Core/src/\*.c")

file(GLOB DRIVERS\_SRC "${CMAKE\_SOURCE\_DIR}/Drivers/STM32F1xx\_HAL\_Driver/Src/\*.c")

add\_executable(${PROJECT\_NAME} ${MAIN\_SRC} ${DRIVERS\_SRC} startup\_stm32f103xb.s)

target\_include\_directories(${PROJECT\_NAME} PRIVATE

    ${CMAKE\_SOURCE\_DIR}/Core/Inc

    ${CMAKE\_SOURCE\_DIR}/Drivers/STM32F1xx\_HAL\_Driver/Inc

    ${CMAKE\_SOURCE\_DIR}/Drivers/STM32F1xx\_HAL\_Driver/Inc/Legacy

    ${CMAKE\_SOURCE\_DIR}/Drivers/CMSIS/Device/ST/STM32F1xx/Include

    ${CMAKE\_SOURCE\_DIR}/Drivers/CMSIS/Include

)

set(ELF\_OUTPUT\_FILE ${CMAKE\_BINARY\_DIR}/bin/${PROJECT\_NAME}.elf)

set(HEX\_OUTPUT\_FILE ${CMAKE\_BINARY\_DIR}/bin/${PROJECT\_NAME}.hex)

set(BIN\_OUTPUT\_FILE ${CMAKE\_BINARY\_DIR}/bin/${PROJECT\_NAME}.bin)

add\_custom\_command(TARGET ${PROJECT\_NAME} POST\_BUILD

    COMMAND ${CMAKE\_COMMAND} -E copy $<TARGET\_FILE:${PROJECT\_NAME}> ${ELF\_OUTPUT\_FILE}

    COMMAND ${CMAKE\_OBJCOPY} -O ihex $<TARGET\_FILE:${PROJECT\_NAME}> ${HEX\_OUTPUT\_FILE}

    COMMAND arm-none-eabi-objcopy -O binary ${ELF\_OUTPUT\_FILE} ${BIN\_OUTPUT\_FILE}

)

Пример CmakeLists.txt:

cmake\_minimum\_required (VERSION \*версия CMake\*) - это самая первая строка которую обязательно нужно прописать.

set(CMAKE\_TOOLCHAIN\_FILE \*путь до toolchain файла\*) – здесь в переменную CMAKE\_TOOLCHAIN\_FILE присваивается путь до toolcgain файла, обязательно прописать эту строку до команды project.

project (\*Имя проекта\* \*языки программирования\*) – здесь нужно придумать имя для проекта и прописать какие язки программирования используются.

file(GLOB \*название переменной\* \*путь до директории\*) – с помощью этой функции мы говорим что в переменной X будут храниться пути до необходимых файлов, так же можно использовать функцию set(\*название переменной\* \*путь до файла или директории\*) но так нам нужно руками прописывать путь до каждого файла, а с помощью функции file(GLOB ) можно передать путь до директории и прописать какого расширения файлы мы хотим поместить в это переменную.

add\_executable(\*имя файла после компиляции\* \*файлы необходимые для сборки проекта\*) – тут мы передаем как мы хотим что бы назывался исходный файл и то какие файлы нужны для сборки проекта, в примере видно что используются не только файлы с расширением .c которые хранятся в переменных MAIN\_SRC и DRIVERS\_SRC но так же еще и ASM файл который нужно подключать в последнюю очередь.

target\_include\_directories(\*имя файла после компиляции\* PRIVATE \*путь до директории \*) – С помощью этой функции мы прописываем пути до директорий которые хранят в себе заголовочные файлы, рекомендуется использовать модификатор PRIVET так как это означает что все зависимости буду использовать только для этой цели (название должны совпадать в функции add\_executable и target\_include\_directories, так как мы говорим что нужно подключить заголовочные файлы к этому проекту)

set(ELF\_OUTPUT\_FILE ${CMAKE\_BINARY\_DIR}/bin/${PROJECT\_NAME}.elf)

set(HEX\_OUTPUT\_FILE ${CMAKE\_BINARY\_DIR}/bin/${PROJECT\_NAME}.hex)

set(BIN\_OUTPUT\_FILE ${CMAKE\_BINARY\_DIR}/bin/${PROJECT\_NAME}.bin)

Здесь просто задается путь где буду храниться файлы .elf .hex и .bin

add\_custom\_command(TARGET ${PROJECT\_NAME} POST\_BUILD

    COMMAND ${CMAKE\_COMMAND} -E copy $<TARGET\_FILE:${PROJECT\_NAME}> ${ELF\_OUTPUT\_FILE}

    COMMAND ${CMAKE\_OBJCOPY} -O ihex $<TARGET\_FILE:${PROJECT\_NAME}> ${HEX\_OUTPUT\_FILE}

    COMMAND arm-none-eabi-objcopy -O binary ${ELF\_OUTPUT\_FILE} ${BIN\_OUTPUT\_FILE}

) – Здесь прописывается команда чтобы сгенерировать .elf .hex и .bin файлы

CMake: toolchain.cmake

set(TARGET\_CPU "cortex-m3")

set(CMAKE\_SYSTEM\_NAME Generic)

set(CMAKE\_C\_COMPILER "arm-none-eabi-gcc")

set(CMAKE\_CXX\_COMPILER "arm-none-eabi-g++")

set(CMAKE\_OBJECT\_PATH\_MAX 500)

set(CMAKE\_TRY\_COMPILE\_TARGET\_TYPE STATIC\_LIBRARY)

set(CMAKE\_FIND\_ROOT\_PATH\_MODE\_PROGRAM NEVER)

set(CMAKE\_FIND\_ROOT\_PATH\_MODE\_LIBRARY ONLY)

set(CMAKE\_FIND\_ROOT\_PATH\_MODE\_INCLUDE ONLY)

set(CMAKE\_C\_STANDARD 11)

set(CMAKE\_CXX\_STANDARD 14)

add\_compile\_options(

    -mcpu=cortex-m3 -std=gnu11 -g3 -DDEBUG -DUSE\_HAL\_DRIVER -DSTM32F103xB -O0 --specs=nano.specs -mfloat-abi=soft -mthumb

)

add\_link\_options(

    -mcpu=cortex-m3 -T "${CMAKE\_SOURCE\_DIR}/STM32F103RBTx\_FLASH.ld" --specs=nosys.specs -Wl,--gc-sections --specs=nano.specs -mfpu=fpv4-sp-d16 -mfloat-abi=hard -mthumb -Wl,--start-group -lc -lm -Wl,--end-group

)

Пример toolchain.cmake

set(CMAKE\_C\_COMPILER "arm-none-eabi-gcc")

set(CMAKE\_CXX\_COMPILER "arm-none-eabi-g++")

Так же можно прописать прямой путь до .exe файла компилятора

set(CMAKE\_TRY\_COMPILE\_TARGET\_TYPE STATIC\_LIBRARY) - Переменная CMAKE\_TRY\_COMPILE\_TARGET\_TYPE используется для указания целевого типа для временных двоичных файлов.

set(CMAKE\_FIND\_ROOT\_PATH\_MODE\_PROGRAM NEVER) - Эта переменная управляет тем, как CMake ищет исполняемые программы во время кросс-компиляции. Установив для него значение NEVER, вы, по сути, говорите CMake не искать программы (исполняемые файлы) в корневых путях, указанных в CMAKE\_FIND\_ROOT\_PATH.

set(CMAKE\_FIND\_ROOT\_PATH\_MODE\_LIBRARY ONLY) - Эта переменная управляет тем, как CMake ищет библиотеки во время кросс-компиляции. Установив значение ONLY, вы указываете CMake искать библиотеки только в корневых путях, указанных в CMAKE\_FIND\_ROOT\_PATH.

set(CMAKE\_FIND\_ROOT\_PATH\_MODE\_INCLUDE ONLY) - Эта переменная управляет тем, как CMake ищет файлы заголовков во время кросс-компиляции. Установив значение ONLY, вы указываете CMake искать файлы заголовков только в корневых путях, указанных в CMAKE\_FIND\_ROOT\_PATH.

add\_compile\_options( \*флаги для компилятора\* )

add\_link\_options(\*флаги для компоновщика\*)

Далее для того что бы собрать проект необходимо прописать команды прописывать команды в терминале. В терминале нужно перейти в директорию проекта и нужно прописать: cmake -S . -B ./build -G “Unix Makefiles”. После нужно прописать cmake –build ./build –clean-first