HowTo

Necessary tools

List of necessary tools:

1. STM32CubeIDE
   You can download it on this link (it will ask for your email) or you can get it from local repository.
2. STM32CubeMX
   You can download it on this link or you can get it from local repository.
3. arm-gnu-toolchain
   You can download it on this link or you can get it from local repository.
   It needs to be installed in specific directory: `C:\Tools\arm-gnu`
   This is because the makefile, which will be used in this tutorial, relies on this particular path. This can be changed if it needs to.

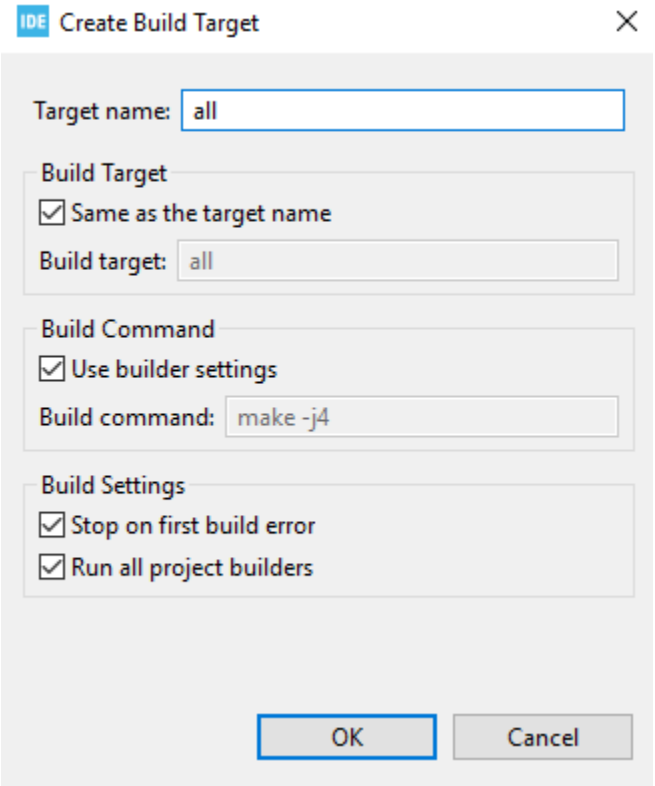
Start new STM32 project

Steps:

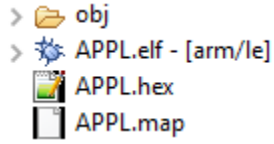
1. Open STM32CubeIDE
2. Choose “Start new STM32 project” from welcome screen
3. If you are working on STM board, choose from “Board selector” tab your board by searching and clicking on your board model and click “Next”

4. Name your project and click “Next”
5. From “Code generator options” section choose “Copy only the necessary library files” and click “Finish”
6. On all prompted windows click “Yes” until the IDE finishes its creation process and prompt the “<your_project_name>.ioc” file settings.
7. Now you can modify all settings related to your STM32 as you wish from the “Pinout & Configuration” tab. Under “Project management” tab, “Code generator” sub-tab in “Generated files” section make sure to check the box “Generate peripheral initialization as a pair of ‘.c/.h’ files per peripheral”, this will ensure that each peripheral is generated inside its own pair of ‘.c/.h’ file instead of the main.c file.
8. Once you are done with your customizations, press CTRL-S to save it. It will prompt for window for generating the configuration, press “Yes”

9. The files are generated and stored inside “Core” and “Drivers” folder, as well two. ld files, one for FLASH and the other for RAM. Delete the one with the RAM in its name.
10. Add makefile to the root of the project provided to you with this document.
11. makefile should be changed accordingly:
    a. BINUTILS_ROOT (line number 18) - should be changed to the install path of the arm-gnu-toolchain
    b. DEFS (line number 31) - should be changed to appropriate microcontroller that you are using in your project
    c. BUILD_DIR (line number 115) - should be set to whichever directory you prefer inside the project directory structure
    d. clean command (line number 199) - should match the BUILD_DIR
    e. PRODUCT_DIR (line number 223) - should be changed to “.” (related to makefile’s location to the root of the project)
12. Open Project -> Properties -> C/C++ Build -> Under “Builder Settings” tab in “Makefile generation” section, uncheck the box “Generate Makefiles automatically”
13. In Build location click “Workspace…” and on prompted window choose your project root folder and click “Ok”, afterwards click “Apply and Close”
14. In the “Build Targets” view create New Build Target with “Target name” as “all” in root directory



15. Do the same thing as previous step but with another target name: clean
16. As a test try building a project with the “all” build target. If everything worked correctly, you will find a folder/file structure inside the BUILD_DIR directory like this:



Building with the “clean” build target, mentioned structure should disappear from the project. Congratulations!

Workspace configuration tips

Changing generated code directory

You can achieve this simply moving "<your_project_name>.ioc" to the directory where you want these files to be generated. However, you might have a problem with opening the file from STM32CubeIDE. This can be solved by opening the file directly using STM32CubeMX. The process of generating differs a little, generating the configuration is done by clicking "GENERATE CODE" in top right corner of the window:

After the generation please be sure to follow second part of the step 7, deleting the .ld files from the directory. Mentioned files are generated only the first time after starting the project or moving the "<your_project_name>.ioc file to another directory, so the deleting part should be done only once.

Changing the location of makefile

You can archive this simply moving the makefile to the directory where you want it to be. Be sure to redo steps 11, 12 and 13 since these are related to the location of the makefile.

Building project

Once steps 12, 13 and 14 are done, building project is done the same way as the first half of step 16. There is no need for adding paths for include and source files to the build script since the script finds them all in the SRC_DIRS. Since the SRC_DIRS is defined as PRODUCT_DIR, all include and source files located in the project will be scheduled for build process.

At this moment there is no way to exclusive remove from build certain directories, however there is a tested method for excluding certain files from the build process:

```
117 INC_DIRS        := $$(call find_includes_in_dir, $$(SRC_DIRS))
118 HEADERS         := $$(foreach dir, $$(SRC_DIRS), $$(shell find $$(dir) -name "*.h")))
119 ASM_SRC         := $$(foreach dir, $$(SRC_DIRS), $$(shell find $$(dir) -name "*.s"))
120 C_SRC           := $$(foreach dir, $$(SRC_DIRS), $$(shell find $$(dir) -name "*.c -not -name "*main.c""))
121 CXX_SRC         := $$(foreach dir, $$(SRC_DIRS), $$(shell find $$(dir) -name "*.cpp"))
122 OBJECTS         := $$(addprefix $$(OBJ_DIR)/, $$(C_SRC:.c=.o) $$(CXX_SRC:.cpp=.o) $$(ASM_SRC:.s=.o))
123 LDSCRIPTS       := $$(addprefix -T, $$(foreach dir, $$(SRC_DIRS), $$(shell find $$(dir) -name "*.ld")))
```