**BUILD TIPS AND ADDING A PROJECT FOLDER IN MBED OS**

Arm Compiler 6 is the default Arm Compiler version for Mbed OS development. We created the Mbed command-line tool (Mbed CLI), a Python-based tool.Build script located in mbed/workspace\_tools/ is our core script solution to drive compilation, linking and building process for:

* Mbed SDK (with libs like Ethernet, RTOS, USB, USB host),
* tests which also can be linked with libraries like RTOS or Ethernet.

To Build a project the central script used for execution is build.py

For Example:python build.py -m LPC1768 -t ARMThis command will build Mbed SDK for LPC1768 platform using ARM compiler.

**build.py script**

Build.py script is a powerful tool to build Mbed SDK for all available platforms using all supported by mbed platforms compilers. Script is using our build script to create desired platform-compiler builds. Use script option **-h** (help) to check all script parameters:c:\temp\repos\mbed\workspace\_tools>build.py -h

Usage: build.py [options]

*Options:*

*-h, --help show this help message and exit*

*-m MCU, --mcu=MCU build for the given MCU (STM32F3XX, UBLOX\_C027,*

*DISCO\_F100RB, NRF51822, LPC11U35\_401, LPC1549, K64F,*

*LPC1347, STM32F407, NUCLEO\_L152RE, LPC11U35\_501,*

*NUCLEO\_F302R8, LPC2368, ARCH\_BLE, XADOW\_M0, LPC810,*

*LPC812, KL05Z, LPC11U68, LPC4088, LPC1114,*

*NUCLEO\_F072RB, NUCLEO\_L053R8, LPC11U24\_301,*

*LPC4330\_M4, LPC11C24, DISCO\_F051R8, KL46Z,*

*DISCO\_F303VC, NUCLEO\_F103RB, KL25Z, NUCLEO\_F030R8,*

*LPC1768, K20D5M, NUCLEO\_F401RE, LPC11U24,*

*DISCO\_F407VG)*

*-t TOOLCHAIN, --tool=TOOLCHAIN*

*build using the given TOOLCHAIN (GCC\_CW\_EWL, GCC\_CR,*

*GCC\_ARM, GCC\_CS, uARM, GCC\_CW\_NEWLIB, ARM, IAR)*

*-c, --clean clean the build directory*

*-o OPTIONS, --options=OPTIONS*

*Add a build option ("save-asm": save the asm generated*

*by the compiler, "debug-info": generate debugging*

*information, "analyze": run static code analyzer")*

*-r, --rtos Compile the rtos*

*-e, --eth Compile the ethernet library*

*-U, --usb\_host Compile the USB Host library*

*-u, --usb Compile the USB Device library*

*-d, --dsp Compile the DSP library*

*-F, --fat Compile FS ad SD card file system library*

*-v, --verbose Verbose diagnostic output*

*-b, --ublox Compile the u-blox library*

*-D MACROS Add a macro definition*

*-x, --extra-verbose-notifications*

*Makes compiler more verbose, CI friendly.*

The command line parameter **-m** specifies the MCUs/platforms for which you want to build the Mbed SDK. More than one MCU(s)/platform(s) may be specified with this parameter using comma as delimiter.In the example below (note there is no space after comma) will compile Mbed SDK for NXP LPC1768 and ST NUCLEO\_L152RE platforms using uARM (ARM Micro lib) compiler:

***python build.py -m \*\*LPC1768,NUCLEO\_L152RE\*\* -t uARM***

Parameter **-t** defined which toolchain should be used for Mbed SDK build. You can build Mbed SDK for multiple toolchains using one command. Below example (note there is no space after comma) will compile Mbed SDK for Freescale KL25Z platform using ARM and GCC ARM compilers:

***python build.py -m KL25Z -t \*\*ARM,GCC\_ARM\*\****

**Note**: You can combine this technique to compile multiple targets with multiple compilers:Below example will compile Mbed SDK for Freescale KL25Z and KL46Z platforms using ARM and GCC ARM compilers:

***python build.py -m \*\*KL25Z,KL46Z\*\* -t \*\*ARM,GCC\_ARM\*\****

Parameters **-r**, **-e**, **-u**, **-U**, **-d**, **-b** will add RTOS, Ethernet, USB, USB Host, DSP, U-Blox libraries respectively. Below example will build Mbed SDK library for for NXP LPC1768 platform together with RTOS (-r switch) and Ethernet (-e switch) libraries.

***python build.py -m \*\*LPC1768\*\* -t ARM -r -e***

if you want to change the path to your [GNU Tools for ARM Embedded Processors](https://launchpad.net/gcc-arm-embedded/4.7/4.7-2012-q4-major) to a path like c:/arm\_gcc/bin, you simply need to have a workspace\_tools/private\_settings.py that contains following line:

|  |  |
| --- | --- |
|  | ***GCC\_ARM\_PATH = "c:/arm\_gcc/bin"*** |

If a target and a toolchain are not specified, the build script will build the mbed library for all the available targets and toolchains.If, for example, you want to build the mbed library for the LPC1768 mbed using the ARM GCC toolchain:

|  |  |
| --- | --- |
|  | ***python workspace\_tools\build.py -m LPC1768 -t GCC\_ARM*** |

* The following options are used to request the build of additional libraries:
* -r: build the RTOS library
* -e: build the Ethernet library
* -u: build the USB device library
* -d: build the DSP library

**Component directories**

* The component labeled directories are used for software that implements funtionality. They are within label directories primarily because we don't expect every program to use this software, and including this software in every build would needlessly increase build time. The configuration value target.components entirely controls the set of directories the COMPONENT label type includes. The following is a shortened version of an example targets.json:
* {
* "NRF52\_DK": {
* "inherits": ["Target"],
* "components": ["SPIF"]
* }
* }
* In the above example, mbed compile includes files in directories named COMPONENT\_SPIF, and not directories such as COMPONENT\_SD or COMPONENT\_FLASHIAP.

**Toolchain directories**

* The toolchain labeled directories are used for toolchain specific files, such as assembly or linker files. The compilers use the following label sets:

| **Toolchain** | **Labels** |
| --- | --- |
| ARM compiler 5 | TOOLCHAIN\_ARM and TOOLCHAIN\_ARM\_STD |
| ARM compiler 5 with microlib | TOOLCHAIN\_ARM and TOOLCHAIN\_ARM\_MICRO |
| ARM compiler 6 | TOOLCHAIN\_ARM, TOOLCHAIN\_ARM\_STD and TOOLCHAIN\_ARMC6 |
| IAR EWARM | TOOLCHAIN\_IAR |
| GCC ARM Embedded | TOOLCHAIN\_GCC and TOOLCHAIN\_GCC\_ARM |

* When compiling with -t GCC\_ARM or mbed toolchain GCC\_ARM, source files found within TOOLCHAIN\_GCC and TOOLCHAIN\_GCC\_ARM are included, and files found within TOOLCHAIN\_IAR and TOOLCHAIN\_ARM are not.

## **.mbedignore**

* The .mbedignore rules override other rules for excluding files from a build. Files matching patterns in an .mbedignore file are excluded from a build even if a label rule or a test directory would include the file.You can place an .mbedignore file in any searched directory.
* Avoid defining rules that would cross library boundaries because these can lead to side effects or build problems that are hard to find.
* Each line in the .mbedignore file is a pattern for matching files. No files that matches any pattern found in any .mbedignore file are included when building or exporting.

The following wildcards are accepted:

| **Pattern** | **Meaning** |
| --- | --- |
| \* | Matches everything. |
| ? | Matches any single character. |
| [seq] | Matches any character in seq. |
| [!seq] | Matches any character not in seq. |

The file is parsed with Python's [fnmatch](https://docs.python.org/2/library/fnmatch.html). The syntax follows basic shell patterns with the following exceptions.Each line is treated as though it were prefixed with the path of the .mbedignore file.

**ADDING A PROJECT FOLDER**

* To add a Project Folder in mbed-os-example-blinky\mbed-os

For example “BSW” go to mbed-os-example-blinky\mbed-os\tools\paths.py

This python script contains paths of include fies needed to complie the mbed os project

Under Mbed Libraries Add

MBED\_BSW=join(ROOT,"BSW")//This adds the BSW folder path to the ROOT path

* In the same directory in projects.py add

from tools.paths import EXPORT\_DIR, MBED\_HAL, MBED\_BSW, MBED\_LIBRARIES, MBED\_TARGETS\_PATH

//This imports the “BSW” path to the Project

* In the same directory in builds\_api.py add the line

from .paths import (MBED\_CMSIS\_PATH, MBED\_TARGETS\_PATH, MBED\_LIBRARIES,

MBED\_HEADER, MBED\_DRIVERS, MBED\_PLATFORM, MBED\_HAL, MBED\_BSW,

MBED\_CONFIG\_FILE, MBED\_LIBRARIES\_DRIVERS,

MBED\_LIBRARIES\_PLATFORM, MBED\_LIBRARIES\_HAL,

BUILD\_DIR)//This adds the MBED\_BSW path to the builds

References:

<https://os.mbed.com/handbook/mbed-tools>

<https://os.mbed.com/docs/mbed-os/v5.15/reference/mbed-os-build-rules.html>

<https://os.mbed.com/teams/SDK-Development/wiki/Mbed-SDK-build-script-introduction>