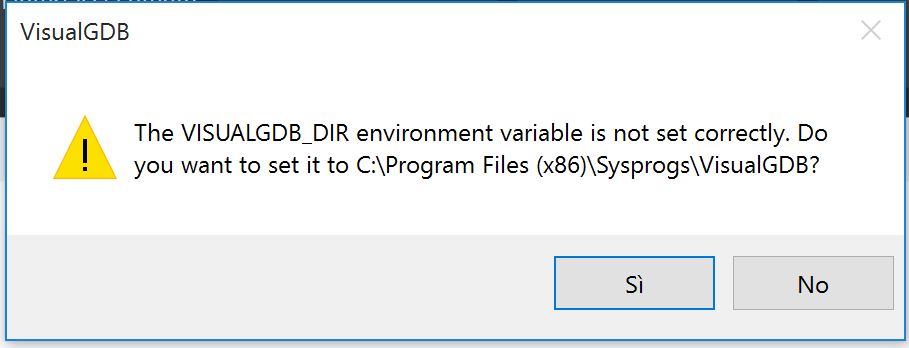
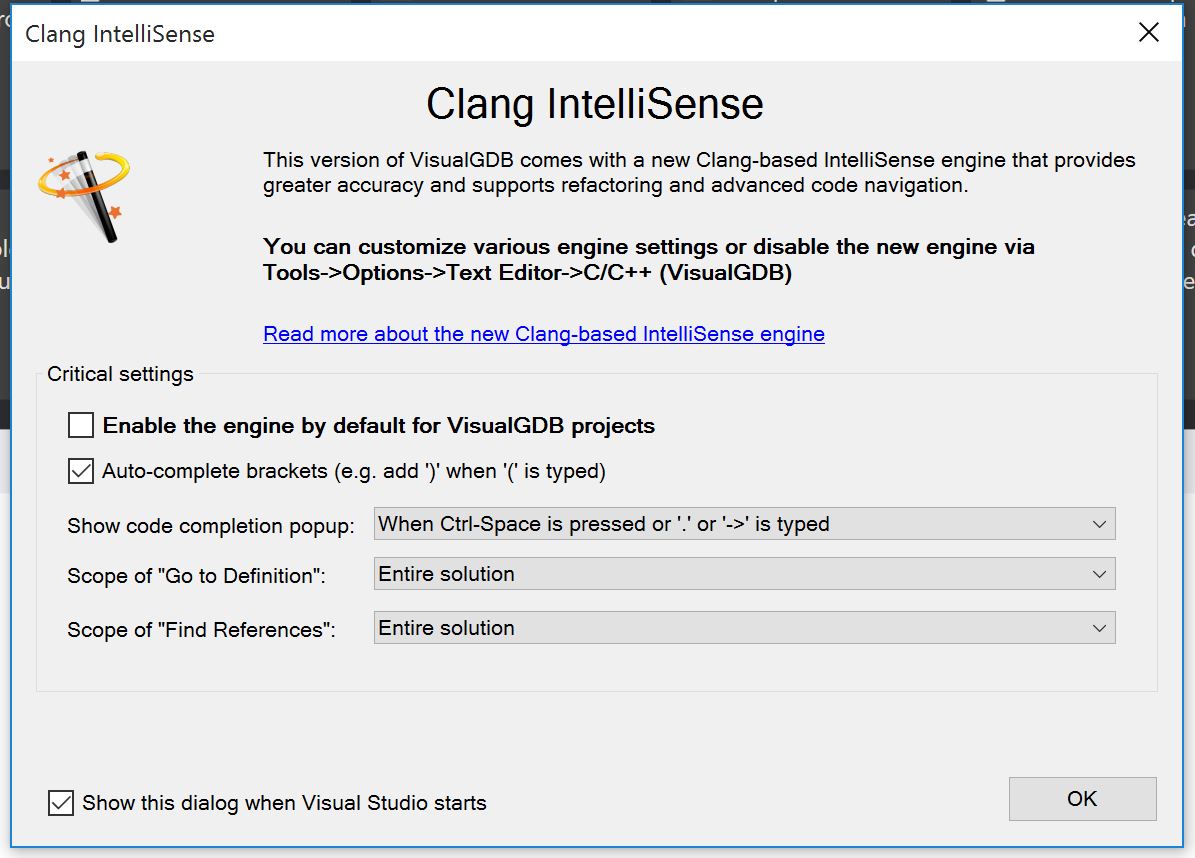
Getting started with Visual Studio, VisualGDB and Miosix

# Installing VisualGDB, Miosix

1. First of all, make sure to have installed Visual Studio, the supported versions are 2005-2015, exception made for the Express Editions that are not supported. If you need Visual Studio, you can download the 2015 Community Edition, that is free.
2. Let’s get the environment ready for Miosix. Go to <https://miosix.org/wiki/index.php?title=Windows_Quick_Start> and complete the steps “Before you begin”, “Install the Miosix Toolchain”, “Get the Miosix kernel sources”. Now enter the miosix-kernel folder and complete the “Configuring the kernel” step. If you work with many boards, you would better clone the kernel many times in different folders (one folder for each board) and repeat the “Configuring the kernel” step for each folder. In the file miosix/config/Makefile.inc please make sure that all variables OPT\_OPTIMIZATION are commented out.
3. Close Visual Studio if it’s open.
4. Download VisualGDB from <http://visualgdb.com/download/> and install it by following the wizard.
5. Open Visual Studio
6. The following prompt will appear:

Answer Yes to continue.

1. Another popup will notify you of the operation success and ask to restart Visual Studio. Do it.
2. Another VisualGDB screen will appear asking for your license details. For the moment you can start a trial (full featured for 30 days) or enter your license key.
3. Now you are prompted to choose wheter to use the VisualGDB-provided IntelliSense or continue to use the default Visual Studio Intellisense.

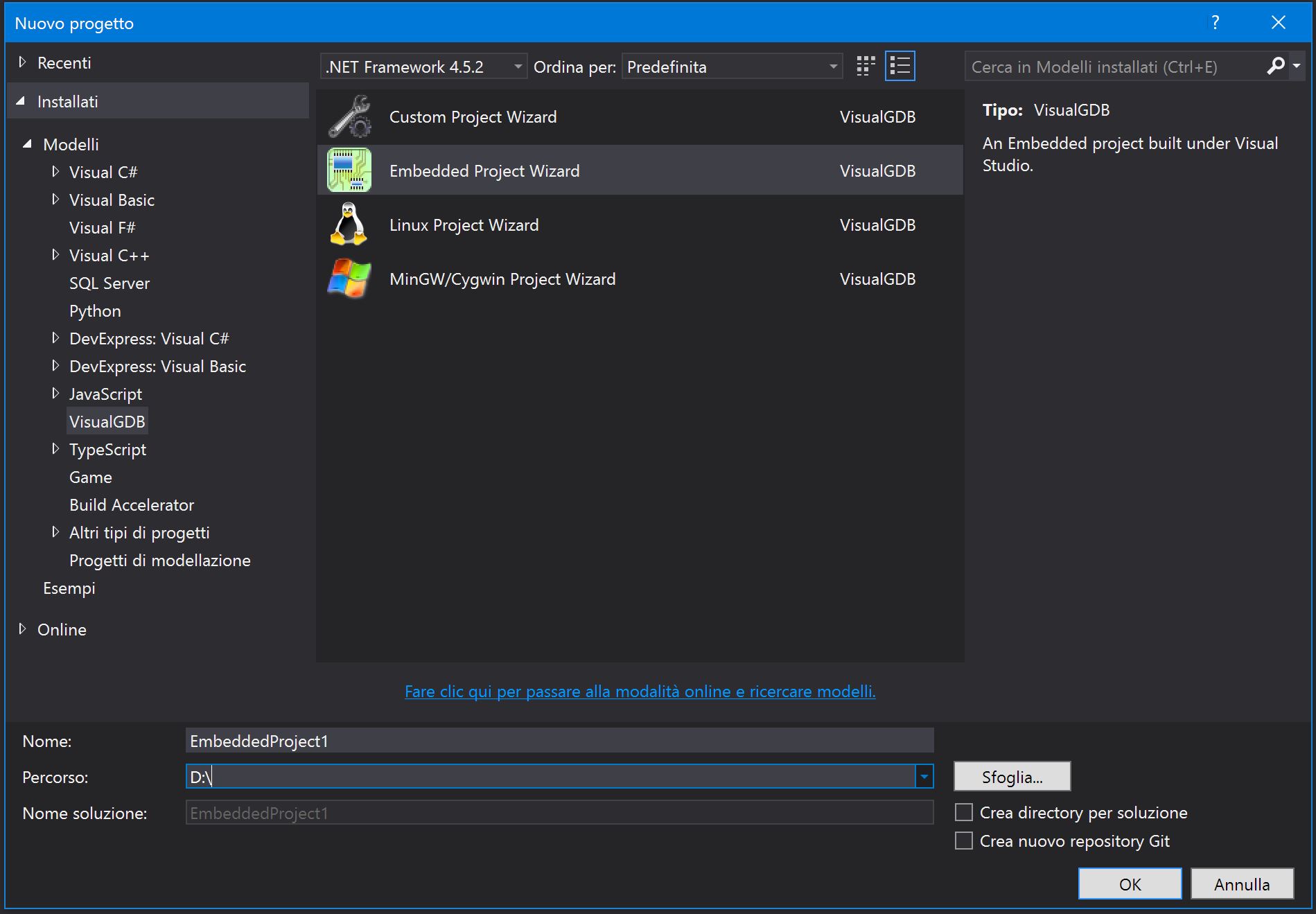


I recommend the Visual Studio IntelliSense. Uncheck the checkbox “enable the engine..” and the “Show this dialog when VS starts” one too.

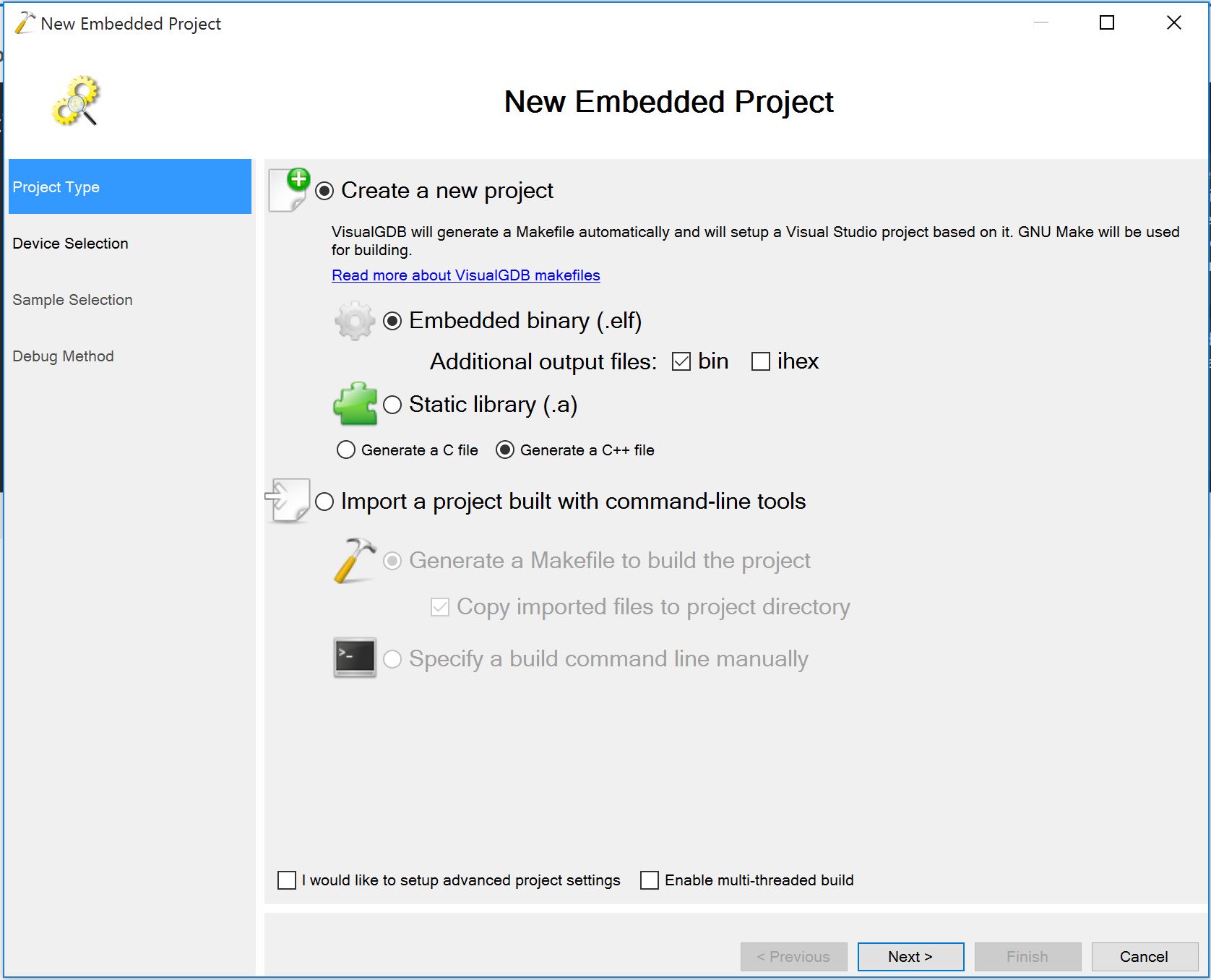
# Creating a new project

You have to repeat this walkthrough every time you would like to create a new project based on miosix. If you feel that the procedure is too long, you can make a project stub for each platform/board you work with, and then reuse it as many times you want by copying it.

1. In Visual Studio, choose File -> New -> Project. Pick VisualGDB from the project templates and then select “Embedded Project”.

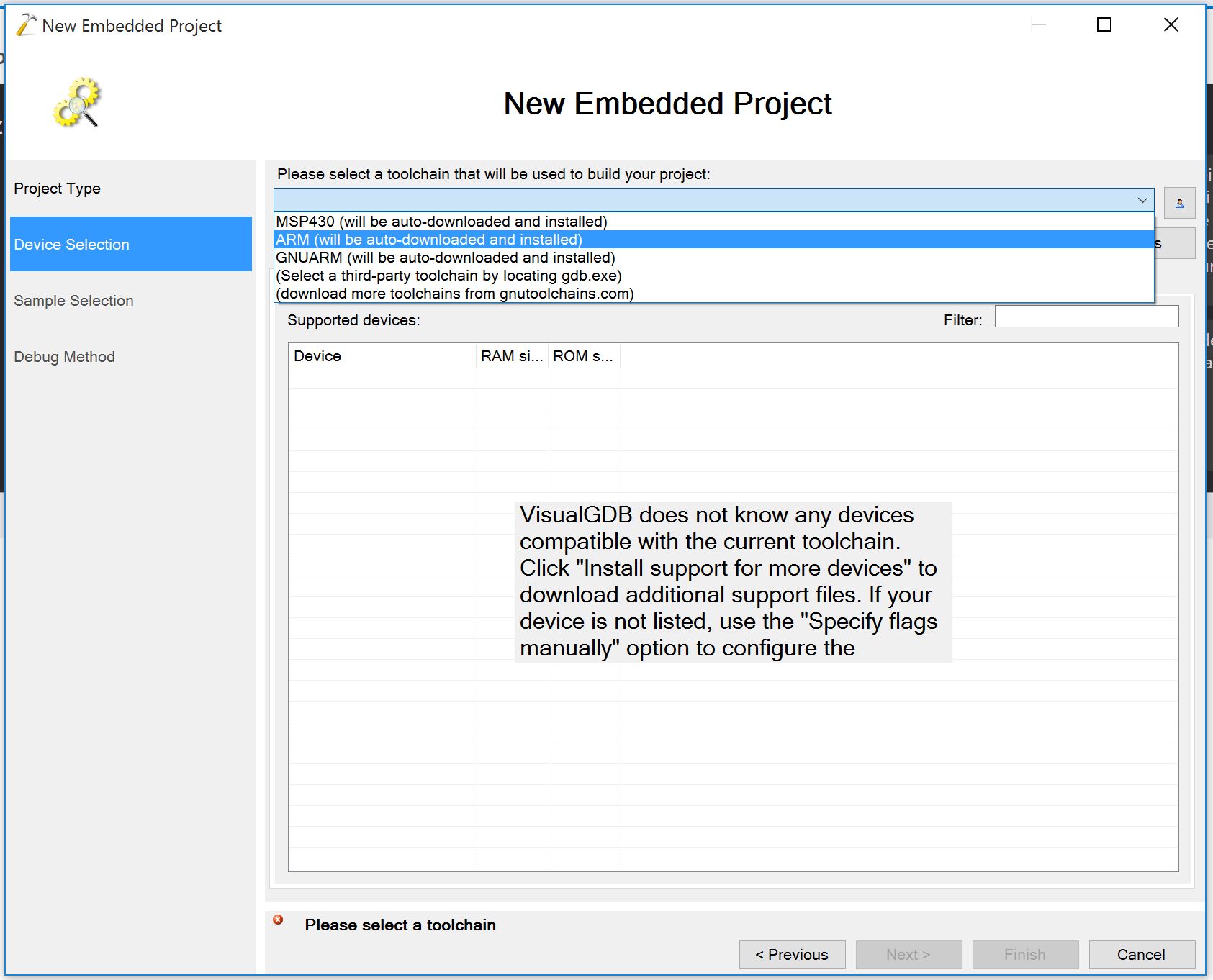


1. If the project path contains spaces, you are warned about it. I strongly recommend to put the project in a path that doesn’t contain spaces.
2. The VisualGDB wizard appears.

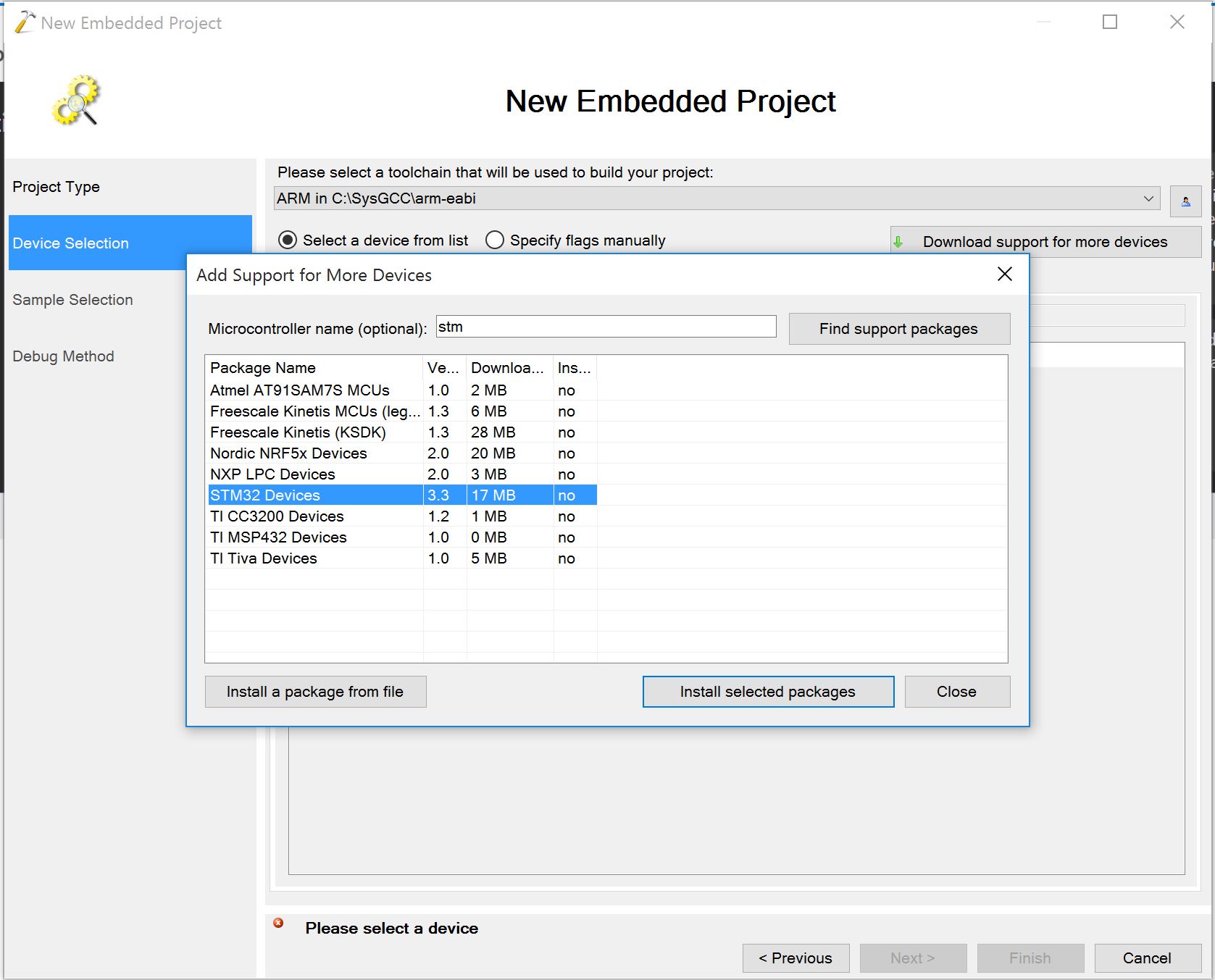
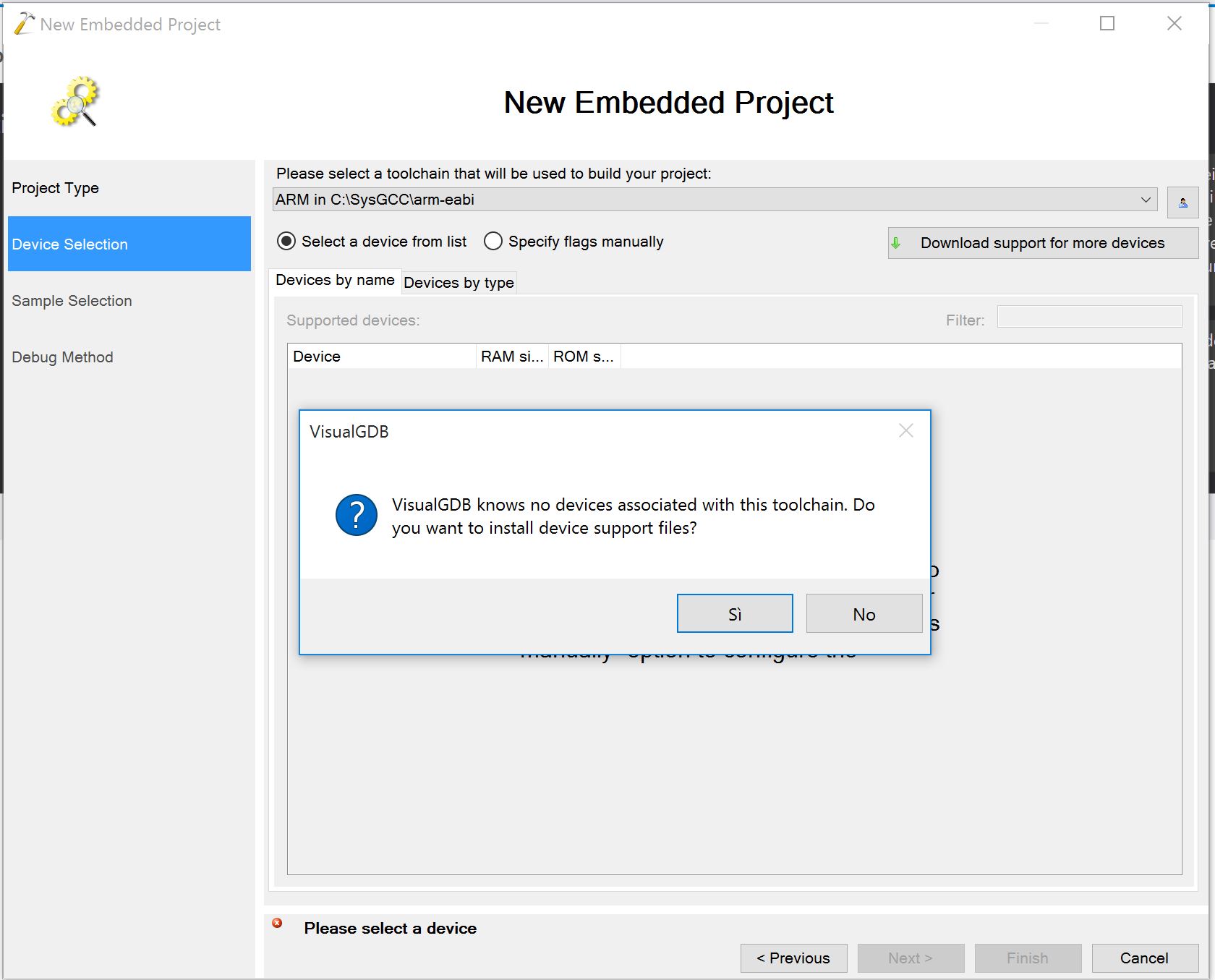


Here you can choose the desired output files. They will be put in the bin folder of the project, as for all other VS projects. Those files will be useful to you only if you need to send or publish them, since VisualGDB automatically deploys the project to the board when you run the project. Hit Next.

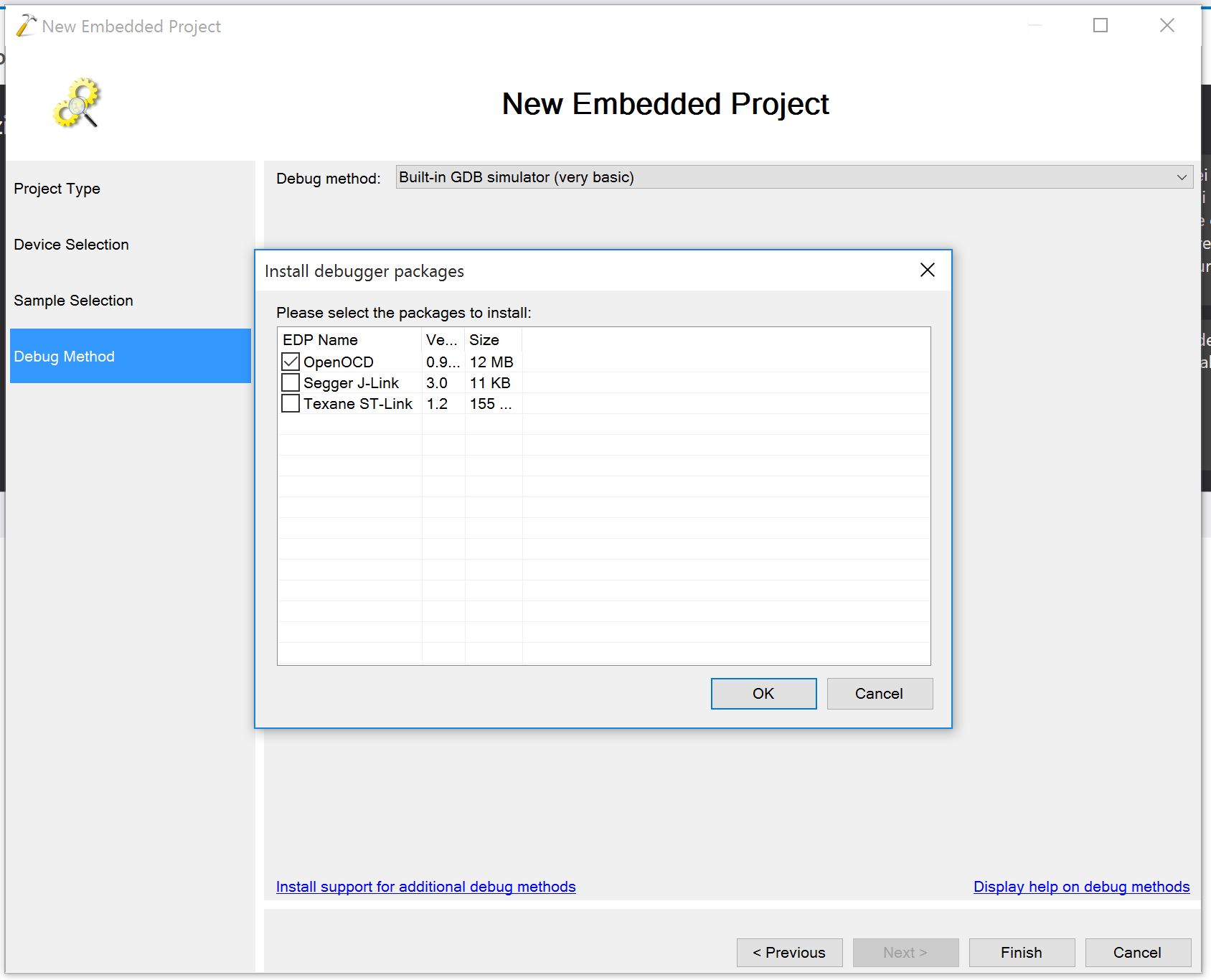
1. The Device Selection Screen appears. Before searching for your SoC from the list, you have to download the ARM toolchain. Due to changes that we will make later, that toolchain won’t be used to compile your project (Miosix toolchain will be used instead) but only to debug it. This has to be done only one time.

Select ARM from the dropdown list and proceed with download after you have selected the latest version.

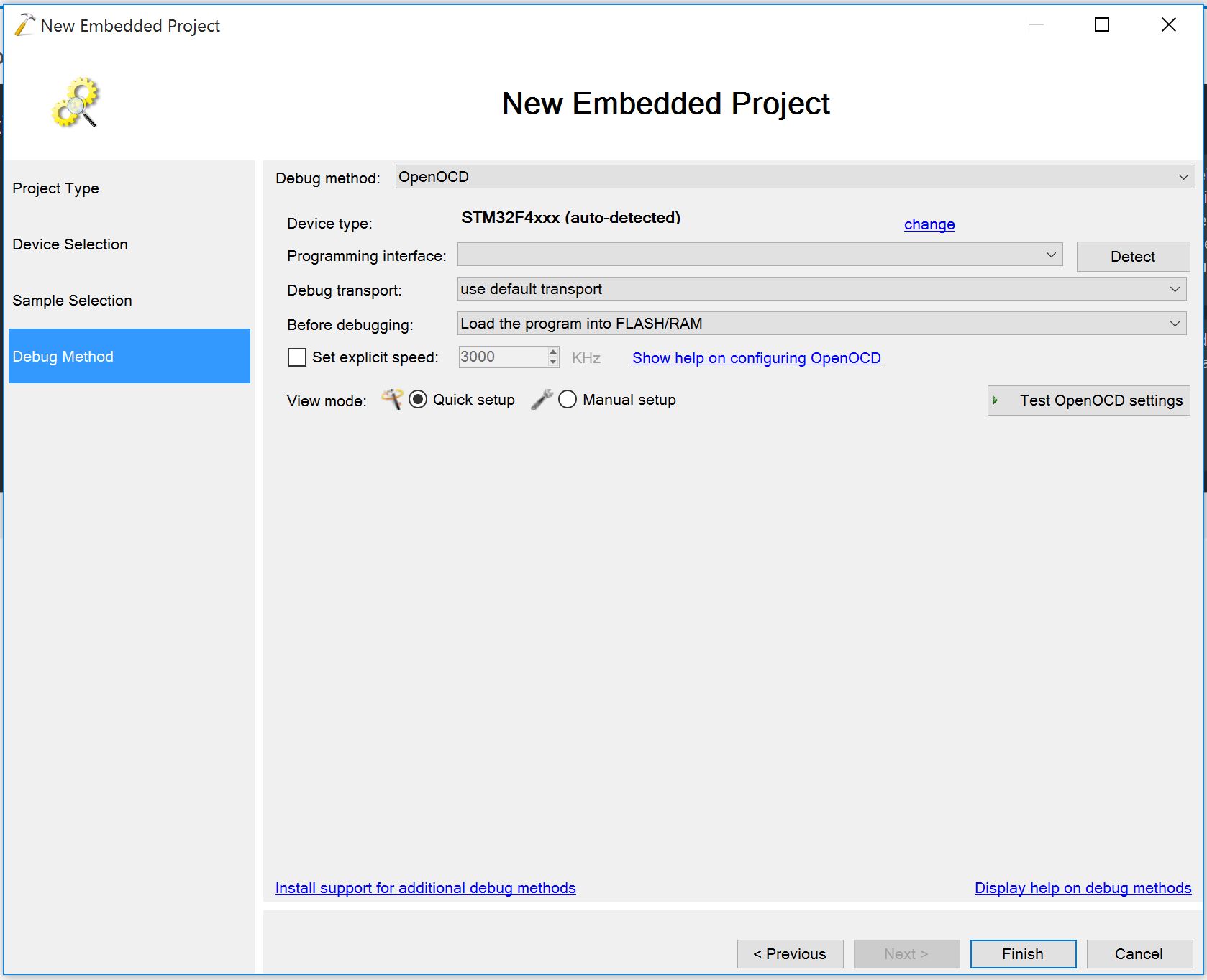
Once completed, you will be asked to install device support files. Answer yes and download the support package you need. Now select your device from the list and hit next. Note that you will need to select the model of SoC on which your board is based and not the model of the board itself.



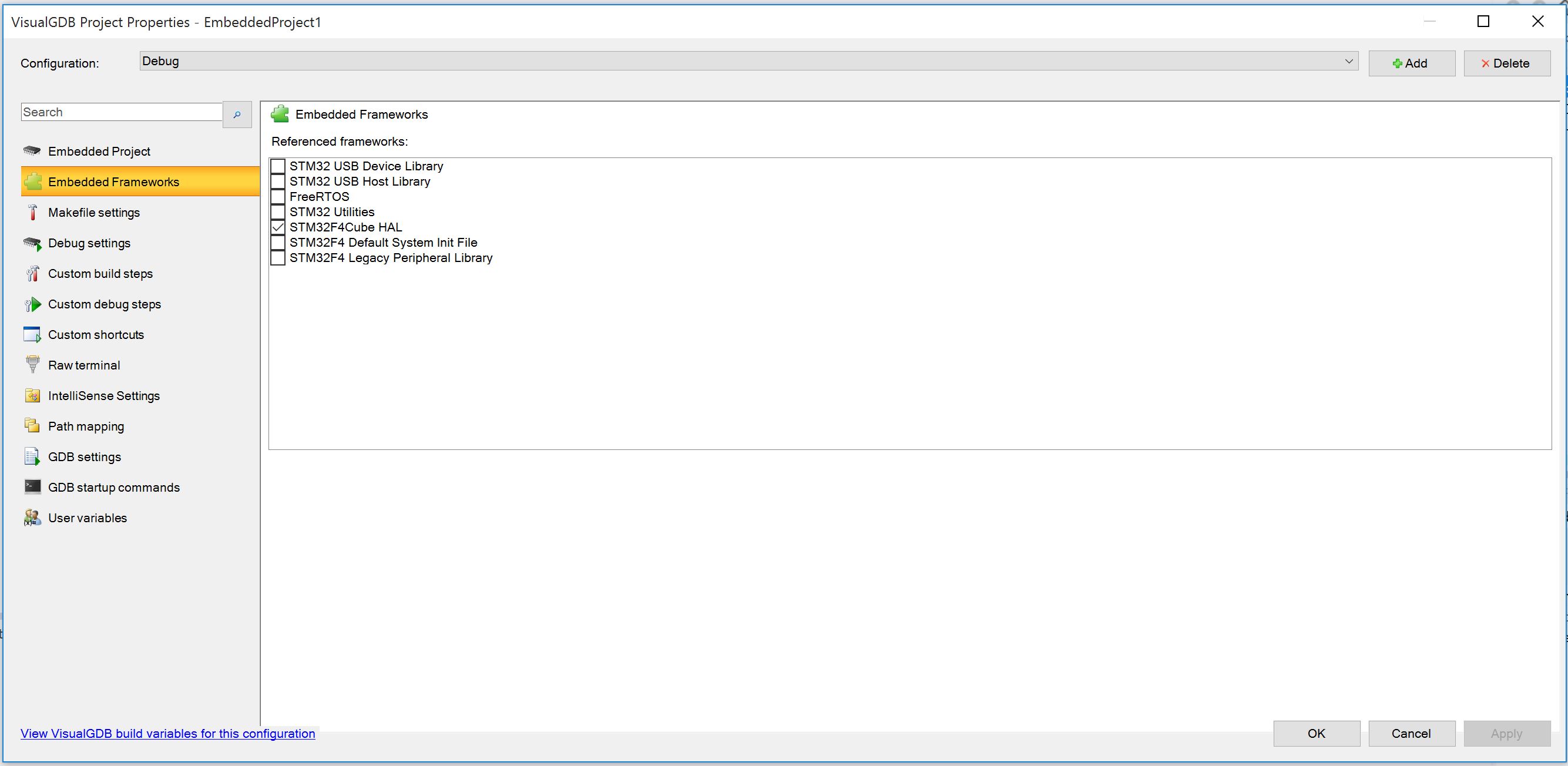
1. Leave the settings of the Sample Selection screen to their defaults, since we will delete all those files. Continue to the next screen.
2. Now it’s time to connect your board to the computer and verify that VisualGDB can connect to it. Start by installing OpenOCD by pressing “Install support for additional debug methods” at the bottom of the screen, selecting OpenOCD and clicking OK. You can also install other debuggers if you need them.



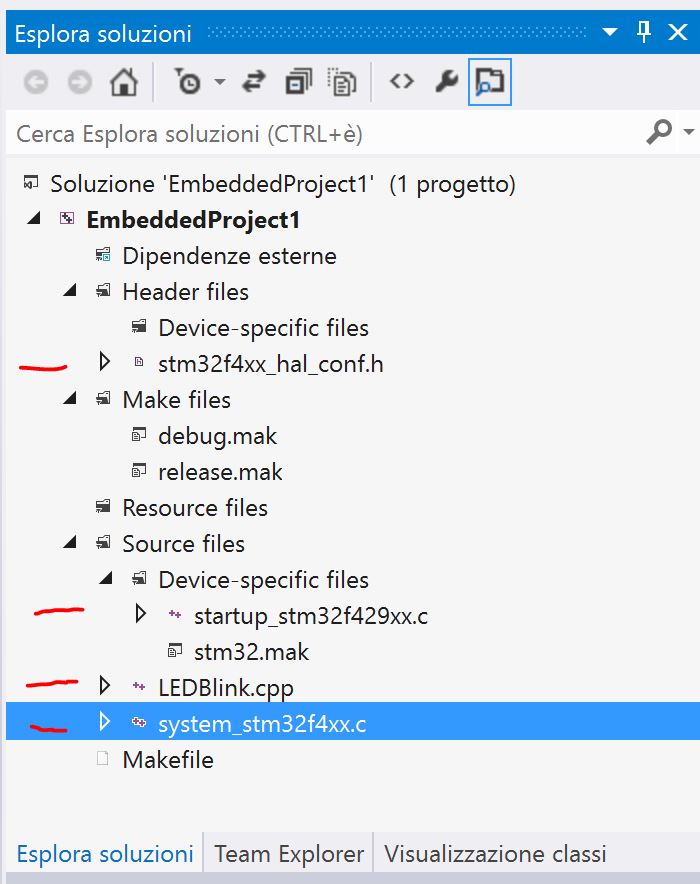
Now select from the top dropdown list “Debug method” OpenOCD.

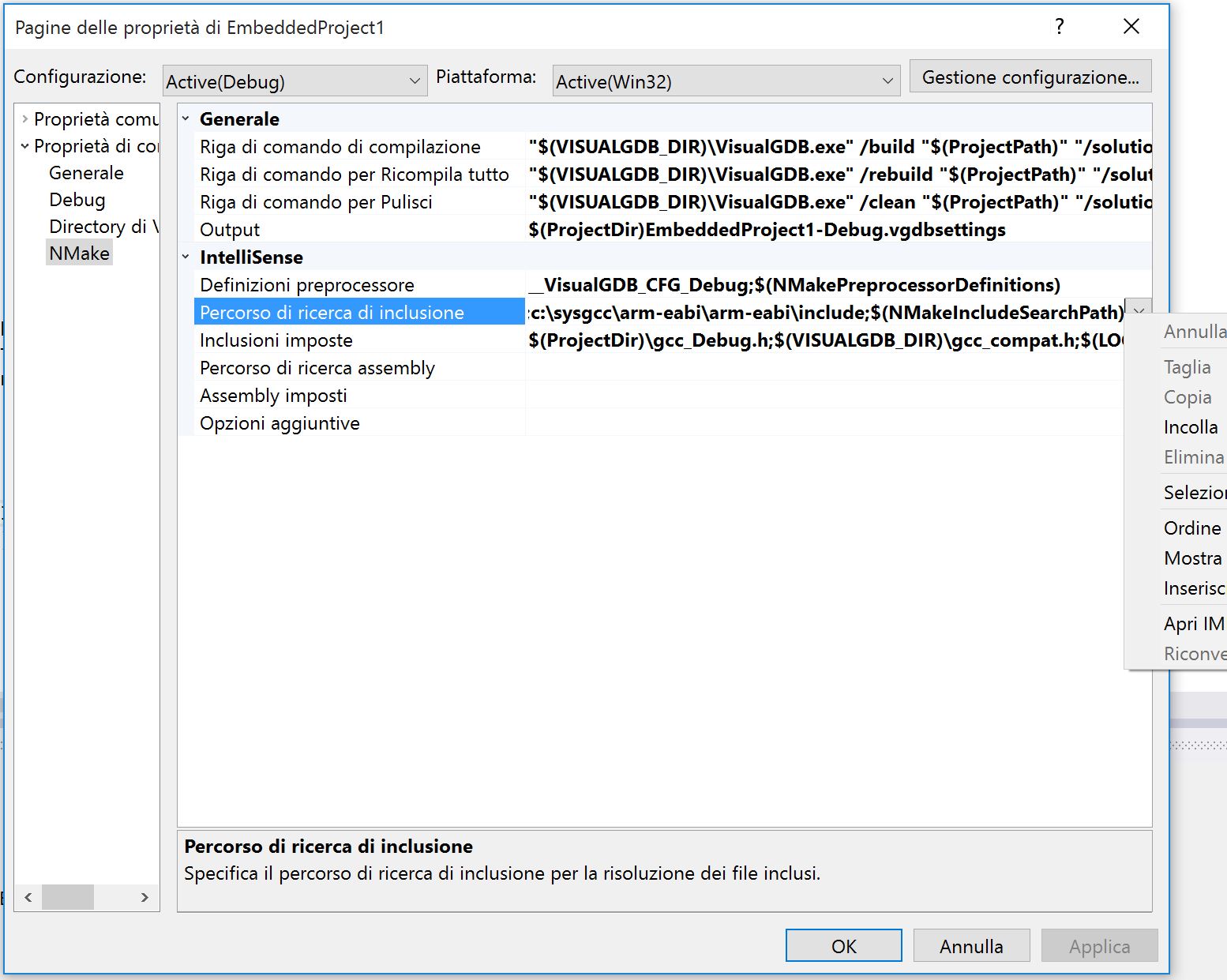
Press “Detect” on the right and VisualGDB will try to connect to your board and configure OpenOCD. Test if all is working correctly by pressing “Test OpenOCD settings”. You can click finish and wait for the project to be created. We are not done yet.

1. In Visual Studio, once the project is loaded, right click the project in Solution Explorer and choose “VisualGDB Project Properties”. In the window that appears, choose Embedded Frameworks from the left and then uncheck “STM32F4Cube HAL”.

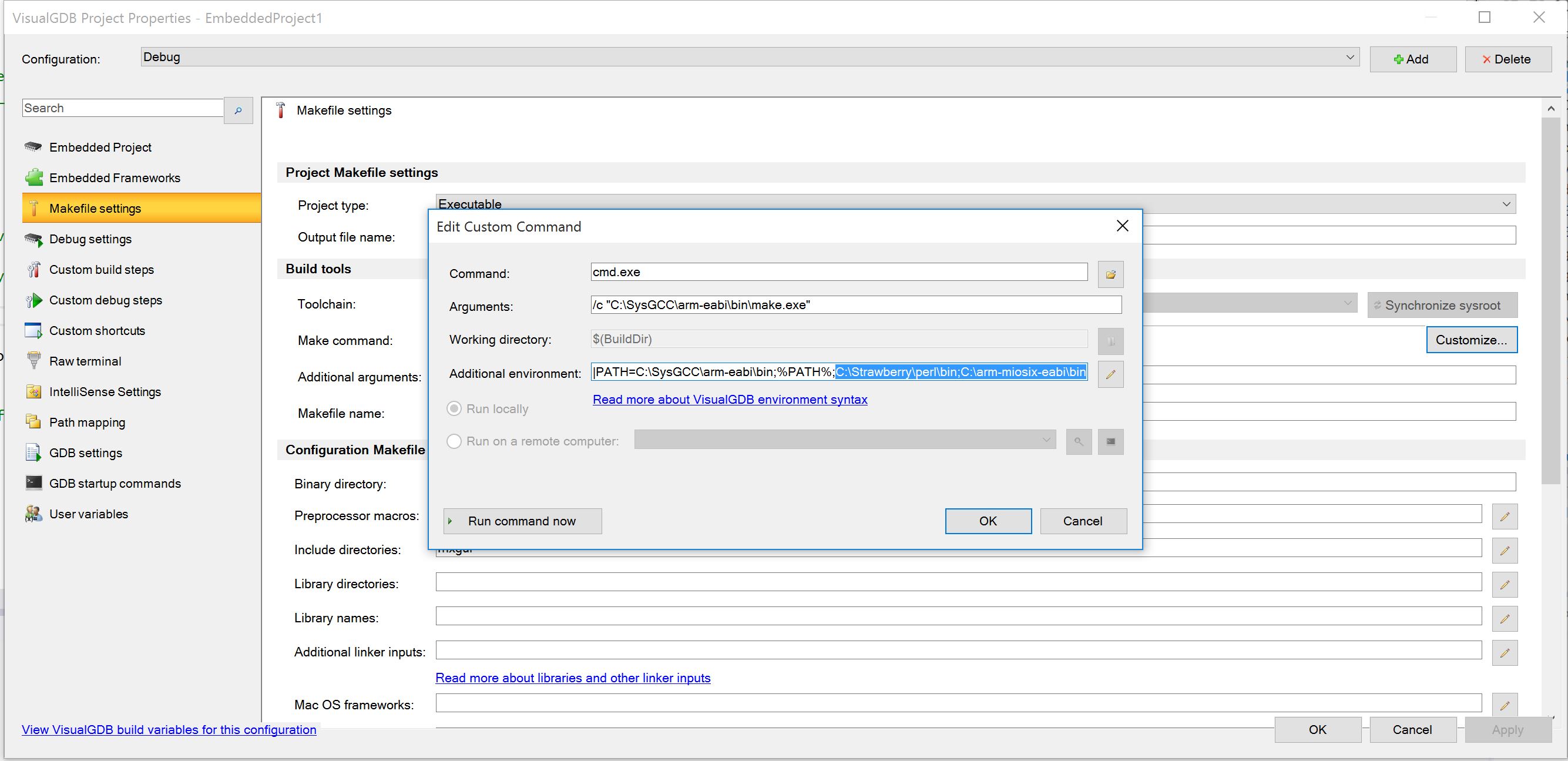


Apply the changes and close the window by pressing OK. Go back focusing the Solution Explorer panel. Delete from the project the files system\_stm32f4xx.c, stm32f4xx\_hal\_conf.h, startup\_stm32f429xx.c, LEDBlink.cpp.



1. It’s time to integrate Miosix in the project. Open the project in Windows Explorer and copy to the project folder the miosix folder (and not the miosix-kernel folder) you got in step 1.2. Now replace debug.mak, release.mak and Makefile in the project folder with those provided with miosix kernel.
2. We are almost ready. Now let’s make the IntelliSense working by specifying the correct miosix folders where to look for .h files. Open a command prompt in the project directory (just right click the project in VS, choose Show in Explorer, and from there right click while pressing SHIFT -> Open command prompt here) and type make print-includes . Select the includes printed and press enter to copy the text. Now go back to VS, right click the project, choose Properties (last entry) and then Configuration properties -> NMake. At the very end of Inclusion file search path, paste the includes you have copied before. 
3. Lastly, we need to tell VisualGDB where to find Perl and the Miosix toolchain. If you haven’t those directories on the PATH variable of your system, or the compilation phase complains about executables not found, proceed with this step, otherwise skip.

Right click the project in Solution Explorer, choose “VisualGDB Project Properties”. In the window that appears, choose Makefile Settings from the left and then, under “Build Tools”, press Customize at the right of “Make command”. Now concatenate to PATH=C:\SysGCC\arm-eabi\bin;%PATH% the paths of your Perl installation and Miosix toolchain installation. You should end up with something like: PATH=C:\SysGCC\arm-eabi\bin;%PATH%;C:\Strawberry\perl\bin;C:\arm-miosix-eabi\bin



1. Create a new main.cpp file and you are ready to go! If you like, you can copy to the root of the project the main.cpp file provided with miosix, and include it in the project to get started quicker. To do so, once you copied the file, choose “show all files” in Solution Explorer, then right click main.cpp and select “Include in the project”.

# Customizing compilation

You can use the release.mak and debug.mak Makefiles to tune the compilation process for both Debug and Release configurations.

Use TOOLCHAIN\_ROOT to specify Miosix toolchain location.

Use the INCLUDE\_DIRS, LIBRARY\_DIRS, LIBRARY\_NAMES, STATIC\_LIBS variables to specify your additional libraries. Includes for mxgui library are already present, remove them if you don’t plan to use that library.