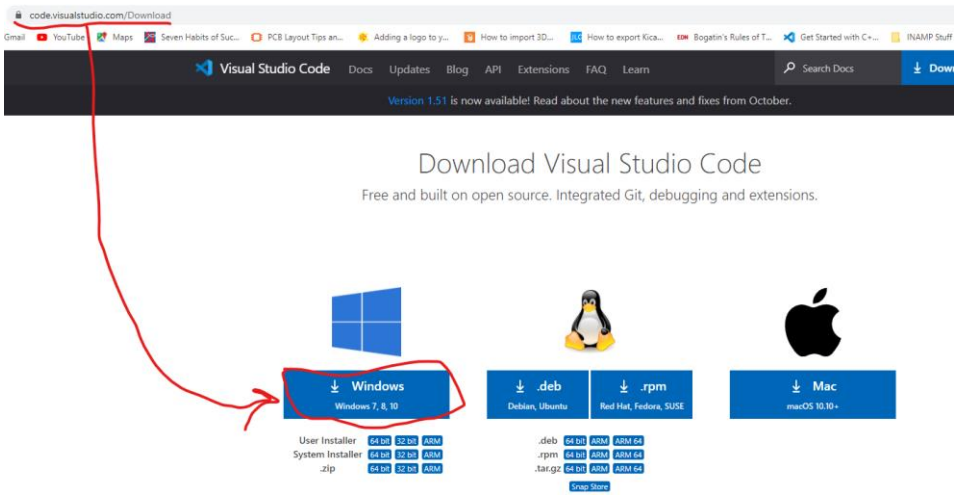
Setting up ARM Cortex M4 Firmware Development Toolchain

For Visual Studio Code on Windows 10, 64 bit

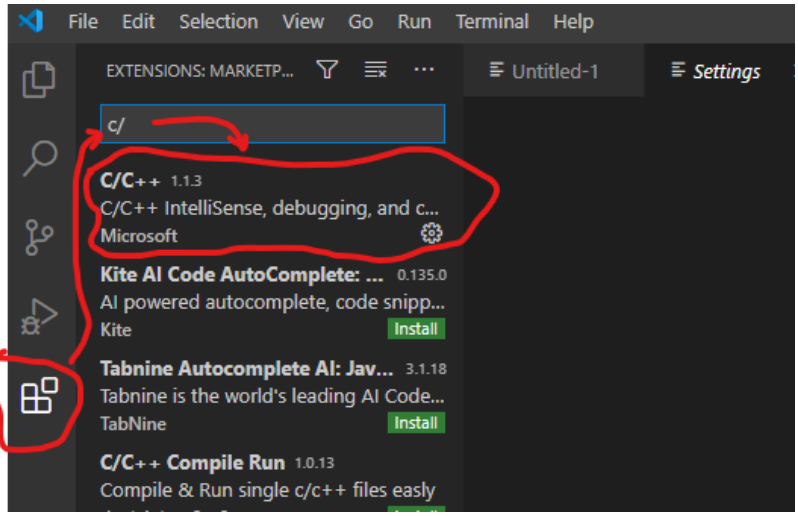
1) Install Visual Studio Code

Download Installer: https://code.visualstudio.com/Download



2) Install C/C++ extension in VSCode

Select the extensions button on the lefthand toolbar and search C/C++ and select the extension:

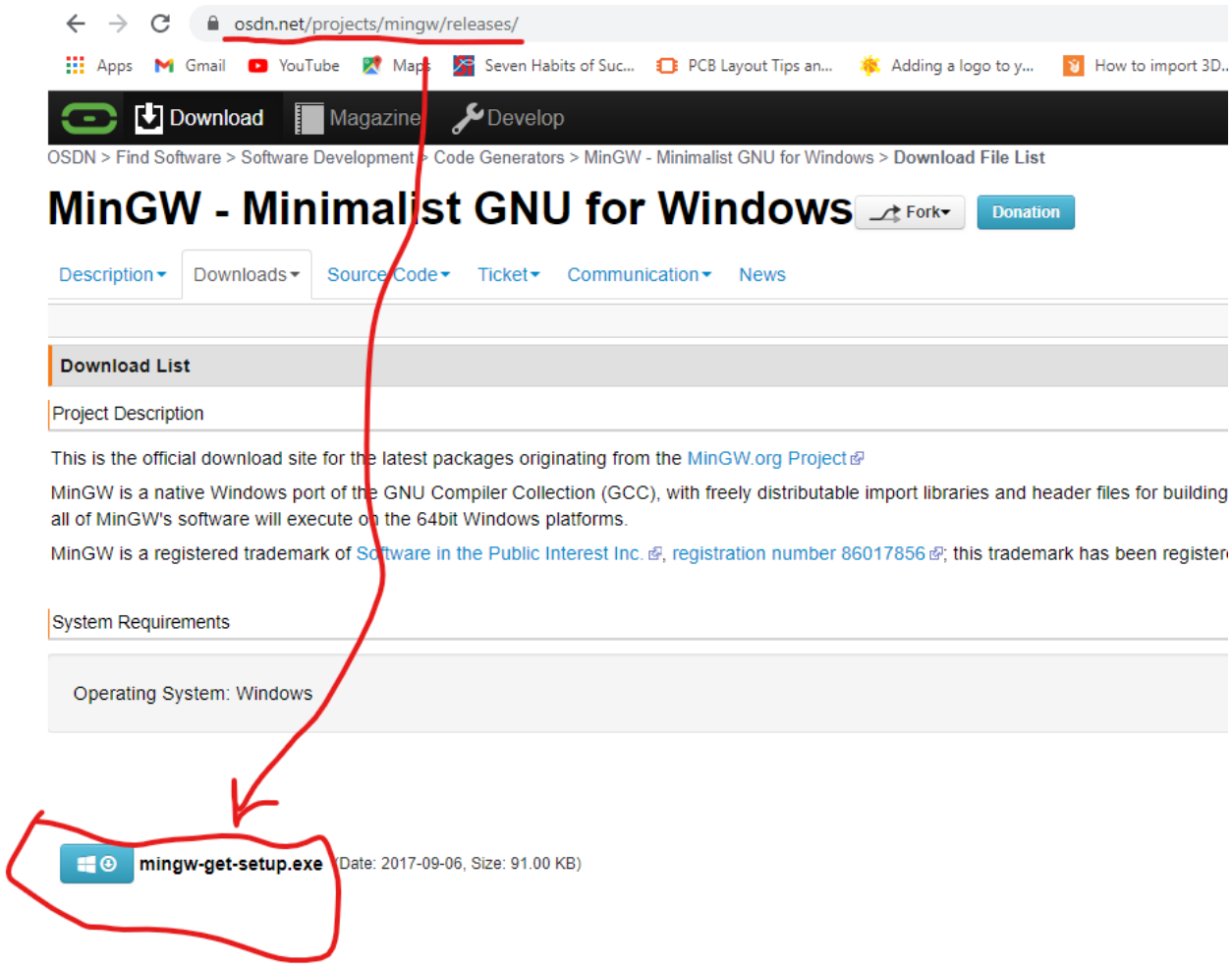


3) Install MinGW with MSYS

Description: https://code.visualstudio.com/docs/cpp/config-mingw

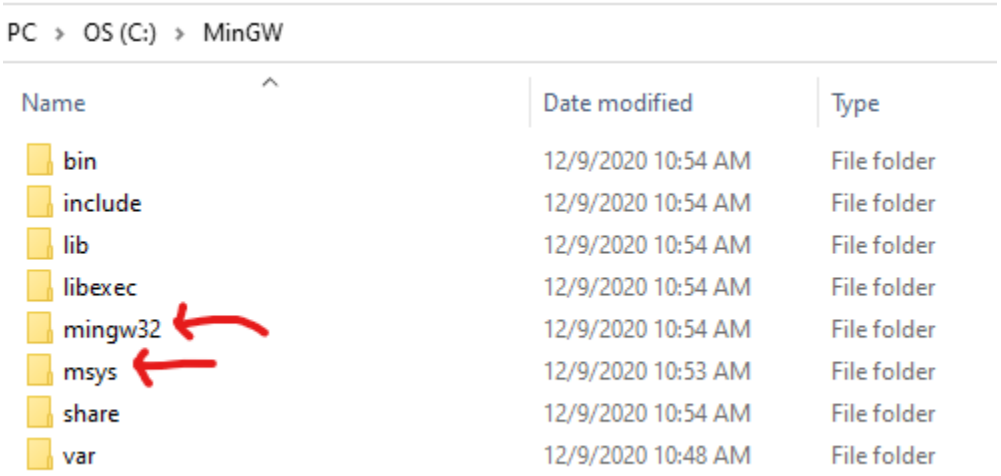
http://www.mingw.org/wiki/Getting_Started

Download installer: https://osdn.net/projects/mingw/releases/



Install to C:/MinGW for simplicity.

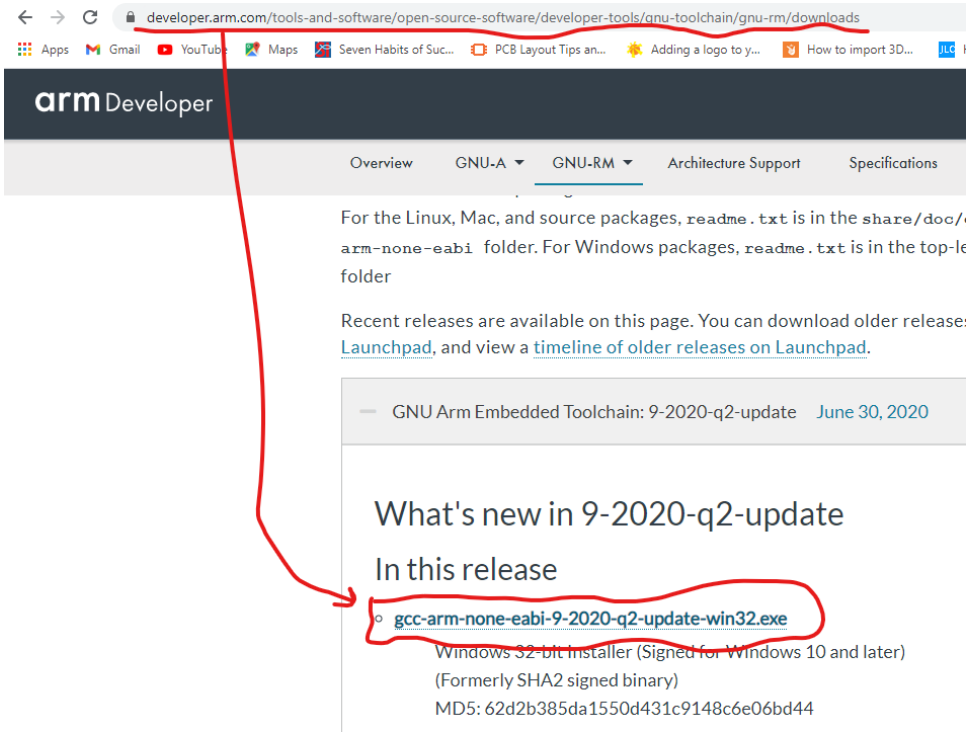
Check to make sure install includes both MinGW and MSYS.



4) Install ARM Cross-Compiler (arm-none-eabi-gcc)

Description

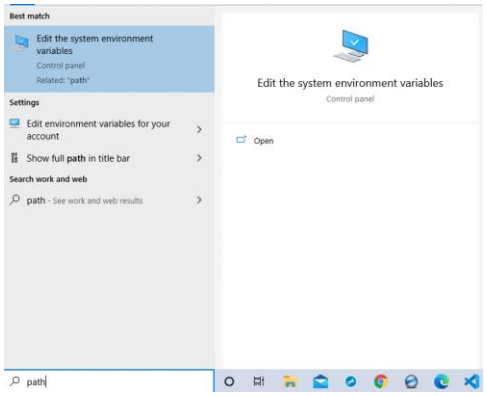
Download installer: https://developer.arm.com/tools-and-software/open-source-software/developer-tools/gnu-toolchain/gnu-rm/downloads



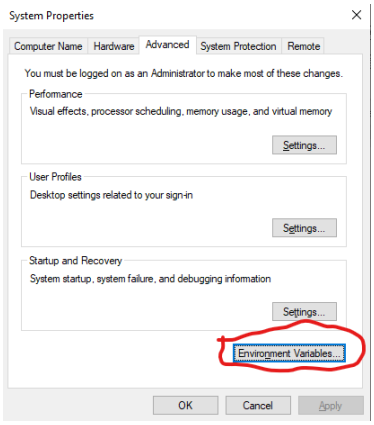
Install to default location – probably Program Files (x86).

5) Add environment variables to windows system path

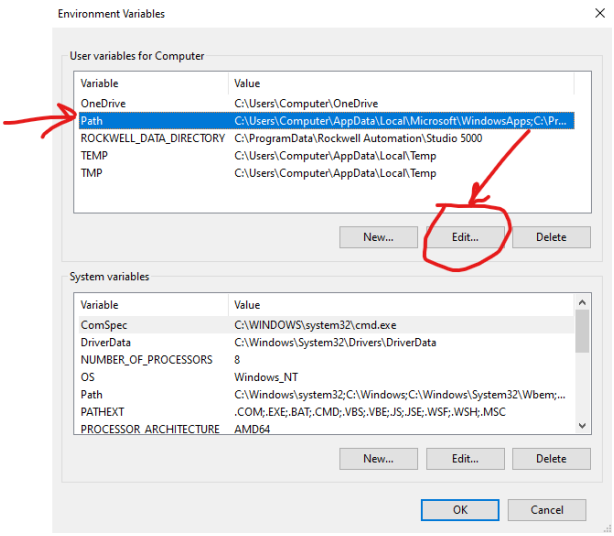
Type “path” into windows search bar, select “Edit the system environment variables”



Select “Environment Variables…” in the System Properties window that pops up



In the Environment Variables window, select “Path” then “Edit”



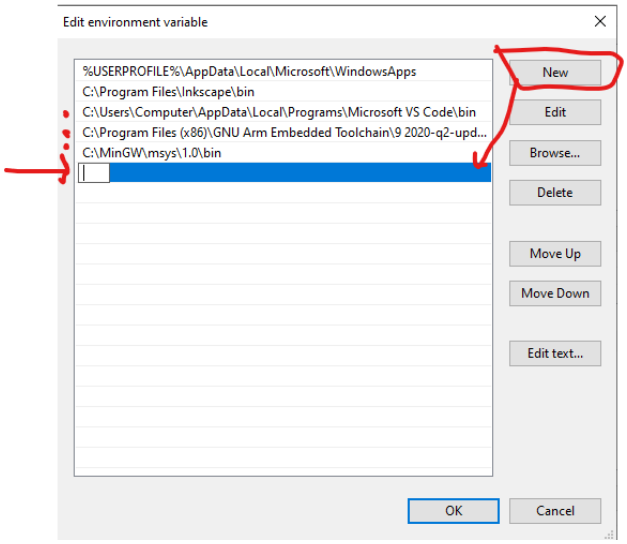
In the Path window select “New”. In the text field that opens, paste in the directory for one of the three .bin folders in VSCode, msys, and arm-none-eabi-gcc. Select “New” again and repeat for each of the three .bin folders.

Example:

C:\Program Files (x86)\GNU Arm Embedded Toolchain\9 2020-q2-update\bin

C:\Users\Computer\AppData\Local\Programs\Microsoft VS Code\bin

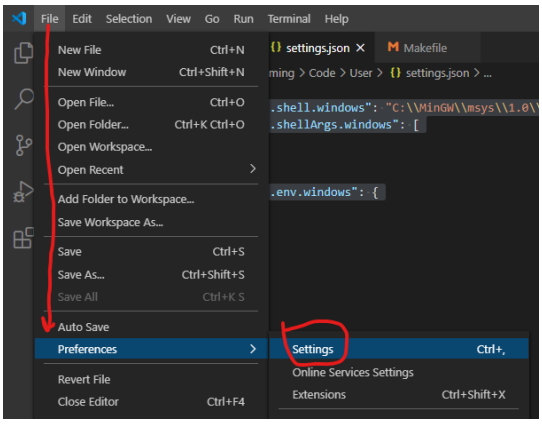
C:\MinGW\msys\1.0\bin



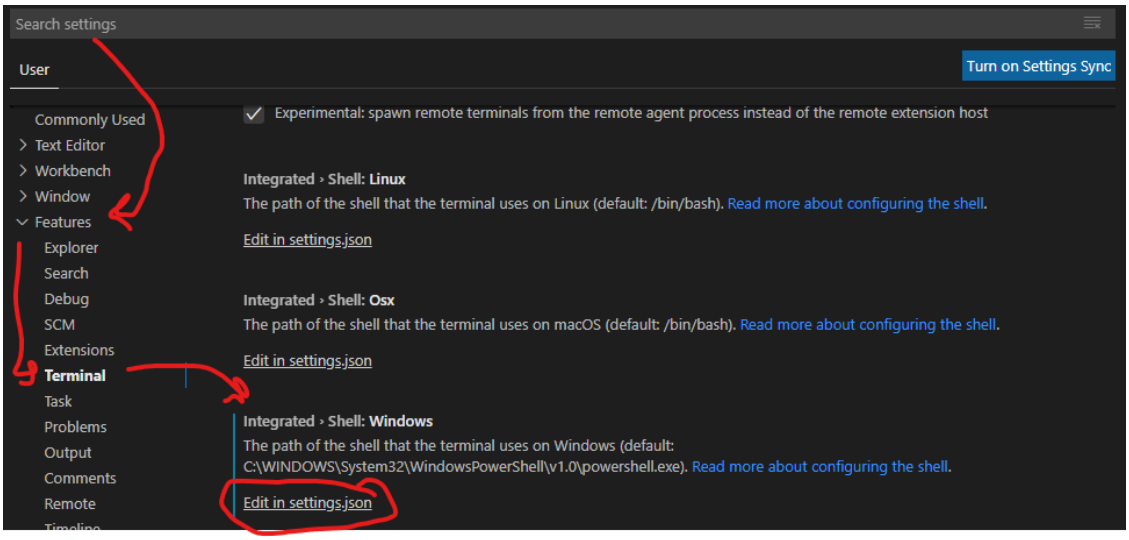
Click OK in all the windows to save your changes to the environment variables.

6) Add msys shell (sh.exe) to VSCode windows terminal

In VSCode, select file-preferences-settings:



In Settings, expand the “Features” pull down and select “Terminal”. Scroll down to “Integrated Shell: Windows” and select “Edit in settings.json”.

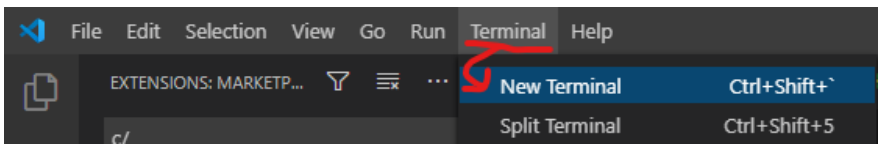


Copy and paste the following code into the settings.json file that pops up, overwriting everything. You may need to change your directory to sh.exe within msys. Note use of ‘\’ escape characters in path.

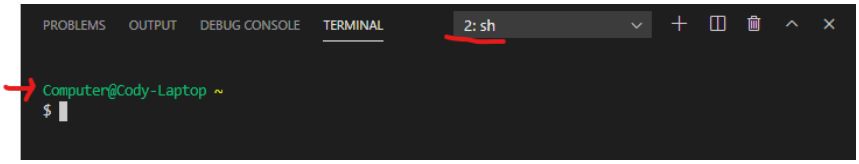
```
{
    "terminal.integrated.shell.windows": "C:\\MinGW\\msys\\1.0\\bin\\sh.exe",
    "terminal.integrated.shellArgs.windows": [
        "--login",
        "-i"
    ],
    "terminal.integrated.env.windows": {

    }
}
```

Test by opening a new terminal in VSCode:



The terminal that opens should be a “sh” terminal that opens in VSCode (not a separate window), with a prompt similar to the following:



7) Create Project Directory (no spaces in folder names!)

Make project folder in root directory (C:/) for simplicity.

Example: C:\VSCode_Projects

Within this directory, create your project folders, for example C:\VSCode_Projects\MyFirstProject

In the VSCode terminal,

Type “cd” to change the directory that the terminal is working in.

Example: “cd c:” changes the directory to C:/

Type “ls” to list all the folders and files in the current directory.

Type “cd ./” to extend from the current working directory.

Example: “cd ./VSCode_Projects” changes to C:/VSCode_Projects if C:/ is the current directory

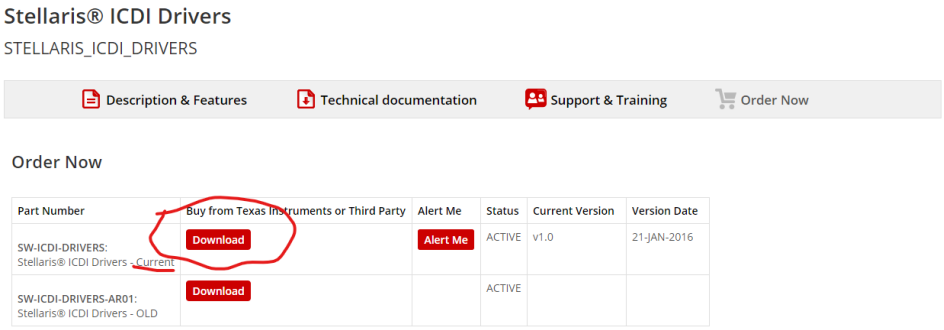
While typing file names in a command, hit Tab to auto complete if you have typed unique identifiers for a file name.

Example: “cd ./My” + Tab will autocomplete to “cd ./MyFirstProject”

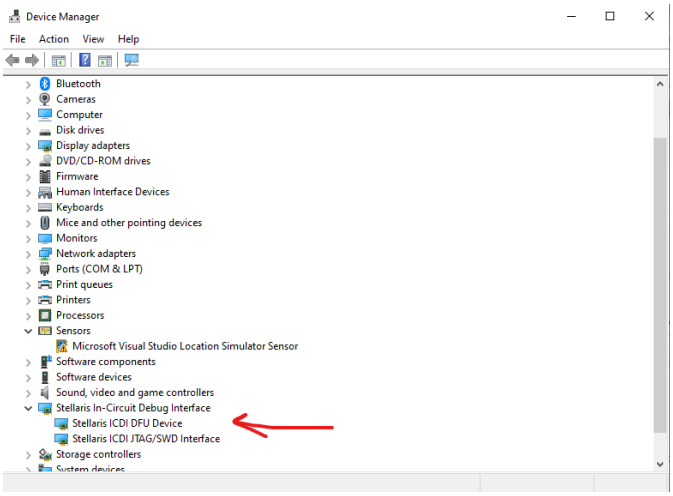
Once in your project directory, and your file is ready to compile with a Makefile, type “make” to compile all your code into a binary file for the ARM Cortex microcontroller!

8) Install Device Drivers

Stellaris Dev Board Drivers: https://www.ti.com/tool/STELLARIS_ICDI_DRIVERS



Extract the zip file and open Device Manager (type device manager into window search). Within Device Manager, find the Stellaris device drivers:



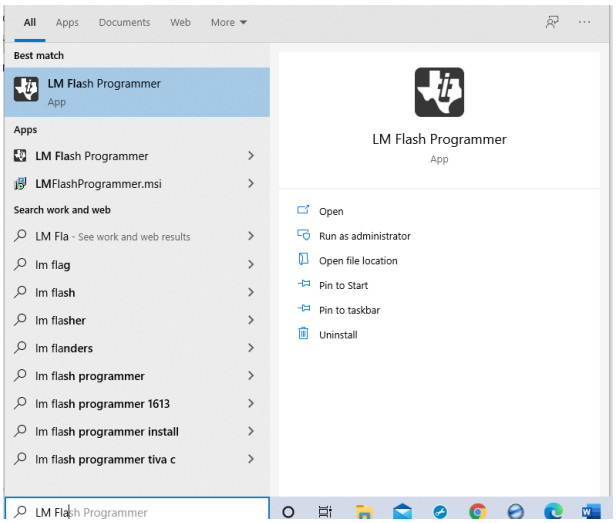
Right click on one of the stellaris devices and select “Update Driver Software…”. Select the option to browse for driver software and navigate to where you unzipped the drivers and select next.

9) Program Binary File to Microcontroller

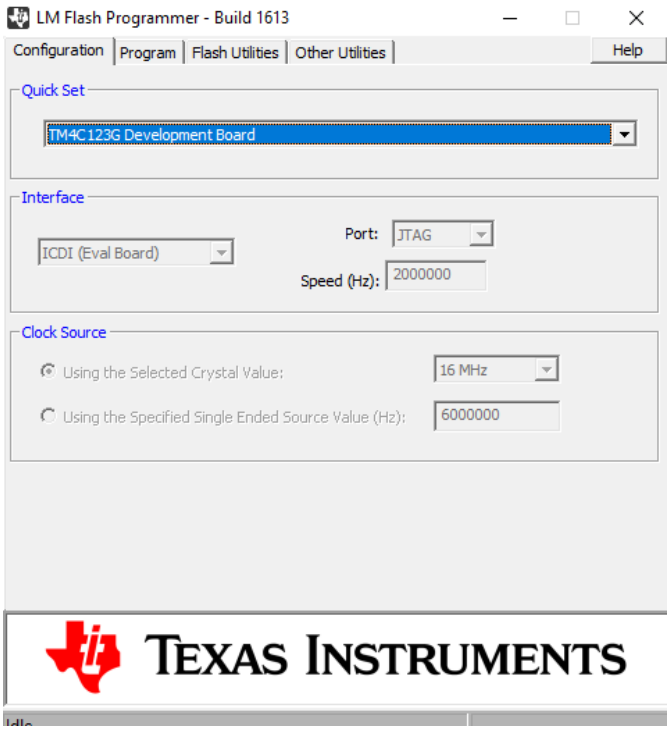
Download the Texas Instruments LM Flash Programmer:
https://www.ti.com/tool/LMFLASHPROGRAMMER

You'll need a TI account, and you'll have to go through a security questionnaire that makes sure you're not making weapons for overseas.

Download and install the program. Once installed, Type LM Flash into windows search to access the program:



Select the appropriate microcontroller in the Configuration tab:



In the Program tab, select the options “Erase Entire Flash”, “Verify After Program”, and “Reset MCU After Program”. Then select your newly compiled .bin file and click “Program”. The program should now be written to the microcontroller! The LM Flash Programmer will display it’s status at the bottom of the window in the grey bar.