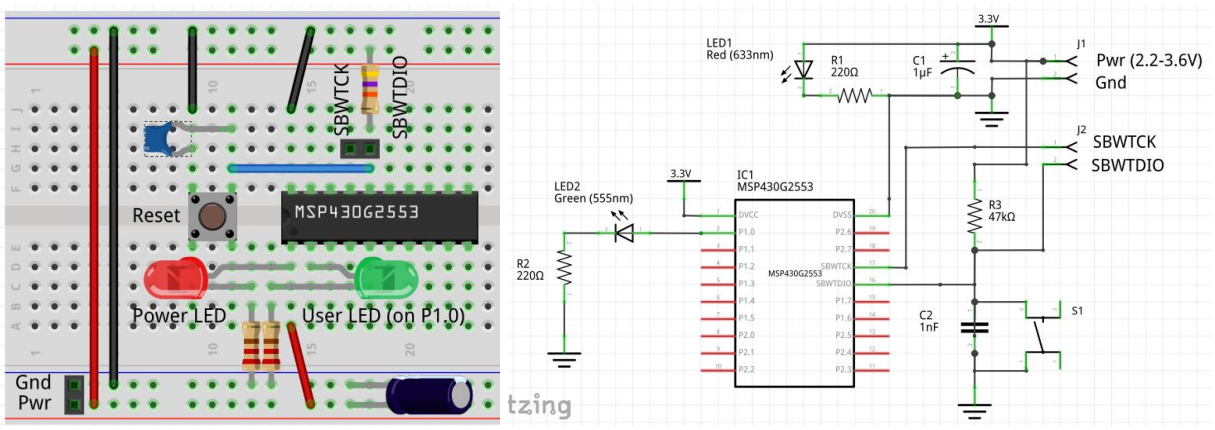
Getting Started with the MSP430

Step 1: Pick an MCU



Figures 1&2: Breadboard layout and schematic for the minimum required circuit to program the MSP430G2553. Images were created using Fritzing; the original Fritzing file is available here[1].

- **Options: All MSP430 MCUs.**
  - I bought the MSP430G2553 (the same one that comes with the Launchpad MSP-EXP430G2ET development board).

[1] https://github.com/nathancharlesjones/Embedded-for-Everyone/tree/master/Getting-Started-Guides/MSP430

- Choose an IC in a DIP package so it will fit on a breadboard; otherwise, you'll need to solder the MCU onto a breakout board with the appropriate headers for it to fit on a breadboard.
- In trying to figure out how to breadboard the MSP430G2553 (the MCU that I tested with), I referenced the documents below. If you aren't prototyping with that exact MCU, the circuits are probably almost identical; compare the reference design schematics for your exact MCU with Figures 2-2 and 2-3 in the "MSP430 Hardware Tools User's Guide" to be sure.
    - "MSP430G2553 Launchpad Development Kit (MSP-EXP430G2ET) User's Guide"[2], pg 25 ("MSP430G2xxx Target" and "Pushbuttons" sections)
    - Reference design schematics for the MSP430G2553[3]
    - "MSP430 Hardware Tools User's Guide"[4], Figures 2-2 (pg 22) and 2-3 (pg 23)
        - Assumes the developer is using a more advanced debug adapter, such as the MSP-FET or the eZ430-2013 (see "Debug Adapter" below). Since we're using the ez-FET, we substitute "TDO/TDI" on the JTAG adapter for "SBWTDIO" on our ez-FET and "TCK" for "SBWTCK". We can also safely leave off any of the components that only connect to a JTAG-only connection.
- Schematic notes for the MSP430G2553:
    - Pin 1 (DVCC) should be connected to 2.2-3.6V for programming (1.8-3.6V for program execution).
    - Pin 20 (DVSS) should be connected to GND.
    - C1 is a 1 uF power filtering capacitor.
    - LED1 and R1 are a power indicator sub-circuit. The value of R1 should be adjusted for the desired brightness level, based on the supply voltage and the forward voltage of LED1.
    - LED2 and R2 are a user-defined output, to enable a basic "Blinky" program. The value of R2 should be adjusted for the desired brightness level, based on the supply voltage and the forward voltage of LED2.
    - R3 pulls the reset pin (Pin 16) high in the absence of another signal to keep the MSP430 from randomly resetting; 47 kOhms is the recommended value but I've successfully prototyped with values as low as 10 kOhms.
    - S1 and C2 (1 nF here; max value is 2.2 nF) are the reset button and a smoothing capacitor (to prevent switch bounce from triggering multiple resets in a row).

[2] http://www.ti.com/lit/ug/slau772/slau772.pdf
[3] http://www.ti.com/product/MSP430G2553/toolssoftware#TIDesigns
[4] http://www.ti.com/lit/ug/slau278ae/slau278ae.pdf

Step 2: Pick a debug adapter

- **Options:**
  - **ez-FET (on Launchpad MSP-EXP430G2ET, possibly others)**
  - **MSP-FET**
  - **eZ430-F2013 (for some MSP430 devices; see below for details)**
- TI offers many feature-rich debug adapters but the cheapest way to program an MSP430 is by using a Launchpad development board, which provides an on-board ez-FET debug adapter and programming pins that can be connected to an off-board MCU.
- ez-FET (on Launchpad MSP-EXP430G2ET, possibly others)
  - "Supports all programmable MSP430 and CC430 devices."[5]
  - "Fixed 3.3-V target supply voltage."[6]
  - "2-wire JTAG" ("The 2-wire JTAG debug interface is also referred to as Spy-Bi-Wire (SBW) interface").[7]
  - "The MSP-EXP430G2ET has the eZ-FET debug probe (see Figure 5), which is a simple and low-cost debugger that supports all MSP430 device derivatives."[8]
  - ez-FET Wiki[9] (referenced in the MSP-EXP430G2ET User's Guide)
  - Supports EnergyTrace (but not EnergyTrace++)[10]
    - EnergyTrace: Current monitoring, Supports all MSP430 MCUs
    - EnergyTrace++: Current monitoring, CPU State, Peripheral and System states, Supports MSP430FR59xx and MSP430FR69xx MCUs[11]
- MSP-FET: can also program all MSP430 devices, but it retails for $115.[12]

---

[5] "MSP Debuggers User's Guide", pg 4 (http://www.ti.com/lit/ug/slau647m/slau647m.pdf)
[6] "MSP Debuggers User's Guide", pg 4 (http://www.ti.com/lit/ug/slau647m/slau647m.pdf)
[7] "MSP Debuggers User's Guide", pg 4 (http://www.ti.com/lit/ug/slau647m/slau647m.pdf)
[8] http://www.ti.com/lit/ug/slau772/slau772.pdf
[9] http://processors.wiki.ti.com/index.php/EZ-FET_lite
[10] "MSP Debuggers User's Guide", pg 4 (http://www.ti.com/lit/ug/slau647m/slau647m.pdf)
[11] "MSP Debuggers User's Guide", pg 4 (http://www.ti.com/lit/ug/slau647m/slau647m.pdf) & "MSP430G2553 Launchpad Development Kit (MSP-EXP430G2ET) User's Guide", pg 8
[12] "MSP430 Hardware Tools User's Guide", pg 9 (http://www.ti.com/lit/ug/slau278ae/slau278ae.pdf)

- eZ430-F2013: can only program certain MSP430 devices (check the "MSP430 Hardware Tools User's Guide"[13] for more information), and it comes in at a more reasonable $25[14], though that's still about $10 more expensive than a simple Launchpad development board with on-board ez-FET.[15]

Step 3: Pick an IDE/code editor

- **Options (all are IDEs unless otherwise noted):**
  - **Code Composer Studio (CC Studio)**
  - **CCS Cloud**
  - **Energia**
  - **IAR Embedded Workbench**
  - **VS Code (with Platform IO; see "Middleware" below)**
  - **Your favorite text editor (NOT an IDE)**
- Code Composer Studio
  - Follow the installation instructions[16] for your operating system.
  - Not all devices are supported for Linux/Mac; see the installation instructions for your specific operating system above for more details.
  - Linux notes
    - You may need to install additional libraries PRIOR to installing CC Studio. Although the installation instructions above list which ones are needed, I still had a lot of trouble getting this program to work on my laptop. If you have trouble, uninstall/delete everything (including the hidden folder, ".ti") and try again. You could also try installing the libraries which were previously needed for older versions of CC Studio; I found the Linux support pages for CCSv5[17] and CCSv6[18] to be useful.
    - If you still have trouble with missing dependencies, the wiki called Checking Linux Dependencies for CCSv5[19] hosts a script that you can download which checks for missing dependencies. Make sure to point the "generate_depends.sh" script to your specific CCS file and to run that script before running the "check_depends.sh" script.
    - During a few installations, the CC Studio executable did not have the proper permissions to be run as such on my laptop. By right-clicking on that file, going to the "Permissions" tab, and checking the box that read "Allow this file to be run as a program", I was able to change that and launch the application.

[13] http://www.ti.com/lit/ug/slau278ae/slau278ae.pdf
[14] http://www.ti.com/tool/EZ430-F2013#buy
[15] "MSP430 Hardware Tools User's Guide", pg 9
[16] http://software-dl.ti.com/ccs/esd/documents/ccs_downloads.html
[17] http://processors.wiki.ti.com/index.php/Linux_Host_Support_CCSv5
[18] http://processors.wiki.ti.com/index.php/Linux_Host_Support_CCSv6
[19] http://processors.wiki.ti.com/index.php/Checking_Linux_Dependencies_for_CCSv5

        - The first time CC Studio opened, it prompted me to perform several updates. However, after performing these updates, CC Studio would need to restart and then wouldn't open again. I was only able to get around this by refusing the initial updates, closing and reopening CC Studio, and only then performing the necessary updates.
    - When CC Studio opens, navigate to the "Getting Started" page (click "View" > "Getting Started" if you're not already there) and choose whether you want to run CC Studio in "Simple Mode" or not.
- CCS Cloud[20]
    - Create a TI account.
    - Download and install TI Cloud Agent and install the Firefox/Chrome extension (CCS Cloud is not compatible with any other browser).
    - The wiki page for TI Cloud Agent[21] says under the "Supported Devices" heading at the bottom of the page, that "The Cloud IDE does not support debugging of MSP-EXP430G2452 and MSP-EXP430G2553 when used on Linux or MacOS. Loading of images to flash is supported on all operating systems." However, I had no issues with this using either my MSP-EXP430G2ET Launchpad development board nor the MSP430G2553 that I build on my breadboard (which was using the ez-FET debug adapter on my development board).
- Energia
    - Seems to be nearly identical to the Arduino IDE, which is to say, no native debugging.

Step 4: Write a program or open an example program

Open an example program

- Code Composer Studio
    - Click "View" > "Resource Explorer" to open the "Resource Explorer" (TIREX, for short).
    - In the page that opens up, enter your MCU or development board part number in the search box in the upper left-hand corner.
    - When the search completes, browse through the folders that show up. These are all of the support files that TI provides for the part number that you entered and could include datasheets, user manuals, header files, device libraries, and examples.
    - Examples are noted by a folder with the letters "CCS" above them in blue. Click on the title of an example you'd like to try and in the page that opens up to the right of the menu, first install the project and then, once that's complete, import the project into the IDE.

[20] https://dev.ti.com/
[21] http://processors.wiki.ti.com/index.php/TI_Cloud_Agent

- CCS Cloud
    - Click “Project” > “Browse Examples...”
    - In the page that opens up, enter your MCU or development board part number in the search box in the upper right-hand corner.
    - When the search completes, browse through the folders that show up. These are all of the support files that TI provides for the part number that you entered and could include datasheets, user manuals, header files, device libraries, and examples.
    - Examples are noted by a CCS logo (a cube formed from three red squares). Find an example you want to try, click the three vertical dots to the right of the name, and click “Import to CCS Cloud IDE.”

Write a program

- Code Composer Studio
    - Click “File” > “New” > “CCS Project”
    - In the window that opens up:
        - Select the MCU or device that you will be programming; selecting a filter first will help narrow down the list that pops up.
        - Give your project a name.
        - Optional: Select a compiler (TI-CGT is selected by default).
        - Select which type of project you want to create.
        - Click “Finish”.
    - Right-click on the “msp430.h” file which was automatically included at the top of your main file and select “Open declaration”.
    - Read through the header file to familiarize yourself with the register names so you can use them in your code to configure the registers on your MSP430.
    - Open the Resource Explorer on TI Cloud[22] (you will need to have a TI account), search for your MCU or development board by part number, and view any of the examples to see how this header file is used.
    - See also Chris Svec’s notes in Embedded Software Engineering 101[23] on the Embedded.fm blog.
- CCS Cloud
    - Click “File” > “New CCS Project…”
    - In the window that opens up:
        - Give your project a name.
        - Select the MCU or device that you will be programming; selecting a filter first will help narrow down the list that pops up.
        - Optional: Select a compiler (TI-CGT is selected by default).
    - Right-click on the “msp430.h” file which was automatically included at the top of your main file and select “Jump to definition”.

[22] https://dev.ti.com/
[23] https://embedded.fm/blog/ese101

    - Read through the header file to familiarize yourself with the register names so you can use them in your code to configure the registers on your MSP430.
    - Open the Resource Explorer on TI Cloud[24] (you will need to have a TI account), search for your MCU or development board by part number, and view any of the examples to see how this header file is used.
- Text editor
    - Locate the header file “msp430.h”; for me, this was in “/ti/msp430-gcc/include/”.
    - Add “#include “msp430.h”” to your code.
    - Read through the header file to familiarize yourself with the register names so you can use them in your code to configure the registers on your MSP430.
    - Open the Resource Explorer on TI Cloud[25] (you will need to have a TI account), search for your MCU or development board by part number, and view any of the examples to see how this header file is used.

Software Development Kits[26]

- Header file: msp430.h
    - Included with MSP430-GCC-OPENSOURCE[27].
- MSP430 DriverLib available in Resource Explorer for the following devices:
    - MSP430F5xx_6xx
    - MSP430FR57xx
    - MSP430FR5xx_6xx
    - MSP430FR2xx_4xx
    - MSP430i2xx
- Other software libraries
    - Includes libraries such as DSPLib, Graphics Library, and FRAM Utilities.
    - Not all driver libraries are available for all MSP430 devices.
    - Open the Resource Explorer on TI Cloud[28] (you will need to have a TI account) and search for your MCU or development board by part number to see which ones are available for your MCU.
- PlatformIO (running inside VS Code)
- Neither SysConfig nor PinConfig are available for any MSP430 device.

---

[24] https://dev.ti.com/
[25] https://dev.ti.com/
[26] Software Development Kits include any piece of software that makes it easier for the developer to write application code. Many such SDKs exist, so the kind focused on here relates to managing the MCU hardware (startup code, MCU core, peripherals, etc). This is typically the first thing a developer needs to start working with their target MCU. Some common examples of this type of SDK include makefile or IDE-specific project configuration files, header files for referencing the MCU registers, or a HAL (hardware abstraction layer) which is sometimes called a low-level driver. Any of these components may be made available either directly as source code or indirectly through a GUI.
[27] http://www.ti.com/tool/MSP430-GCC-OPENSOURCE
[28] https://dev.ti.com/

Step 5: Compile the program

- This will simply compile the program without downloading it to your MSP430 or starting a debug session.
    - To compile the program and download it to your MSP430, see “Flash the program to the MCU” below.
    - To compile the program, download it to your MSP430, and start a debug session, see “Start a debug session” below.
- Code Composer Studio
    - Click the “Build” button (the icon is a hammer) or click “Project” > “Build Project”.
- CCS Cloud
    - Click “Project” > “Build Project” or “Project” > “Rebuild Project”.
- Text editor
    - **Options:**
        - **GCC**
        - **TI-CGT**
    - GCC
        - Installed by default with CCS or download from MSP430-GCC-OPENSOURCE[29].
        - Can be invoked by CCS (TI-CGT invoked by default).
        - Command line GCC
            - Install MSP430-GCC-OPENSOURCE[30]
            - Build executable by:
                - navigating to an example and running make, or
                - following the instructions under “2.2: Building Manually with GCC” in the guide, “GCC for MSP430 Microcontrollers”.
                - On Linux, my user profile did not have the accesses required to run make, so I had to run that command as the super user (i.e. “sudo make”).
        - See GCC for MSP430 Microcontrollers[31] and MSP430 GCC user guide[32] for additional details.
    - TI Compiler[33] (aka TI-CGT or MSP-CGT)
        - Installed by default with CCS or download from MSP-CGT[34].
        - Invoked by default by CCS.

---

[29] http://www.ti.com/tool/MSP430-GCC-OPENSOURCE
[30] http://www.ti.com/tool/MSP430-GCC-OPENSOURCE
[31] http://www.ti.com/lit/ml/slau591c/slau591c.pdf
[32] http://www.ti.com/lit/ug/slau646e/slau646e.pdf
[33] http://downloads.ti.com/docs/esd/SLAU132/index.html##viewer?document=%257B%2522href%2522%253A%2522%252Fdocs%252Fesd%252FSLAU132%2522%257D&url=read-this-first-stdz0496557.html%23STDZ0496557
[34] http://www.ti.com/tool/MSP-CGT

Step 6: Flash the program to the MCU

- This will compile your program and download it to your MSP430.
    - To simply compile your program without downloading it to your MSP430, see "Compile the program" above.
    - To compile the program, download it to your MSP430, and start a debug session, see "Start a debug session" below.
- Code Composer Studio
    - Click the "Flash" button (looks like a pair of blue braces, "{ }", situated inside a folder icon) or click "Run" > "Load" > Your program.
- CCS Cloud
    - Click "Target" > "Connect COM port…" to make sure that your debug adapter is connected to CCS Cloud.
    - Click the "Run" button to the right of the "Help" menu.
- Text editor
    - **Options:**
        - **MSP430-FLASHER**
        - **GDB Agent & GDB**
    - MSP430-FLASHER (to flash the program to the MCU WITHOUT starting a debug session)
        - Installed by default by Code Composer Studio or download from MSP430-FLASHER[35]
            - If downloading on Linux, copy "libmsp430.so" to /usr/lib.
        - Invoked by Code Composer Studio
        - Command line MSP430-FLASHER
            - Follow the command line GCC instructions above to compile your program and create an executable from your code (a file that ends in ".out").
            - Convert executable to the proper format with by executing the following in a terminal window (from the directory where your program executable is located):
                - $EXE_DIR/msp430-elf-objcopy -O ihex FILENAME.out NEW_FILENAME.hex
                - "EXE_DIR" is the location of "msp430-elf-objcopy". For me, this was in "/ti/msp430-gcc/bin/".
                - FILENAME.out is the name of your executable.
                - NEW_FILENAME.hex is the name of your new hex file.
                - On Linux, my user profile did not have the accesses required to run make, so I had to run that command as the

[35] http://www.ti.com/tool/MSP430-FLASHER

super user (i.e. “sudo EXE_DIR/msp430-elf-objcopy -O ihex FILENAME.out NEW_FILENAME.hex”).
- Now flash the hex file to your MSP430 by executing the following in a terminal window (from the directory where the “MSP430Flasher” executable is located; for me, this was “/ti/MSPFlasher_1.3.20/”):
  - $./MSP430Flasher -n MSP430G2xx3 FILE_PATH/NEW_FILENAME.hex -z [VCC] -i USB
  - “FILE_PATH/NEW_FILENAME.hex” is the path and filename to the hex file you just created.
- See the MSP Flasher User's Guide[36] for additional details.

- GDB Agent & GDB (to flash the program to the MCU from INSIDE a debugging session)
  - See below.

Step 7 (Optional): Start a debug session

- This will compile your program, download it to your MSP430, and start a debug session.
  - To simply compile your program without downloading it to your MSP430, see “Compile the program” above.
  - To simply compile the program and download it to your MSP430, see “Flash the program to the MCU” above.
- Code Composer Studio
  - Click the “Debug” button (the icon looks like a green bug) or click “Run” > “Debug”.
- CCS Cloud
  - Click “Target” > “Connect COM port…” to make sure that your debug adapter is connected to CCS Cloud.
  - Click the “Debug” button to the right of the “Help” menu.
- Text editor
  - GDB Agent & GDB
    - Installed by default with Code Composer Studio or MSP430-GCC-OPENSOURCE
    - Invoked by Code Composer Studio
    - Command line GDB Agent & GDB
      - GDB Agent initiates a connection to the ez-FET debug adapter (necessary for using GDB).
      - In a terminal window from the folder where gdb_agent_console is located (for me, this was “/ti/msp430-gcc/bin/”): $./gdb_agent_console ../msp430.dat
      - Or, on Windows, double-click the file “gdb_agent_gui”

[36] http://www.ti.com/lit/ug/slau654e/slau654e.pdf

        - "Click the Configure button and, in the Select board configuration file window, select the msp430.dat file. If successfully configured, an MSP430 device is displayed in the <Targets> list. The TCP/IP port for the GDB Agent is displayed when the MSP430 device is selected from the list. To start the GDB server, click the <Start> button when the MSP430 device is selected."[37]
    - Then, in another terminal window, from the same folder:
        - Run on of the following:
            - $make debug
            - $./msp430-elf-gdb FILE_PATH/FILENAME.out
        - (gdb) target remote :55000
        - (gdb) load FILENAME.out
        - To run: (gdb) continue
        - To set a breakpoint: (gdb) break file:line
        - To halt: Ctrl+C
        - To quit: (gdb) q
- See GCC for MSP430 Microcontrollers[38] and MSP430 GCC user guide[39] for additional details.

Helpful tools

- EnergyTrace[40] (possibly included with CC Studio)
- ULP (Ultra-Low Power) Advisor[41] (possibly included with CC Studio)

References

- Product page for MSP-EXP430G2ET[42]
- Product page for MSP430G2553[43]
- MSP430 GCC user guide[44]
- GCC for MSP430 Microcontrollers[45]
- MSPFlasher User's Guide[46]

---

[37] GCC for MSP430 Microcontrollers, pg 3 (http://www.ti.com/lit/ml/slau591c/slau591c.pdf)
[38] http://www.ti.com/lit/ml/slau591c/slau591c.pdf
[39] http://www.ti.com/lit/ug/slau646e/slau646e.pdf
[40] http://www.ti.com/tool/ENERGYTRACE
[41] http://www.ti.com/tool/ULPADVISOR
[42] http://www.ti.com/tool/MSP-EXP430G2ET?keyMatch=Msp-exp430g2et&tisearch=Search-EN-Keyword
[43] http://www.ti.com/product/MSP430G2553?keyMatch=MSP430G2553&tisearch=Search-EN-everything&usecase=part-number
[44] http://www.ti.com/lit/ug/slau646e/slau646e.pdf
[45] http://www.ti.com/lit/ml/slau591c/slau591c.pdf
[46] http://www.ti.com/lit/ug/slau654e/slau654e.pdf

- MSP430 Hardware Tools User's Guide[47]
- Linux Host Support CCSv5[48]
- Linux Host Support CCSv6[49]
- MSP Debuggers User's Guide[50]
- Finite State Machines for MSP430[51]
- TI Compiler Wiki[52]
- Embedded Software Engineering 101[53] (Chris Svec)

[47] http://www.ti.com/lit/ug/slau278ae/slau278ae.pdf
[48] http://processors.wiki.ti.com/index.php/Linux_Host_Support_CCSv5
[49] http://processors.wiki.ti.com/index.php/Linux_Host_Support_CCSv6
[50] http://www.ti.com/lit/ug/slau647m/slau647m.pdf
[51] http://www.ti.com/lit/an/slaa402a/slaa402a.pdf
[52] http://processors.wiki.ti.com/index.php/TI_Compiler_Information
[53] https://embedded.fm/blog/ese101

Tested Toolchain Combinations

See Compatible Toolchains[54] or something more legible.

MCU family	MCU part number	Board	Debug adapter	IDE/Code editor	Middleware	Compiler/Linker	Adapter driver	Debug software	Tested?	OS	Notes
MSP430	MSP430G2553	MSP-EXP430G2ET	ez-FET	CCStudio	msp430.h	TI-CGT	CCStudio	CCStudio	Yes	Windows 10	Used CCS in "Simple mode". Process was relatively painless.
MSP430	MSP430G2553	MSP-EXP430G2ET	ez-FET	CCS Cloud	msp430.h	TI-CGT	CCS Cloud	CCS Cloud	Yes	Ubuntu 18.04.03	Requires TI Cloud Agent and a Firefox/Chrome extensic (is not compatible with other browsers).
MSP430	MSP430G2553	MSP-EXP430G2ET	ez-FET	Text editor	msp430.h	MSP430-GCC	GDB Agent	MSP430-GDB	Yes	Ubuntu 18.04.03	GDB GUI available in Windows.
MSP430	MSP430G2553	Breadboard	ez-FET	CCStudio	msp430.h	TI-CGT	CCStudio	CCStudio	Yes	Windows 10	ezFET requires 4-wire connection (Pwr, Gnd, SBWTCK, SBWTDIO).
MSP430	MSP430G2553	Breadboard	ez-FET	CCS Cloud	msp430.h	TI-CGT	CCS Cloud	CCS Cloud	Yes	Ubuntu 18.04.03	Requires TI Cloud Agent and a Firefox/Chrome extensic (is not compatible with other browsers). ezFET requires 4-wire connection (Pwr, Gnd, SBWTCK, SBWTDIO).
MSP430	MSP430G2553	Breadboard	ez-FET	Text editor	msp430.h	MSP430-GCC	GDB Agent	MSP430-GDB	Yes	Ubuntu 18.04.03	
MSP430	MSP430G2553	MSP-EXP430G2ET	ez-FET	CCStudio	msp430.h	TI-CGT	CCStudio	CCStudio	Yes	Ubuntu 18.04.03	Extremely difficult to set up on Ubuntu.
MSP430	MSP430G2553	Breadboard	ez-FET	CCStudio	msp430.h	TI-CGT	CCStudio	CCStudio	Yes	Ubuntu 18.04.03	Extremely difficult to set up on Ubuntu.

[54] https://github.com/nathancharlesjones/Embedded-for-Everyone/tree/master/Getting-Started-Guides