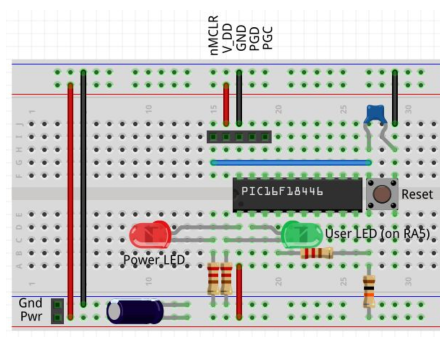
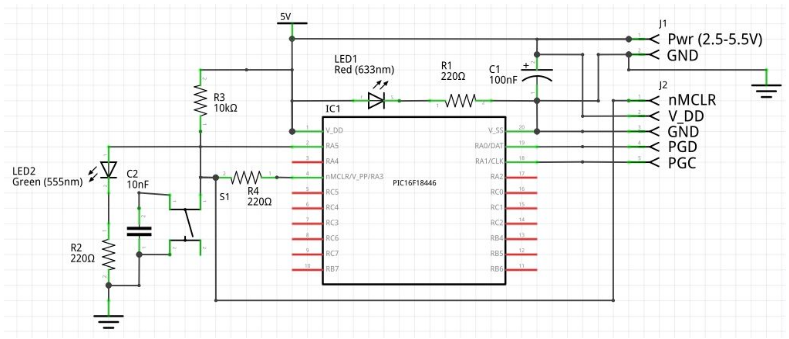
Getting Started with the PIC16/PIC18/dsPIC33

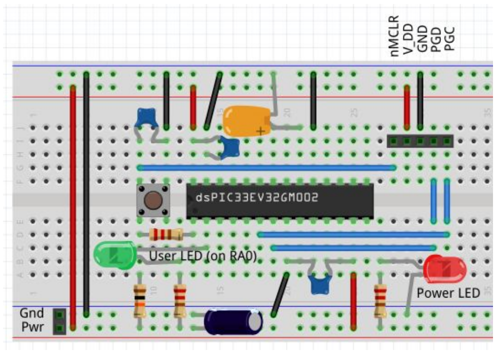
Step 1: Pick an MCU

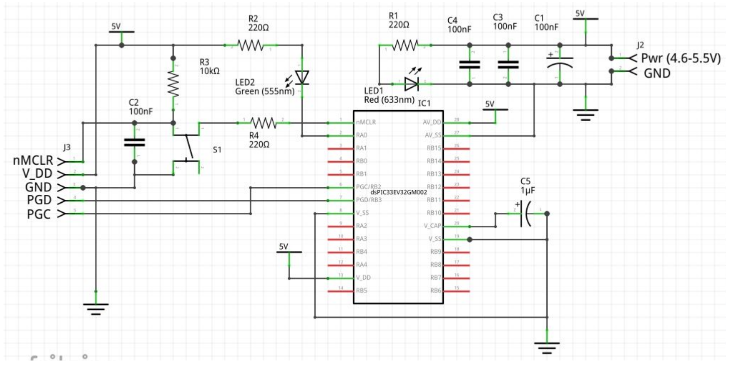




Figures 1&2: Breadboard layout and schematic for the minimum required circuit to program the PIC16F18446. Images were created using Fritzing; the original Fritzing file is available here[1].

[1] https://github.com/nathancharlesjones/Embedded-for-Everyone/tree/master/Getting-Started-Guides/PIC16-PIC18-dsPIC33





Figures 3&4: Breadboard layout and schematic for the minimum required circuit to program the dsPIC33EV32GM002. Images were created using Fritzing; the original Fritzing file is available here[2].

- **Options: The subset of PIC16, PIC18, and dsPIC33 MCUs that are compatible with the MPLAB SNAP debugger.**
  - There are lots of PIC MCUs to play with, of course, but all those outside of this subset would require either an expensive debugger ($50+) or SMT soldering.
  - The MPLAB SNAP debugger retails for $15[3] and just about every other debugger compatible with Microchip MCUs is $50 or more, so those are the MCUs I'm targeting.
  - To find the full list of Microchip MCUs that are compatible with the MPLAB SNAP debugger, you'll need to download the latest version of MPLAB X IDE[4] and open up a document in the download folder called "Device Support.htm". A copy of that document from late 2019 is located here[5].
  - However, not all of the MCUs that are compatible with the MPLAB SNAP debugger also come in DIP form. In October 2019 I searched Digikey for those that did, and came up with the list here[6]. The list is composed of a handful of MCUs from the PIC16, PIC18, and dsPIC33 product lines.
  - I bought the PIC16F18446 and the dsPIC33EV32GM002.
- In trying to figure out how to breadboard the PIC16F18446, I referenced the following document: PIC16(L)F18426/46 Datasheet[7], pg 21 ("Guidelines for Getting Started with PIC16(L)F18426/46 Microcontrollers" section). If you aren't prototyping with that exact MCU, the circuits are probably almost identical; find the datasheet for your exact MCU and look for a "Guidelines for Getting Started" section to be sure.

2 https://github.com/nathancharlesjones/Embedded-for-Everyone/tree/master/Getting-Started-Guides/PIC16-PIC18-dsPIC33
3 https://www.digikey.com/products/en?keywords=mplab%20snap
4 https://www.microchip.com/mplab/mplab-x-ide
5 https://github.com/nathancharlesjones/Embedded-for-Everyone/tree/master/Supporting-documents
6 https://github.com/nathancharlesjones/Embedded-for-Everyone/tree/master/Supporting-documents
7 http://ww1.microchip.com/downloads/en/DeviceDoc/40001985B.pdf

    - That document recommends a jumper connected between the reset capacitor (C1 in Figure 2-2, pg 22, in the above document) and a series resistor (R2 in Figure 2-2, pg 22, in the above document) before the nMCLR pin. The purpose of the jumper is to isolate C1 from nMLCR during programming or debugging as the added capacitance might mess up specific resets triggered by the debug adapter. I excluded the jumper, connecting C1 directly to R2, and had no issues. I also included a reset button and have tested it without C1 or R2 with no issues.
    - The MPLAB SNAP user's guide[8] states that its 8-pin connector has the same pinout as the PICKit4. The PICKit4 user's guide[9] shows the pinout for the PICKit4 in Figure B-2 on pg 63. It DOES NOT provide power to the MCU through pin 2, V_DD, as this is just a sensing pin for the MPLAB SNAP. Other debug adapters have the ability to provide power to the target MCU. If you are using the MPLAB SNAP, you'll need to find another way to power the device. A simple setup would be a 9V battery into a 5V regulator; an easier setup would be to use a breadboard power supply[10].
- Schematic notes for the PIC16F18446:
    - Pin 1 (V_DD) should be connected to 2.5-5.5V. The MPLAB SNAP DOES NOT provide this power and an external power supply MUST be added. Other debug adapters have the ability to provide power to the target MCU.
    - Pin 20 (V_SS) should be connected to GND.
    - C1 is a 0.1 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 4) high in the absence of another signal to keep the PIC16 from randomly resetting; 10 kOhms is the recommended value.
    - S1 and C2 (10 nF) are the reset button and a smoothing capacitor (to prevent switch bounce from triggering multiple resets in a row).
    - R4
- In trying to figure out how to breadboard the dsPIC33EV32GM002, I referenced the following document: dsPIC33EVXXXGM00X/10X Family Datasheet[11] ("GUIDELINES

[8] http://ww1.microchip.com/downloads/en/DeviceDoc/50002787B.pdf; Finding this guide was a bit tough, since the Microchip website wouldn't let me search for the MPLAB SNAP product page directly. I had to first search for a compatible MCU (such as either of the MCUs listed here), navigate to its product page, select the "Development Environment" button, select the "Emulators & Debuggers" button, and, finally, select the "Learn More" button underneath the debug adapter I was interested in seeing.

[9] http://ww1.microchip.com/downloads/en/DeviceDoc/MPLAB_PICkit_4_ICD_User's_Guide_DS50002751C.pdf; Follow the steps above to get to the PICKit4 product page.

[10] https://www.digikey.com/product-detail/en/dfrobot/DFR0140/1738-1057-ND/6588479

[11] http://ww1.microchip.com/downloads/en/DeviceDoc/dsPIC33EVXXXGM00X-10X-Family-Data-Sheet-DS70005144H.pdf

FOR GETTING STARTED WITH 16-BIT DIGITAL SIGNAL CONTROLLERS" section). If you aren't prototyping with that exact MCU, the circuits are probably almost identical; find the datasheet for your exact MCU and look for a "Guidelines for Getting Started" section to be sure.

    - That document recommends a jumper connected between the reset capacitor (C1 in Figure 2-2, pg 22, in the datasheet) and a series resistor (R2 in Figure 2-2, pg 22, in the datasheet) before the nMCLR pin. The purpose of the jumper is to isolate C1 from nMLCR during programming or debugging as the added capacitance might mess up specific resets triggered by the debug adapter. I excluded the jumper, connecting C1 directly to R2, and had no issues. I also included a reset button and have tested it without C1 or R2 with no issues.
    - According to Section 2.3 on pg 22 of the datasheet, V_CAP should be a low-ESR tantalum capacitor in the range 4.7-10 uF. I used an electrolytic (10 uF) without problems.
    - Note 1 on Figure 2-1 on pg 22 of the datasheet suggests using an inductor to connect AV_DD to V_DD instead of a wire to improve ADC noise rejection. I used a wire with no issues, though I did not test noise levels on the ADC.
    - This MCU has multiple programming or ICSP pins which can be connected to the MPLAB SNAP: PGD1/PGC1 on pins 7 and 6 (the same as used in the schematic above), PGD2/PGC2 on pins 14 and 15, and PGD3/PGC3 on pins 4 and 5. PGD1/PGC1 are enabled by default; the other two can be selected by changing the ICS bits in the FICD register (see Table 27-1 on pg 322 and Table 27-2 on pg 325 of the datasheet for slightly more information).
    - See note about the MPLAB SNAP pinout and power supply, above.
- Schematic notes for the dsPIC33EV32GM002:
    - Pin 13 (V_DD) should be connected to 4.6-5.5V. The MPLAB SNAP DOES NOT provide this power and an external power supply MUST be added. Other debug adapters have the ability to provide power to the target MCU.
    - Pins 8 and 19 (V_SS) and 27 (AV_SS) should be connected to GND.
    - C1, C3, and C4 are 0.1 uF power filtering capacitors.
    - 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 4) high in the absence of another signal to keep the PIC16 from randomly resetting; 10 kOhms is the recommended value.
    - S1 and C2 (10 nF) are the reset button and a smoothing capacitor (to prevent switch bounce from triggering multiple resets in a row). R4 protects the nMCLR pin in the event that C2 accidentally discharges into it.
    - C5 is a required component connected to V_CAP (see section 2.3 o pg 22 of the datasheet for more information).

Step 2 (Optional): Pick a (different) debug adapter

- **Options:**
    - **MPLAB SNAP**
    - **PICKit 3**
    - **PICKit 4**
    - **ICD 3**
    - **ICD 4**
- Microchip offers many feature-rich debug adapters but the cheapest way to program a PIC or dsPIC is by using the MPLAB SNAP (as previously discussed), which only costs $15. Other debug adapters cost $50 or more.
- In terms of number of compatible MCUs, the ICD 3 and PICKit 3 seem to have the most, followed closely by the ICD 4 and PICKit 4. The MPLAB SNAP has, by far, the fewest supported MCUs, though it still supports approximately 350 MCUs, with about 150 of those being available in DIP form. See the document “Device Support.htm” (how to get it is described above) for more information.
- In terms of features, it is my understanding that at least the PICKit 3/4 can supply power to a target MCU (eliminating the need for a separate power supply during programming) and I would hazard a guess that the ICD 3/4 supports this feature also. These devices may also support faster programming and debugging speeds as well as other, more advanced, debugging features.
- The ICD 3 and PICKit 3 are not recommended for new designs.
- The ICD 4 sells for $250 on Digikey[12] and the PICKit 4 for $50[13]. The ICD 3 and PICKit 3 are listed on Digikey as “obsolete”.

Step 3: Pick an IDE/code editor

- **Options (all are IDEs unless otherwise noted):**
    - **MPLAB X IDE**
    - **MPLAB Xpress**
    - **Your favorite text editor (NOT an IDE)**
- MPLAB X IDE

[12] https://www.digikey.com/products/en?keywords=microchip%20icd4
[13] https://www.digikey.com/products/en?keywords=microchip%20pickit4

    - Download and install the correct version for your OS from the MPLAB X IDE website[14].
    - Also install 32-bit binaries if needed[15].
    - Once MPLAB X IDE is open, select the “Install More Plugins” link on the “Start Page”. Find and install “MPLAB Code Configurator”, along with any other plugins that seem useful. This is not required, but MPLAB Code Configurator (MCC) is the only piece of middleware that I could find for these MCUs and it makes development on them much easier.
- MPLAB Xpress
    - This is an online IDE that only supports Microchip development boards or developers using the PICKit 4.
    - Debugging requires Java 8 and USB bridge tool

Step 4: Write a program

- NOTE: If example programs exist for Microchip MCUs, their location was not made clear to me throughout this limited development project and, as such, I cannot tell you how to locate them.
- MPLAB X IDE
    - Click “File” > “New project”.
    - In the window that opens up, select “Microchip embedded” under categories and “Standalone project” under “Projects”. Then click “Next”.
    - In the next window, find the device or MCU that you’re creating a project for and click “Next”.
        - If you’re creating a project for the PIC16F18446, you can narrow down the device list by selecting “Mid-range 8-bit MCUs (PIC10/12/16/MCP)” under the “Family” list.
        - If you’re creating a project for the dsPIC33EV32GM002, select “16-bit DSCs (dsPIC33)” from the “Family” list.
    - In the next window, select the debug adapter that you’ll be using and click “Next”.
    - In the next window, download and then select your desired compiler. Then click “Next”.
        - For 8-bit devices (such as the PIC16F18446), you’ll need the “XC8” compiler. See the product page[16] for more information; the files you’ll need to download to install this compiler are located towards the bottom of the page, under the “Downloads” tab.
        - For 16-bit devices (such as the dsPIC33EV32GM002), you’ll need the “XC16” compiler. See the product page[17] for more information; the files

[14] https://www.microchip.com/mplab/mplab-x-ide
[15] https://microchipdeveloper.com/install:mplabx-lin64
[16] https://www.microchip.com/mplab/compilers
[17] https://www.microchip.com/mplab/compilers

        you'll need to download to install this compiler are located towards the bottom of the page, under the "Downloads" tab.
    - In the next window, create a project name and click "Finish".
    - Although you can begin writing code now, it will be easier to leverage the MCC plugin. Select the MCC icon from the toolbar (shaped like a blue shield with "MCC" in the middle).
    - Select any desired options. When you're finished, click the "Generate code" button in the "Resource Management" window.
        - This was a touch difficult for me to find, at first. For me, the "Resource Management" window was located in the upper-left of my screen and had sections called "Project Resources" and "Device Resources". Due to screen size limitations, the "Generate code" button, for me, was truncated to "Gen…".
    - The generated code will appear inside your project tree within folders called "MCC Generated Files". The header files are likely the best place to start looking for the interface functions that you'll want to start using. When I set up just some basic parameters and a GPIO (to toggle an LED), I found the GPIO functions I needed inside "Header Files" > "MCC Generated Files" > "pin_manager.h".
- Text editor
    - I could not find any generic header files for use with Microchip MCUs. If you prefer to develop from inside a text editor, my best suggestion is to use MPLAB X IDE and MCC to generate your starter code, and then transition to editing the resulting project files inside your text editor of choice.

Software Development Kits[18]

- MPLAB Code Configurator
    - Available as a plugin to MPLAB X IDE. Possibly also available inside MPLAB Xpress.
- MPLAB Harmony Configurator
    - Only available for 32-bit PIC or SAM MCUs.
- Possibly other plugins to MPLAB X IDE

Step 5: Compile the program

- This will simply compile the program without downloading it to your PIC/dsPIC or starting a debug session.

[18] 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.

    - To compile the program and download it to your PIC/dsPIC, see “Flash the program to the MCU” below.
    - To compile the program, download it to your PIC/dsPIC, and start a debug session, see “Start a debug session” below.
- MPLAB X IDE
    - Click the “Build” button (the icon is a hammer), click “Production” > “Build Project”, or press F11.
- Text editor
    - **Options:**
        - **XC8 (for 8-bit MCUs such as the PIC16/PIC18)**
        - **XC16 (for 16-bit MCUs such as the dsPIC33)**
        - **SDCC (for 8-bit MCUs such as the PIC16/PIC18)**
    - XC8 (use with 8-bit MCUs such as the PIC16/PIC18)
        - The easiest way to compile from the command line using XC8 is to build your project in MPLAB X IDE and run “make” from the project’s root directory.
        - You can also observe and copy the commands being echoed in the output window in MPLAB X when you build your project. To see what those looked like for me, see the section “Example XC8 Commands”.
        - Running these commands inside a terminal window from the project’s root folder performs the same functions as building from inside MPLAB X IDE.
            - The command line options are likely the same for you, but you’ll, of course, need to update each of the filepaths.
        - More information can be found in the XC8 user’s guide[19].
    - XC16 (use with 16-bit MCUs such as the dsPIC33)
        - Much of the information for running XC16 from the command line is the same as for XC8, above.
        - I used “make” again, but you could also copy the necessary commands from the output window of MPLAB X IDE. To see what those looked like for me, see the section “Example XC16 Commands”.
        - More information can be found in the XC16 user’s guide[20] or the “XC16 User’s Guide for Embedded Engineers”[21].
    - SDCC[22] (use with 8-bit MCUs such as the PIC16/PIC18)
        - Seems to support some PIC devices, though it may require GPUTILS[23].

[19] http://ww1.microchip.com/downloads/en/DeviceDoc/MPLAB_XC8_C_Compiler_User_Guide_for_PIC.pdf
[20] http://ww1.microchip.com/downloads/en/DeviceDoc/XC16%20C%20Compiler%20UG%20DS50002071J.pdf
[21] http://ww1.microchip.com/downloads/en/DeviceDoc/50002446C.pdf
[22] http://sdcc.sourceforge.net/
[23] https://gputils.sourceforge.io/

- Of the MCUs that are compatible with the MPLAB SNAP, it seems that only 43 are also supported by SDCC. Only those with an asterisk also come in DIP packages:
  - PIC18LF8493
  - PIC18LF8490
  - PIC18LF8410
  - PIC18LF8393
  - PIC18LF8390
  - PIC18LF8310
  - PIC18LF6493
  - PIC18LF6490
  - PIC18LF6410
  - PIC18LF6393
  - PIC18LF6390
  - PIC18LF6310
  - PIC18LF45J10*
  - PIC18LF44J10*
  - PIC18LF25K50*
  - PIC18LF25J11*
  - PIC18LF25J10*
  - PIC18LF24K50*
  - PIC18LF24J11
  - PIC18LF24J10
  - PIC18F97J94
  - PIC18F97J60
  - PIC18F96J94
  - PIC18F96J65
  - PIC18F96J60
  - PIC18F95J94
  - PIC18F67J11
  - PIC18F66J65
  - PIC18F66J55
  - PIC18F66J16
  - PIC18F66J11
  - PIC18F25K50*
  - PIC18F24K50*
  - PIC16LF1824T39A
  - PIC16LF1779*
  - PIC16LF1777*
  - PIC16LF1567*
  - PIC16LF1559*
  - PIC16LF1554*
  - PIC16LF1527

        - PIC16F1829LIN
        - PIC16F1779*
        - PIC16F1777*
    - Additional resources
        - Hackaday: How to Program PICs using Linux[24]
        - SDCC user's guide[25]

Step 6: Flash the program to the MCU

- This will compile your program and download it to your PIC/dsPIC.
    - To simply compile your program without downloading it to your PIC/dsPIC, see "Compile the program" above.
    - To compile the program, download it to your PIC/dsPIC, and start a debug session, see "Start a debug session" below.
- MPLAB X IDE
    - Select either the "Run" button (icon looks like a green "play" button) or the "Make and program flash" button from the toolbar (icon looks like a green arrow pointing downward from a piece of code to an MCU).
- Text editor
    - See below.
    - A program that comes with MPLAB X called "ipecmd.sh" seems to be able to flash the PICs (it's likely the very program that MPLAB X uses, in fact), but I couldn't figure out how to use it.

Step 7 (Optional): Start a debug session

- This will compile your program, download it to your PIC/dsPIC, and start a debug session.
    - To simply compile your program without downloading it to your PIC/dsPIC, see "Compile the program" above.
    - To simply compile the program and download it to your PIC/dsPIC, see "Flash the program to the MCU" above.
- MPLAB X IDE
    - Click the "Debug" button (icon looks like a green "play" button in front of a few lines of code, one of which is highlighted in red) or click "Debug" > "Debug Project" from the menus.
    - Setting breakpoints and various stepping commands are available from the "Debug" menu; I was not able to locate any such buttons next to the code or in any debug "perspective".
- Text editor

[24] https://hackaday.com/2010/11/03/how-to-program-pics-using-linux/
[25] http://sdcc.sourceforge.net/doc/sdccman.pdf

- Compile your program as described in Step 5.
- ****NOTE**: I discovered only recently that programming PIC MCUs using mdb.sh, described below, seems to work fine until the MPLAB SNAP is disconnected from the PIC, at which point the PIC will not restart. The issue seems to be described here[26]. I could not find a workaround, making command line programming of the PIC useless, more or less.**
- Run the “Start-MDB-for-PIC16.sh” script, updating the path to mdb.sh and the “MDB-for-PIC16.init” files as discussed below.
    - Replace the path to mdb.sh to where it resides in your installation folder for MPLAB X.
    - Replace “PIC16F18446” with your specific PIC device.
    - Change “SNAP” if you’re using a different debug adapter.
    - Replace the path to “Blinky.debug.elf” to point to your executable.
- The script and .init files simply the commands, but you could also perform them manually, as described below.
    - Navigate to the place where MPLAB X IDE was installed and find a file called “mdb.sh”. Run it from a terminal window.
    - First, set your device: At the prompt that appears after starting mdb.sh (“>”), enter “Device PIC16F18446”, replacing “PIC16F18446” with your specific device if it is not that.
    - Next, set your debug adapter: At that same prompt, enter “Hwtool SNAP”, replacing “SNAP” with your specific debug adapter if you’re not using the MPLAB SNAP.
    - Finally, load your code: At the same prompt, enter “Program FILEPATH_TO_COF_ELF_OR_HEX_FILE_FOR_DESIRED_PROGRAM ”, replacing the argument above with your actual filepath.
- Use “Run”, “Halt”, “Reset”, “Break FILE:LINE_NUMBER”, etc to control the debug session.
- For more information, read the MDB user’s guide[27].

Helpful tools

- None, save the plugins available in MPLAB X IDE.

Example XC8 Commands

What follows are the commands displayed in the output window of MPLAB X IDE when I built my “Blinky” project for a PIC16F18446:

- /opt/microchip/xc8/v2.10/bin/xc8-cc -mcpu=16F18446 -c -D__DEBUG=1 -fno-short-double -fno-short-float -O0 -fasmfile -maddrqual=ignore -xassembler-with-cpp

[26] https://www.microchip.com/forums/m1061151.aspx
[27] http://ww1.microchip.com/downloads/en/DeviceDoc/50002102D.pdf

```
  -mwarn=-3 -Wa,-a -DXPRJ_default=default  -msummary=-psect,-class,+mem,-hex,-file
  -ginhx032 -Wl,--data-init -mno-keep-startup -mno-osccal -mno-resetbits
  -mno-save-resetbits -mno-download -mno-stackcall   -std=c99 -gdwarf-3
  -mstack=compiled:auto:auto  -o
  build/default/debug/mcc_generated_files/device_config.p1
  mcc_generated_files/device_config.c
● /opt/microchip/xc8/v2.10/bin/xc8-cc  -mcpu=16F18446 -c  -D__DEBUG=1
  -fno-short-double -fno-short-float -O0 -fasmfile -maddrqual=ignore -xassembler-with-cpp
  -mwarn=-3 -Wa,-a -DXPRJ_default=default  -msummary=-psect,-class,+mem,-hex,-file
  -ginhx032 -Wl,--data-init -mno-keep-startup -mno-osccal -mno-resetbits
  -mno-save-resetbits -mno-download -mno-stackcall   -std=c99 -gdwarf-3
  -mstack=compiled:auto:auto  -o build/default/debug/mcc_generated_files/mcc.p1
  mcc_generated_files/mcc.c
● /opt/microchip/xc8/v2.10/bin/xc8-cc  -mcpu=16F18446 -c  -D__DEBUG=1
  -fno-short-double -fno-short-float -O0 -fasmfile -maddrqual=ignore -xassembler-with-cpp
  -mwarn=-3 -Wa,-a -DXPRJ_default=default  -msummary=-psect,-class,+mem,-hex,-file
  -ginhx032 -Wl,--data-init -mno-keep-startup -mno-osccal -mno-resetbits
  -mno-save-resetbits -mno-download -mno-stackcall   -std=c99 -gdwarf-3
  -mstack=compiled:auto:auto  -o
  build/default/debug/mcc_generated_files/pin_manager.p1
  mcc_generated_files/pin_manager.c
● /opt/microchip/xc8/v2.10/bin/xc8-cc  -mcpu=16F18446 -c  -D__DEBUG=1
  -fno-short-double -fno-short-float -O0 -fasmfile -maddrqual=ignore -xassembler-with-cpp
  -mwarn=-3 -Wa,-a -DXPRJ_default=default  -msummary=-psect,-class,+mem,-hex,-file
  -ginhx032 -Wl,--data-init -mno-keep-startup -mno-osccal -mno-resetbits
  -mno-save-resetbits -mno-download -mno-stackcall   -std=c99 -gdwarf-3
  -mstack=compiled:auto:auto  -o build/default/debug/main.p1 main.c
● /opt/microchip/xc8/v2.10/bin/xc8-cc  -mcpu=16F18446
  -Wl,-Map=dist/default/debug/Blinky.debug.map  -D__DEBUG=1
  -DXPRJ_default=default  -Wl,--defsym=__MPLAB_BUILD=1  -fno-short-double
  -fno-short-float -O0 -fasmfile -maddrqual=ignore -xassembler-with-cpp -mwarn=-3
  -Wa,-a -msummary=-psect,-class,+mem,-hex,-file  -ginhx032 -Wl,--data-init
  -mno-keep-startup -mno-osccal -mno-resetbits -mno-save-resetbits -mno-download
  -mno-stackcall -std=c99 -gdwarf-3 -mstack=compiled:auto:auto
  -Wl,--memorysummary,dist/default/debug/memoryfile.xml -o
  dist/default/debug/Blinky.debug.elf  build/default/debug/main.p1
  build/default/debug/mcc_generated_files/device_config.p1
  build/default/debug/mcc_generated_files/mcc.p1
  build/default/debug/mcc_generated_files/pin_manager.p1
```

The first four commands compile the project's source code (device_config.c, mcc.c, pin_manage.c, and main.c). The last command links the object files together and links in any libraries. The output from the last command is a file called “Blinky.debug.elf” in the project folder dist/default/debug.

Example XC16 Commands

What follows are the commands displayed in the output window of MPLAB X IDE when I built my “Blinky” project for a dsPIC33EV32GM002:

- /opt/microchip/xc16/v1.41/bin/xc16-gcc   main.c -o build/default/production/main.o -c -mcpu=33EV32GM002 -MMD -MF build/default/production/main.o.d -mno-eds-warn -g -omf=elf -DXPRJ_default=default -legacy-libc   -O0 -msmart-io=1 -Wall -msfr-warn=off
- /opt/microchip/xc16/v1.41/bin/xc16-gcc   -o dist/default/production/Blinky_dsPIC33.X.production.elf build/default/production/mcc_generated_files/reset.o build/default/production/main.o build/default/production/mcc_generated_files/system.o build/default/production/mcc_generated_files/traps.o build/default/production/mcc_generated_files/clock.o build/default/production/mcc_generated_files/interrupt_manager.o build/default/production/mcc_generated_files/mcc.o build/default/production/mcc_generated_files/pin_manager.o -mcpu=33EV32GM002   -omf=elf -DXPRJ_default=default -legacy-libc -Wl,--local-stack,,--defsym=__MPLAB_BUILD=1,,--script=p33EV32GM002.gld,--stack=16,--check-sections,--data-init,--pack-data,--handles,--isr,--no-gc-sections,--fill-upper=0,--stackguard=16,--no-force-link,--smart-io,-Map="dist/default/production/Blinky_dsPIC33.X.production.map",--report-mem,--memorysummary,dist/default/production/memoryfile.xml
- /opt/microchip/xc16/v1.41/bin/xc16-bin2hex dist/default/production/Blinky_dsPIC33.X.production.elf -a -omf=elf

References

- Product page for PIC16F18446[28]
- PIC16(L)F18426/46 Datasheet[29] and errata[30]
- Product page for dsPIC33EV32GM002[31]
- dsPIC33EVXXXGM00X/10X Family Datasheet[32]
- MPLAB SNAP user’s guide[33]

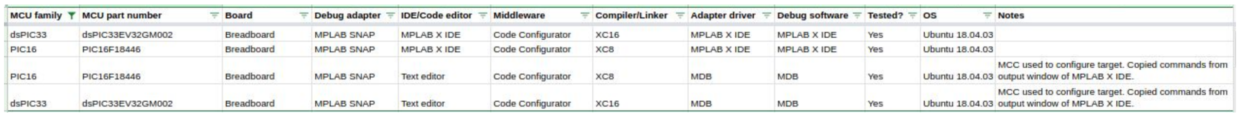
[28] https://www.microchip.com/wwwproducts/en/PIC16F18446
[29] http://ww1.microchip.com/downloads/en/DeviceDoc/40001985B.pdf
[30] http://ww1.microchip.com/downloads/en/DeviceDoc/PIC16LF18426_46-Errata-70000799B.pdf
[31] https://www.microchip.com/wwwproducts/en/dsPIC33EV32GM002
[32] http://ww1.microchip.com/downloads/en/DeviceDoc/dsPIC33EVXXXGM00X-10X-Family-Data-Sheet-DS70005144H.pdf
[33] http://ww1.microchip.com/downloads/en/DeviceDoc/50002787B.pdf

- PICKit4 user's guide[34]
- XC8 user's guide[35]
- XC16 user's guide[36]
- "XC16 User's Guide for Embedded Engineers"[37]
- SDCC user's guide[38]
- Linux terminal only PIC programming[39]
- Programming PICs with Python[40]

[34] http://ww1.microchip.com/downloads/en/DeviceDoc/MPLAB_PICkit_4_ICD_User's_Guide_DS50002751C.pdf
[35] http://ww1.microchip.com/downloads/en/DeviceDoc/MPLAB_XC8_C_Compiler_User_Guide_for_PIC.pdf
[36] http://ww1.microchip.com/downloads/en/DeviceDoc/XC16%20C%20Compiler%20UG%20DS50002071J.pdf
[37] http://ww1.microchip.com/downloads/en/DeviceDoc/50002446C.pdf
[38] http://sdcc.sourceforge.net/doc/sdccman.pdf
[39] https://www.electropepper.org/blog/item/linux-terminal-only-pic-programming
[40] https://hackaday.io/project/11864-tritiled/log/72158-programming-pics-with-python

Tested Toolchain Combinations

See Compatible Toolchains[41] 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
dsPIC33	dsPIC33EV32GM002	Breadboard	MPLAB SNAP	MPLAB X IDE	Code Configurator	XC16	MPLAB X IDE	MPLAB X IDE	Yes	Ubuntu 18.04.03	
PIC16	PIC16F18446	Breadboard	MPLAB SNAP	MPLAB X IDE	Code Configurator	XC8	MPLAB X IDE	MPLAB X IDE	Yes	Ubuntu 18.04.03	
PIC16	PIC16F18446	Breadboard	MPLAB SNAP	Text editor	Code Configurator	XC8	MDB	MDB	Yes	Ubuntu 18.04.03	MCC used to configure target. Copied commands from output window of MPLAB X IDE.
dsPIC33	dsPIC33EV32GM002	Breadboard	MPLAB SNAP	Text editor	Code Configurator	XC16	MDB	MDB	Yes	Ubuntu 18.04.03	MCC used to configure target. Copied commands from output window of MPLAB X IDE.

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