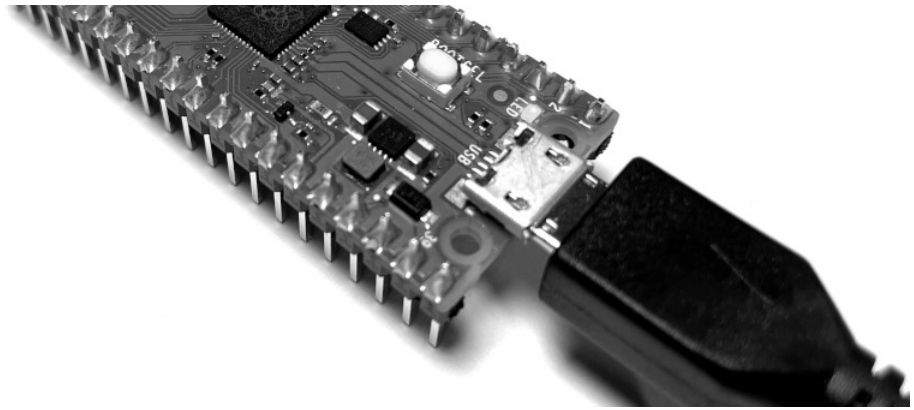
How to Set Up Raspberry Pi Pico C/C++ Toolchain on Windows with VS Code

APRIL 9, 2021 / TUTORIAL / 85 COMMENTS



This tutorial will show you how to install the Raspberry Pi Pico toolchain on Windows 10 for C and C++ development.

Raspberry Pi has a fantastic getting started guide for the Pico that covers installation steps for the major operating systems. On Linux (specifically, most flavors of Debian), you can run a single script that will install everything for you. On macOS, you need to use Homebrew to install the toolchain, which is only a few commands in the terminal.

Windows, however, is a different story. Installing the toolchain is an arduous process, requiring multiple programs and manually modifying the Windows Path. The Raspberry Pi Pico getting started guide shows you how to do this, but I have issues with two parts: you need to install Build Tools for Visual Studio (around 6 GB) and you must run VS Code from within a Developer Command Prompt every time.

CATEGORIES

- Classroom
- Opinion
- Personal
- Project
- Tutorial

TAGS

ARDUINO ARM BACKUP BLE BLUETOOTH BOTTLE

Directory Setup

With the exception of CMake and Python, we will want all of our tools and SDK files to exist in one folder on Windows. This setup will make it easy to configure the Path and find things later on.

Create a folder named **VSARM** in the top level of your C drive.

In C:\VSARM, create the following folders:

- C:\VSARM\armcc
- C:\VSARM\lib
- C:\VSARM\mingw
- C:\VSARM\sdk

IMAGE INTEL IOT ISR
JAVASCRIPT LED LINUX
MACHINE LEARNING NVIDIA
OLIMEX OLINUXINO PCB
PHASER PROGRAMMING
PYTHON QUICK TIPS
RASPBERRY PI PICO SAMD21
SD CARD SPARKFUN STM32
TEENSY TENSORFLOW
TUTORIAL WEBSOCKET
WIFI WINDOWS

We use these short, one-word folder names to keep paths short and to avoid spaces. Some Unix/Linux tools (that have been ported to Windows) do not work well with paths that have spaces in them.

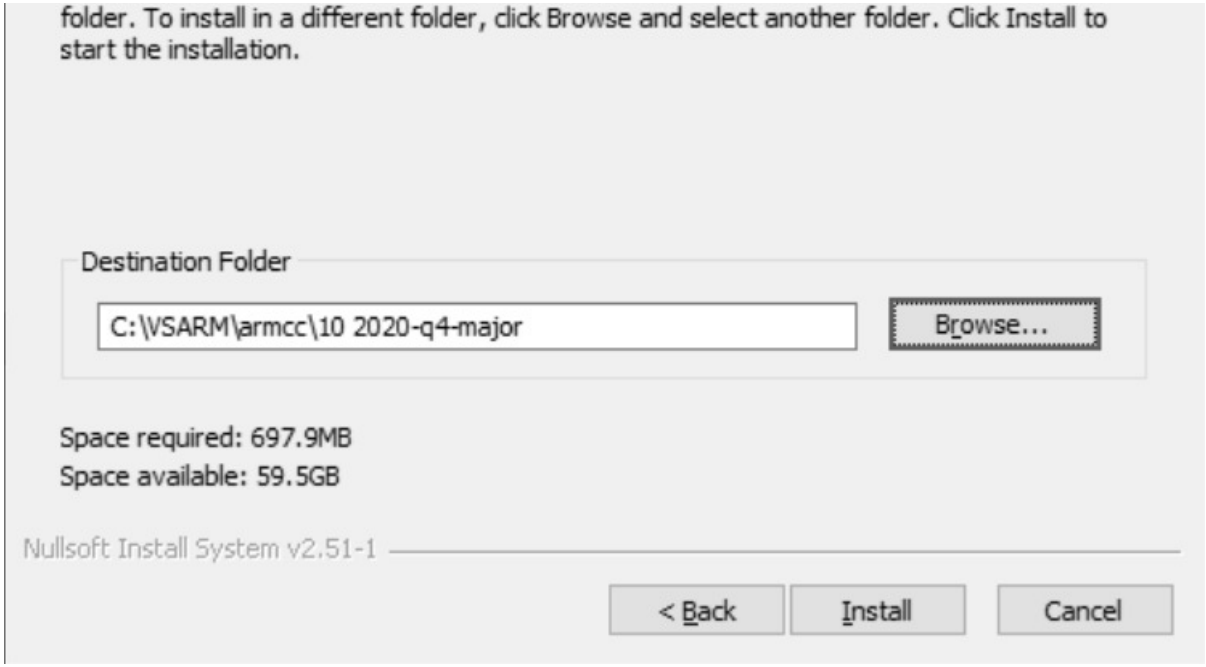
Install GNU Arm Embedded Toolchain

The GNU Arm Embedded Toolchain contains the Arm GCC compiler that we need to compile C and C++ code for the RP2040.

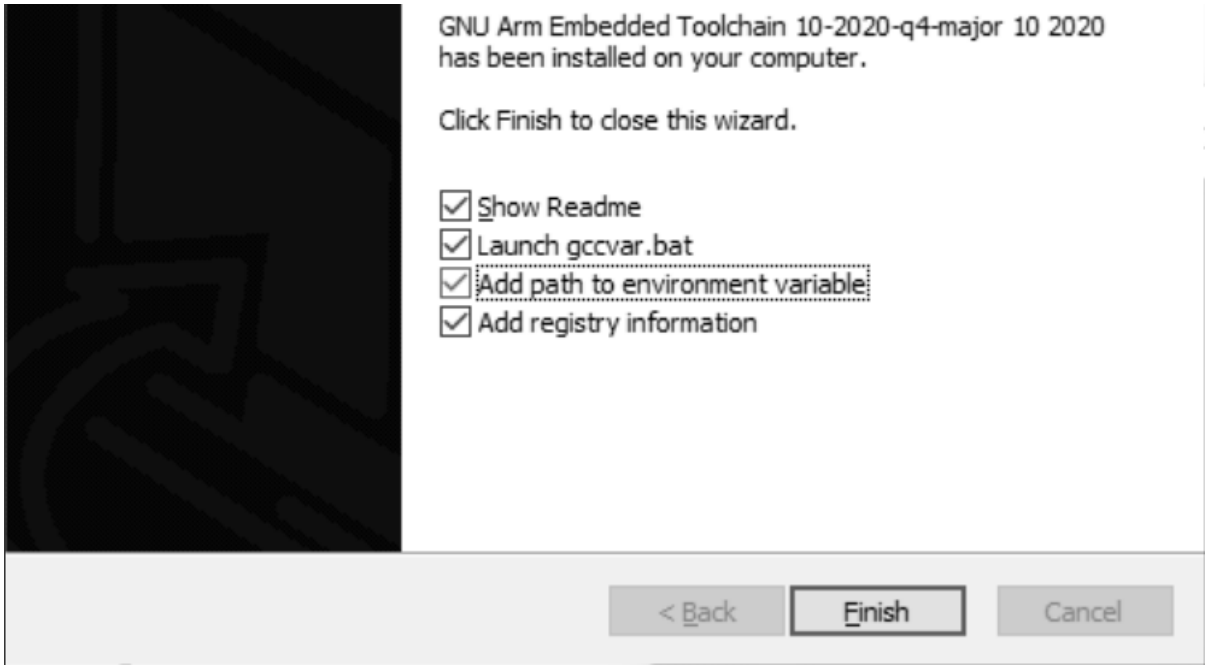
Head to the GNU Arm Embedded Toolchain download page and download the latest installer for Windows. For me, this was *gcc-arm-none-eabi-10-2020-q4-major-win32.exe*.

[Update Apr 19, 2022] I have verified that Arm GNU Embedded Toolchain 11.2-2022.02 works.

Run the installer. When asked, change the installation location to **C:\VSARM\armcc**. It will auto-fill the destination directory to the name of the current toolchain release.



Continue with the installation process. When it is complete, select the option to **Add path to environment variable**.



At this point, you should be able to call any of the Arm compiler tools in a new command prompt, such as *arm-none-eabi-gcc.exe*.

Install MinGW-w64 GCC Tools

MinGW (short for Minimalist GNU for Windows) is a collection of open-source utilities, such as compilers and linkers, that allow us to build applications for Windows.

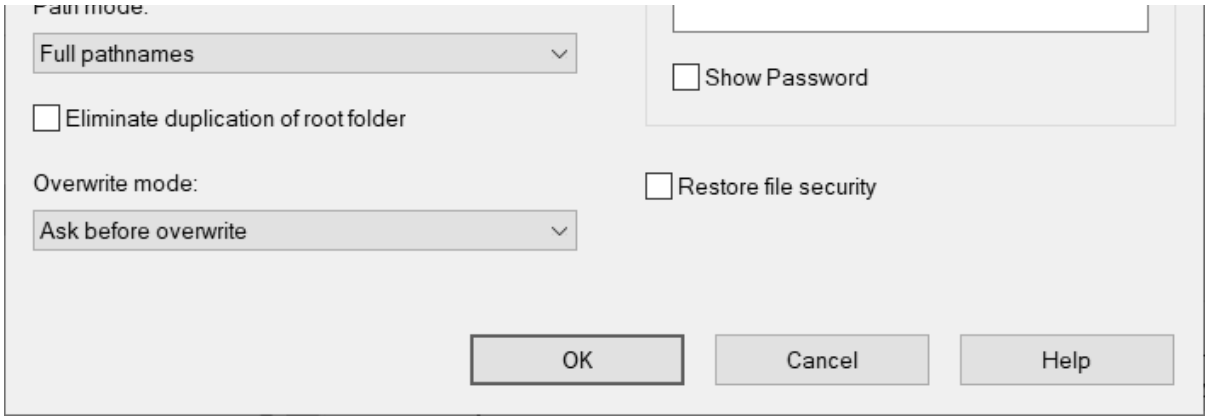
When we build Pico projects, we need to compile the *elf2uf2* and *pioasm* tools from source. These tools run on our computer (not the target RP2040 microcontroller), and

[Updated Apr 19, 2022] Note: at this time, the MinGW (.exe) installer appears to be broken. You will likely see the error message "The file has been downloaded incorrectly!" As a result, I have updated the portion below to download only the MinGW-W64 GCC files (tested with v8.1.0), as that's all we need to compile Pico programs.

Head to the MinGW-W64 files page on SourceForge: https://sourceforge.net/projects/mingw-w64/files/.

Download the **i686-posix-sjlj** zip file for your desired MinGW-W64 GCC version (e.g. 8.1.0).

Use 7-Zip to unzip it into the **C:\VSARM\mingw** directory. Uncheck the named unzip directory so that when everything unzips, you should have C:\VSARM\mingw\mingw32.



When it's done, open a Windows Command Prompt and enter the following into the terminal:

```
echo mingw32-make %* > C:\VSARM\mingw\mingw32\bin\make.bat
```

This creates a wrapper batch file that will call the *mingw32-make* tool whenever you type *make* into a Windows terminal. We will update the Windows Path to find all of the tools in *mingw32\bin* (along with this .bat file) in a later step.

Install CMake

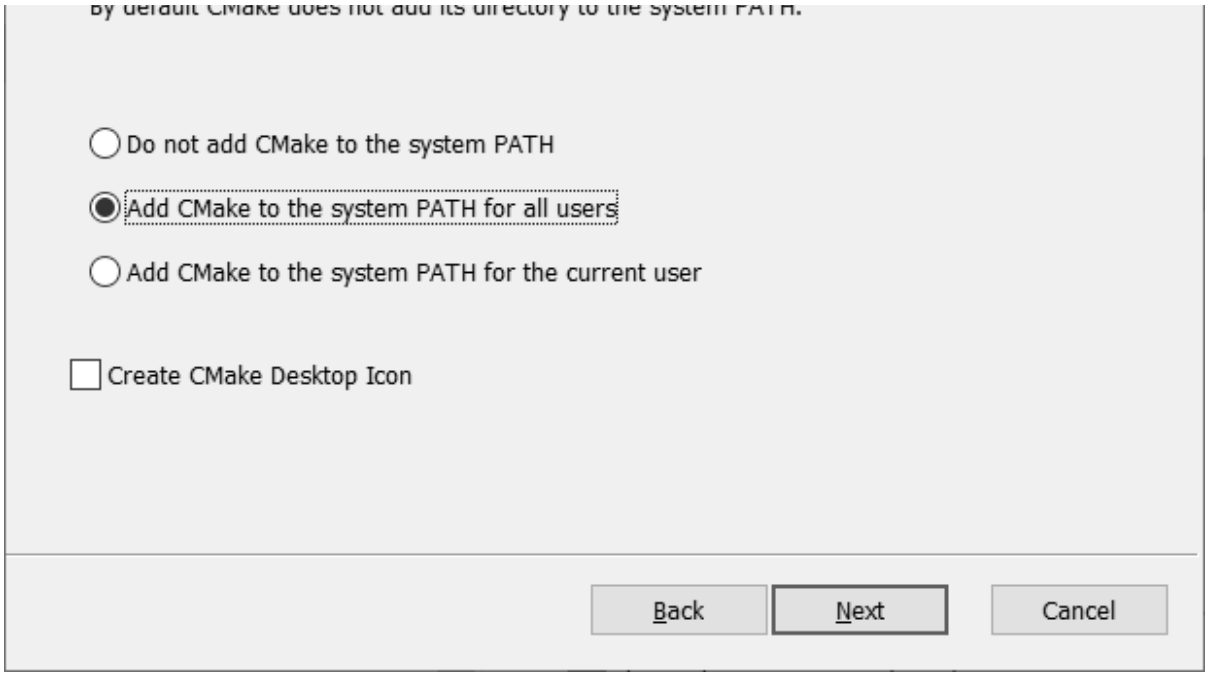
CMake is a tool that helps you automate the build process of programs. It does not build/compile (like Make does), but rather, it can generate the directory structures and

Important: There is a bug in CMake version 3.20 (at the time of writing). On the second run of make or nmake (after running cmake), the process will fail. If you're using nmake, you'll get an error like `fatal error U1033: syntax error : ':' unexpected`, or if you're using mingw32-make, something like `*** multiple target patterns.  Stop.` To prevent this, install **CMake version 3.19**. Future versions of CMake may fix this bug, but for now, know that version 3.19 worked for me.

[Update Apr 19, 2022] I have verified that CMake 3.23.1 now works. The bug has been fixed.

Download the version 3.19.8 installer for Windows (*cmake-3.19.8-win64-x64.msi*).

Run the installer and accept the user license. On *Install Options*, select **Add CMake to the system PATH for all users**.



Continue the installation process, accepting all the defaults. Note that this will install CMake to C:\Program Files\CMake, which is fine, as it will be used as a system-wide tool (not just for VSARM projects).

Install Python

The Pico SDK relies on Python to script and automate some of the build functions.

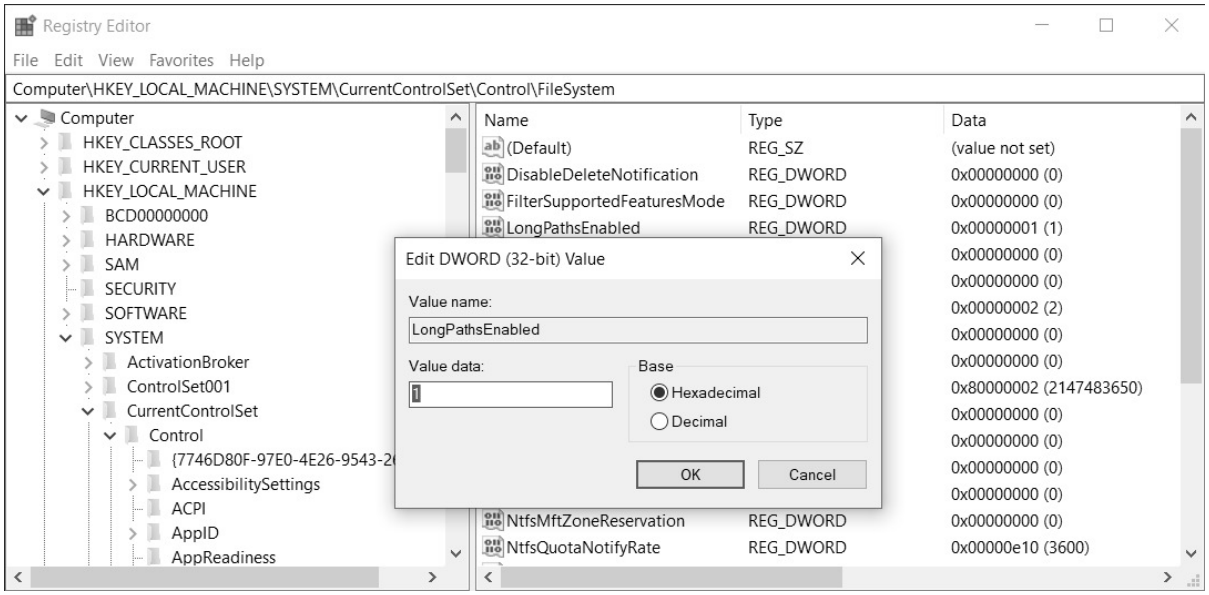
Important! At the time of writing, the Pico SDK recommends **Python version 3.9**. I do not know if other versions will work.

Run the installer. On the first screen, make sure that **Install launcher for all users (recommended)** is checked and check **Add Python 3.9 to PATH**.

Click **Install Now** and wait while it installs Python.

At the end of the installation process, select the option to **disable the MAX_PATH length limit**.

To enable long paths, search for **regedit** in the Windows search bar and run the **Registry Editor** program. Navigate to `Computer\HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\FileSystem` and add an entry (if one is not already there) for `LongPathsEnabled`. Change *Value data* to **1** and *Base* to **Hexadecimal**.

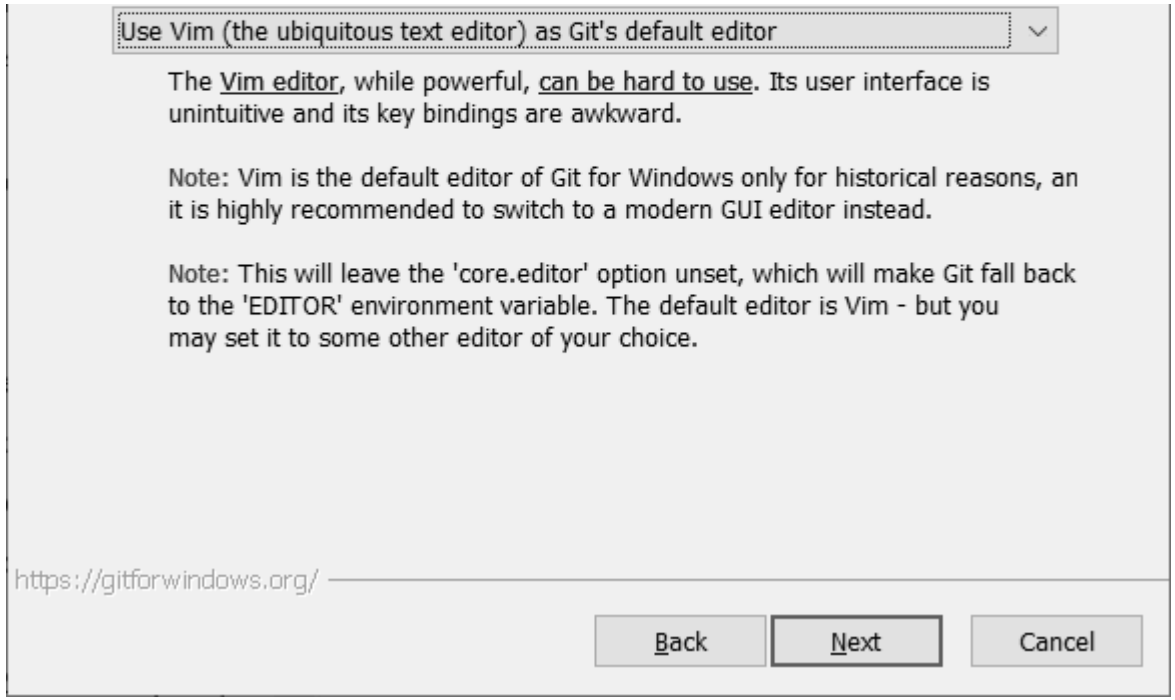


Here is a full guide if you need help modifying the registry to disable the pathname limit.

Install Git

(e.g. *32-bit Git for Windows Setup*).

Run the installer. When you get to the screen asking you to choose a default editor, feel free to pick whatever you want. I kept *vim* because I know it (barely) well enough to edit git comments.



Continue the installation process, accepting all the defaults.

Download Pico SDK and Examples

The Pico SDK conains a collection of tools and libraries used to make developing on the Pico (and RP2040) much easier. We can also download a set of C/C++ examples that are useful demonstrations of how to use the SDK.

To start, create a new folder named **pico** in C:\VSARM\sdk.

Open Git Bash and enter the following commands:

```
cd /c/VSARM/sdk/pico
git clone -b master https://github.com/raspberrypi/pico-sdk.git
cd pico-sdk
git submodule update --init
cd ..
git clone -b master https://github.com/raspberrypi/pico-examples
```

At this point, you should have all of the necessary build tools, SDK, and examples installed to start developing programs for the Raspberry Pi Pico and RP2040.

Update Environment Variables

Some of the tools we installed automatically updated the Windows environment variables (specifically, the Path). However, a few (like MinGW and the SDK) did not. We need to update the environment variables so that the shell and various build tools know where to find things in our filesystem.

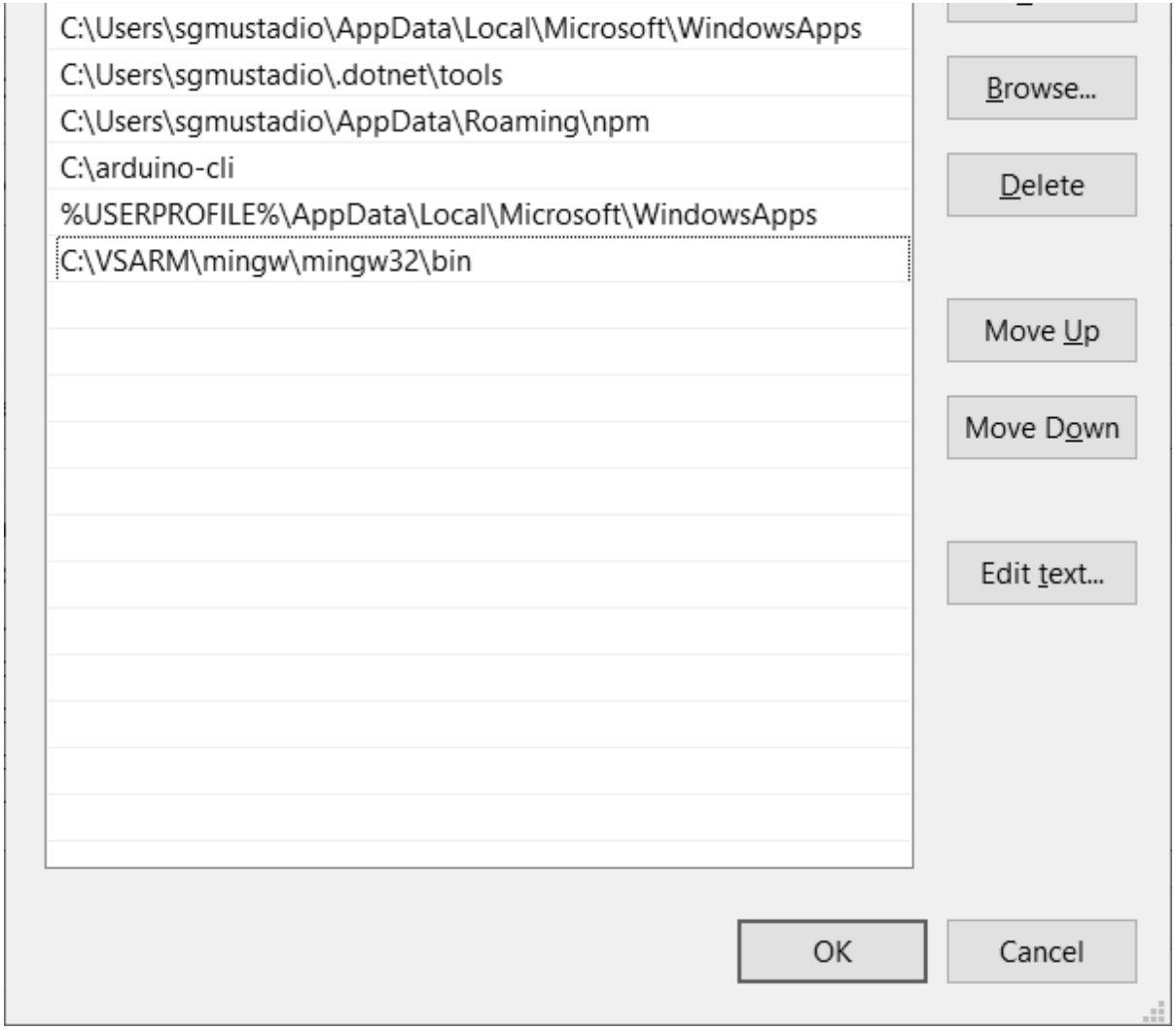
In the Windows search bar, enter **env**. Click on **Edit the system environment variables**.

In that window, click on **Environment Variables...**

Make sure you see the following entries listed:

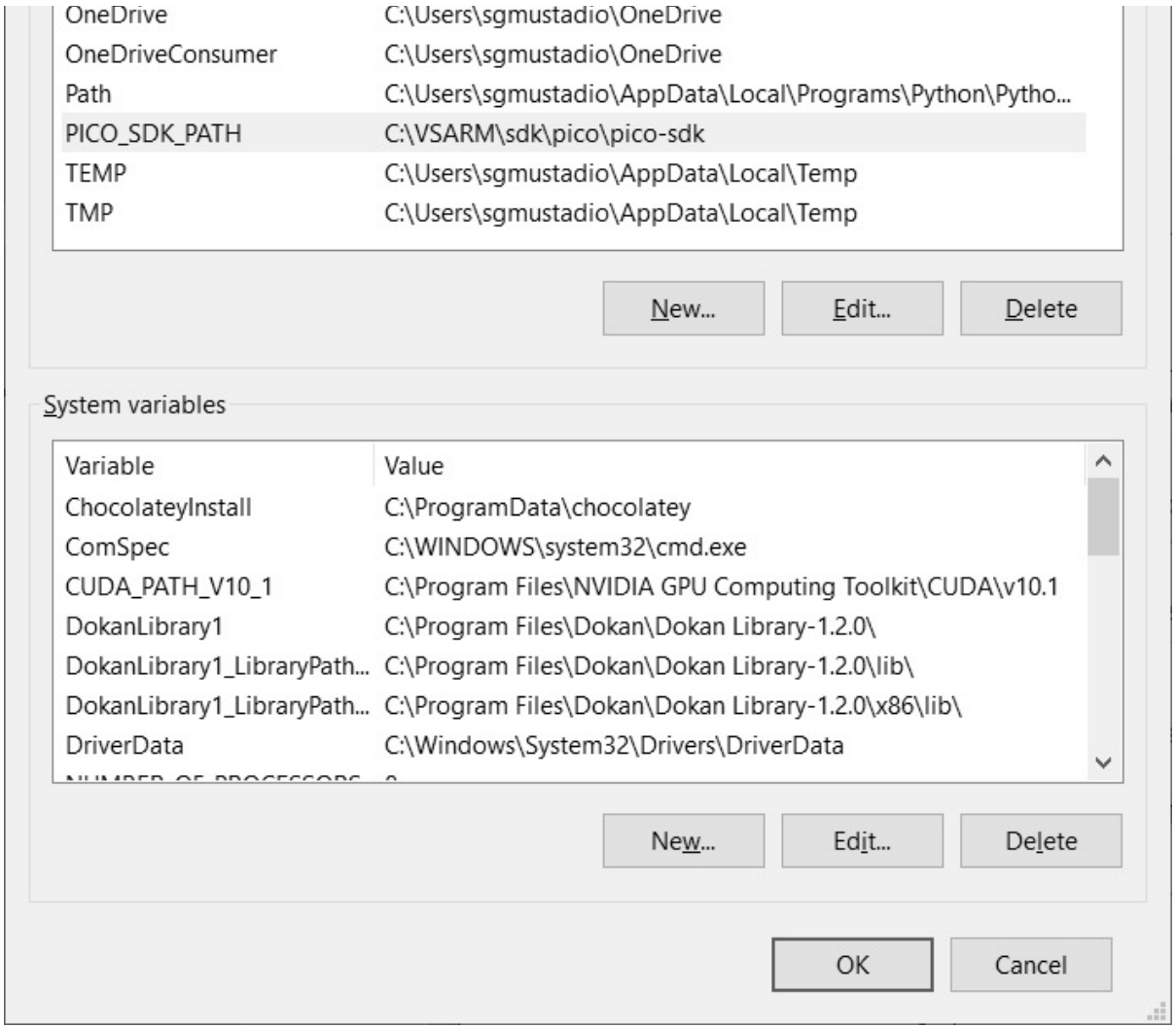
- C:\VSARM\armcc\<release version>\bin
- C:\VSARM\mingw\mingw32\bin

You might see an entry for Python39 if you chose to install Python for the current user (as I did).



Click **OK** to exit the user Path window.

Click **OK** to save the environment variable. At this point, your User Variables should have an updated Path as well as PICO_SDK_PATH variables.

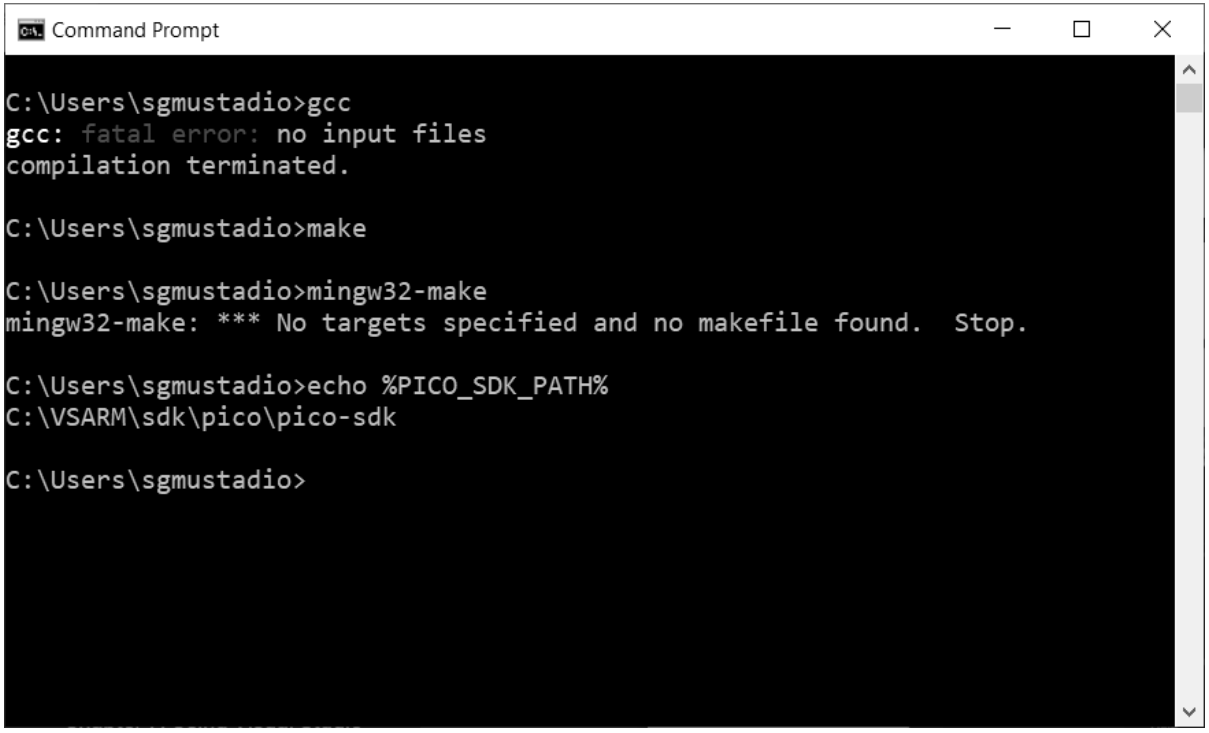


Under *System variables*, select **Path** and click **Edit**. Check to make sure you see the following entries (add them if you do not see them):

chose to install Python for all users.

Click **OK** on all 3 of the open windows to close them and save changes.

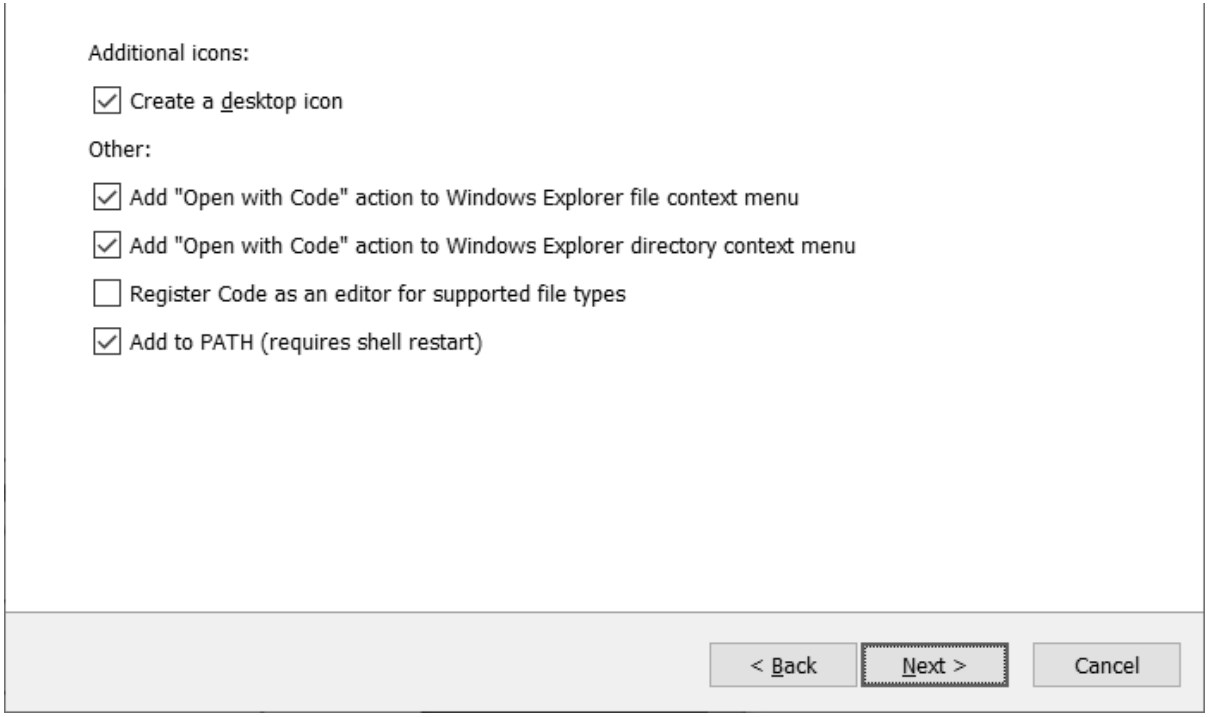
At this point, you should be able to open a commend prompt and enter commands like `gcc`, `make`, and `echo %PICO_SDK_PATH%` to make sure the environment variables were set correctly.



I'll only show the basics for now so you can get your VS Code on Windows to act like VS Code on other operating systems when working with the Pico.

Head to code.visualstudio.com and download the latest release for Windows.

Run the installer and accept all of the defaults. You're welcome to create a desktop icon and add "Open with Code" to the Windows Explorer context menu. I like to enable these options, but they're not necessary.



When the installer is done, it's time to test building a program for the Pico.

Build Blink Example

You should be able to build Pico code from any command prompt at this point. However, we're going to do so in VS Code to show how it might be done during development.

an alias for *mingw32-make.exe* in Git Bash. Enter the following commands (you will only need to do this once):

```
echo "alias make=mingw32-make.exe" >> ~/.bashrc
source ~/.bashrc
```

Build the blink example by entering the following commands into your terminal:

```
cd /c/VSARM/sdk/pico/pico-examples/
mkdir build
cd build
cmake -G "MinGW Makefiles" ..
cd blink
make
```

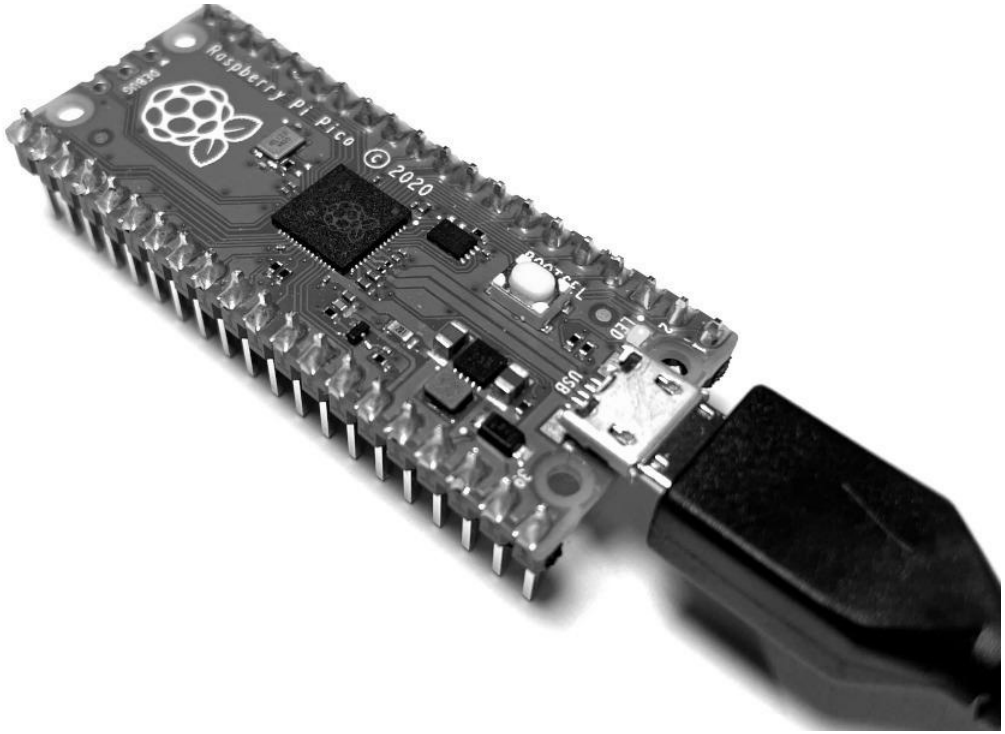
Important! Any time you call CMake from a terminal like this, you will need to specify "MinGW Makefiles" as the build system generator (-G option).

This should build the blink example and generate the .elf, .hex, and .uf2 binary files. We can easily upload the .uf2 file to our Pico without any special tools.

Put the Pico board into bootloader mode (press and hold the BOOTSEL button while plugging a USB cable into the Pico).

Find which drive letter the RPI-RP2 drive is mounted to (e.g. it was G: for me). Enter the following into Git Bash (change the drive letter as necessary):

```
cp blink.uf2 /g/
```



I hope this has helped you get started using the Raspberry Pi Pico SDK on Windows! I personally found that relying on MinGW for the compiler tools was easier than using Build Tools for Visual Studio.

From here, you can set up VS Code any way you like with various plugins like CMake Tools. I did not cover setting up debugging and other tools (picotool, gdb, OpenOCD, etc.), which are topics for another time. For now, I recommend referring to the Pico C/C++ Getting Started Guide to see how to configure those tools in VS Code.

C C++ RASPBERRY PI PICO TUTORIAL VS CODE WINDOWS

Share this Article

Previous Article
Custom Wake Word Part 1: Capturing Data

Next Article
How to Build OpenOCD and Picotool for the Raspberry Pi Pico on Windows

Author
ShawnHymel

85 thoughts on “How to Set Up Raspberry Pi Pico C/C++ Toolchain on Windows with VS Code”

gautom Bose on April 5, 2022 at 1:11 am REPLY

Excellent.
Is this process suitable for windows 10, x86, 32 bit
Pc.
Can you please confirm if it will work on a 32 bit windows 10 pc ?

ShawnHymel on April 15, 2022 at 4:53 pm REPLY

I don't have a 32-bit version of Windows to try, but I have to imagine it will work. Just make sure you install the 32-bit version of the tools.

irawan on April 19, 2022 at 10:59 pm REPLY

thank you for detailed explanation…

vidal on May 23, 2021 at 10:06 am REPLY

ShawnHymel on May 23, 2021 at 3:33 pm REPLY

Yes, I cover using plugins for one-click CMake and building in this video: https://www.youtube.com/watch?v=B5rQSoOmR5w. This post was made mostly as a supplement to that video, as setting up the toolchain in Windows is such a pain ☺

Smith on May 27, 2021 at 5:55 am REPLY

Just want to say that this was exactly what I needed! Awesome job, especially with the details and the screenshots.

ShawnHymel on May 27, 2021 at 2:03 pm REPLY

I'm glad to hear it helped!

dmytro on June 1, 2021 at 11:28 pm REPLY

Pena on June 13, 2021 at 8:50 am REPLY

Thanks Shawn. After couple installation rounds I finally managed to compile and load blink.uf2 on my picoboard. I got this error : “bash: $’\377\376alias’: command not found” when I opened Git Bash from Desktop. I had to clean all files from my home direcotory. Yes I had previous VSCode installations and others too before starting your project. But now everything works fine and I am going to proceed with your tutorials.

Abe on April 28, 2022 at 3:29 pm REPLY

Hi, How did you fix the error “bash: $’\377\376alias’: command not found”?
What did you click in order to clean files from the home directory? (Im new to all this.)

Thanks!

Stan on June 13, 2021 at 11:11 pm REPLY

Thank you. It is very helpful and clear.

bitluni on June 24, 2021 at 11:03 pm REPLY

I cant find make only mingw32-make.exe after the path configuration

ShawnHymel on June 25, 2021 at 4:03 pm REPLY

Did you create an alias for make as shown in the “Build Blink Example” section?

bitluni on June 24, 2021 at 11:09 pm REPLY

It works… this is the first tutorial that works for me. Thanks!

Max on June 26, 2021 at 5:48 pm REPLY

Wanted to put a comment on here — my make was dying on the elf2uf2 step of the makefile. If the same thing is happening to you, you will get a code '-1073741511' in this scenario.

The fix was a direct copy of the mingw32 libstdc++-6.dll into the created build/elf2uf2 directory, alongside elf2uf2.exe

There was a silent dll failure buried in there, which will open up a dialog only if you run the tool on its own whiel this error is present.. This is a fairly mature system. I don't know which libstdc++-6.dll it was /trying/ to use, but the quickest way to get rid of it is to move the library into that same location.

ShawnHymel on July 22, 2021 at 2:53 pm REPLY

Thank you for pointing this out!

Byron on April 15, 2022 at 1:42 pm REPLY

The thing that worked for me was to insure that the path to the MinGW bin directory was in my system Path variable (ie. C:\VSARM\mingw\minggw64\bin).

One was the “Error -1073741511” issue Zoot mentioned above and his fix worked. I think I ended up copying libstdc++-6.dll to C:\VSARM\sdk\pico\pico-examples\build\ .

The other issue was setting up VS Code to run git bash in the terminal. Apparently, the “new” version of VS Code I’m running (2022-05-17718…) has changed that method and the instructions above no longer work. I fumbled around with a couple of suggestions on stackoverflow and gave up.

The chain runs fine by launching git bash directly from start menu and typing make.

I’m happy. You got me to a place that works after trying several other site instructions. I think it’s a mistake for the RPi Foundation to make so much effort getting the Pico SDK instructions tied around the Pi, especially with the total lack of Pi’s in the market now.

Thanks!

Krzysztof on July 7, 2021 at 10:58 am REPLY

ShawnHymel on July 22, 2021 at 2:51 pm REPLY

I don't have that board, so I won't be able to help. Your best bet would be to ask for help from SparkFun on the product page (https://www.sparkfun.com/products/16985) or the forums (https://forum.sparkfun.com/).

Jonnyboy on July 22, 2021 at 10:46 am REPLY

Have followed your tutorial to the letter, but with VS Code already previously installed. When I try to build the 'blink' example, I get the following error:

mingw32-make[2]: *** [blink\CMakeFiles\blink.dir\build.make:812: blink/blink.elf] Error -1073741511
mingw32-make[2]: *** Deleting file 'blink/blink.elf'
mingw32-make[1]: *** [CMakeFiles\Makefile2:3723: blink/CMakeFiles/blink.dir/all] Error 2
mingw32-make: *** [Makefile:90: all] Error 2

Any ideas what the problem could be?

ShawnHymel on July 22, 2021 at 2:54 pm REPLY

I just noticed a comment from someone else with the same issue. Here's the proposed fix:

"The fix was a direct copy of the mingw32 libstdc++-6.dll into the created build/elf2uf2 directory, alongside elf2uf2.exe"

Jonnyboy on July 23, 2021 at 6:32 am REPLY

Aha! Perfect! Thank you – that fixes the issue, I can build the examples now. Thanks for an excellent tutorial too. 🙂

Lucas on March 23, 2022 at 11:54 am REPLY

This worked for me too. Had a few problems locating libstdc++-6.ddl but found it under C:\VSARM\mingw\mingw64\x86_64-w64-mingw32\lib32. I also had a few problems with the sourceforge installer as it currently doesn't work and presents me with the message "the

luc on July 31, 2021 at 5:55 pm REPLY

Hi Shawn,
Many thanks for the explanation. Learned from it.
An alternative when you want to de Arduino development on windows for the Pi Pico you could consider start using the VsCode / PlatformIO. This combination does also support the Pi Pico. All toolchain stuff is done for you by PlatformIO.

Kind regards, Luc Looman

Luc Looman on August 4, 2021 at 6:14 am REPLY

A little rectification needed on my Rely.
Indeed installing and running blink example in win10/vscode/platformio worked very quick.
Unfortuneatly this is a Arduino-mbed implementation which means you can not use the full Pico C++ SDK.
There is hope it will be supported soon with earlephilhower/arduino-pico.

$ cmake -G “MinGW Makefiles” ..
bash: cmake: command not found

Thank you

ShawnHymel on August 13, 2021 at 8:41 pm REPLY

It looks like cmake is not in your system PATH. When you installed CMake (https://cmake.org/download/), did you click the option to “Add CMake to system PATH for all users”? If not, check to make sure “C:\Program Files\CMake\bin” is included in the Path variable under Environment Variables > System Variables.

Tarik Bell on August 16, 2021 at 3:51 pm REPLY

It works now after I installed the last version of cmake-3.21.1-windows-x86_64.msi.
My Pico is blinking Thank you.

tortik92 on September 17, 2021 at 9:00 pm REPLY

Hi Shawn!

After executing “cmake -G “MinGW Makefiles” ..” in Git Bash Terminal of VS Code I get CMake Error:

“Using PICO_SDK_PATH from environment (‘ C:\VSARM\sdk\pico\pico-sdk’)
CMake Error at pico_sdk_import.cmake:52 (message):
Directory ‘C:/VSARM/sdk/pico/pico-examples/build/
C:/VSARM/sdk/pico/pico-sdk’ not found
Call Stack (most recent call first):
CMakeLists.txt:4 (include)
— Configuring incomplete, errors occurred!”

This path is existing and I can see it in explorer….

Thank you!

ShawnHymel on September 18, 2021 at 2:33 pm REPLY

tortik92 on September 18, 2021 at 9:30 am REPLY

Hi Shawn!

After executing “cmake -G “MinGW Makefiles” ..” in Bash in VS Code I recieve an error:

CMake Error at pico_sdk_import.cmake:52 (message):
Directory ‘C:/VSARM/sdk/pico/pico-examples/build/
C:/VSARM/sdk/pico/pico-sdk’ not found
Call Stack (most recent call first):
CMakeLists.txt:4 (include)
— Configuring incomplete, errors occurred

This directory exists and I can open it in explorer…

Thank you!

Patrik Källback on September 23, 2021 at 6:42 am REPLY

I was using the windows command prompt as the terminal, than the make was working perfectly. Wieard, but then I decided not to think more about it… just running make in the windows command prompt.

David Marshall on November 2, 2021 at 1:50 am REPLY

The link to “ming-w64” is broken.

…..Head to the downloads page on the Mingw-w64 project site, and go to the Mingw-w64 Builds page for Windows: http://mingw-w64.org/doku.php/download/mingw-builds.

ShawnHymel on November 2, 2021 at 3:44 am REPLY

Fixed. Thanks!

David Marshall on November 2, 2021 at 1:44 pm REPLY

Is this the correct MinGW ,
https://sourceforge.net/projects/mingw-

ShawnHymel on November 2, 2021 at 2:10 pm REPLY

Yes, I believe that is the correct one. It looks like they changed the installer slightly, but if you go through the steps, it should install ming-w64 as shown in the tutorial.

David Marshall on November 5, 2021 at 6:58 pm

The tutorial works for me, I am having an issue with an older win7 machine.

Przemysław Kurzak on November 12, 2021 at 9:35 pm REPLY

Hi guys, I got this issue after calling make:

mingw32-make[2]: *** [pico-sdk\src\rp2_common\boot_stage2\CMakeFiles\bs2_default_padded_checksummed_asm.dir\build.make:72: pico-sdk/src/rp2_common/boot_stage2/bs2_default_padded_checksummed.S] Error 1
mingw32-make[1]: *** [CMakeFiles\Makefile2:4091: pico-sdk/src/rp2_common/boot_stage2/CMakeFiles/bs2_default_padded_

2

John on November 17, 2021 at 5:32 pm REPLY

Thank you, it is really helpful for the installing!
But there is a problem, the build is already complete but when I was trying to flash it to rp2040, it didnt blink. Can you help me?

1

ShawnHymel on November 17, 2021 at 6:39 pm REPLY

I'm not sure why it's not blinking for you. Are you seeing any errors? Did you copy the .uf2 file? Can you verify that the LED will blink using other methods (e.g. MicroPython)?

2

Alexander Kirillov on November 20, 2021 at 3:50 pm REPLY

Thanks a lot – great tutorial!!
Two comments:
1. In section "installing Mingw", where you say "head to the downloads page on the Mingw-w64 project site, and go to the Mingw-w64 Builds page for Windows", your link is broken – link text is correct, but link URL is a placeholder:

VScode, deleted Git Bash terminal in VScode and launched it again, everything worked. Apparently VScode didn't read all env variables on first launch…

ShawnHymel on November 20, 2021 at 9:36 pm REPLY

Thanks for the heads up! I really don't know why the link didn't take in WordPress…I updated it, and it should work now. That's weird about VSCode not reading the environment variables, but good to know about needing to relaunch it.

Serhii on November 22, 2021 at 8:27 pm REPLY

Thank you for the excellent guide
Everything works

Only one note that I'd add

echo mingw32-make %* > C:\VSARM\mingw\mingw32\bin\make.bat

Alex on December 9, 2021 at 6:01 pm REPLY

You're a lifesaver! This is so much better than the toolchain in the Pico user guide (which was giving me opaque build errors)

Richard W on December 14, 2021 at 4:29 pm REPLY

Thank you very much, I've been struggling for days trying to use the extensions in VScode, your method worked seamlessly!

David Marshall on December 15, 2021 at 6:29 pm REPLY

For some reason "GIT BASH" is not available in the terminal. I've removed VS code and followed your method.

Robbok on December 18, 2021 at 4:39 pm REPLY

Hi there, I know you spent a ton of time making this page and being thorough in showing every aspect of what needed to be done to get these tools up and working. I just wanted to say THANK YOU. You saved me earlier this week when I needed to compile some files for my

ShawnHymel on December 18, 2021 at 6:11 pm REPLY

Glad it helped! I ran into the same issue–there was very little information on how to set up the build environment for the Pico on Windows.

Dave on January 12, 2022 at 8:33 pm REPLY

Jan 12 2022

I've made it to the step where visual studio code is installed. When visual studio code first runs, I encounter a dialog box as follows:

"Unable to determine what CMake generator to use. Please install or configure a preferred generator, or update settings.json, your Kit configuration or PATH variable."

In a command prompt, cmake can be started:

Microsoft Windows [Version 10.0.19044.1415]
(c) Microsoft Corporation. All rights reserved.

In the user's path environment variable, the following value is set:

C:\Program Files\CMake\bin

Would be very grateful for advice as to how to proceed.

Thank you!

Dave

ShawnHymel on January 12, 2022 at 9:41 pm REPLY

I have not seen that error yet. Maybe something in this thread can help? https://github.com/microsoft/vscode-cmake-tools/issues/880

James on January 14, 2022 at 5:11 am REPLY

Thank you Hymel.
However, I needed to restart my computer twice, for some reason I guess, the environmental variables didn't apply immediately.

preferred generator, or update settings.json, your Kit configuration or PATH variable."

I was able to resolve this by uninstalling visual studio code, and then (important) removing two directories which had retained state information from visual studio code. I then re-installed visual studio code and the start-up error messages were not present.

I continued in the installation procedure to this point:

```
echo "alias make=mingw32-make.exe" >> ~/.bashrc
source ~/.bashrc
```

The command "source ~/.bashrc" returns the following error:

```
$ source ~/.bashrc
bash: alias: alias: not found
```

I continued past this error without resolution.

I was able to complete the rest of the demonstration resulting in a working "blink LED" example.

to bash. I would prefer bash permanently.

Thank you again. After many hours this week with failed installation attempts using (apparently untested) documentation from raspberry pi foundation, your procedure produced my first end-to-end working result.

ShawnHymel on January 15, 2022 at 3:11 pm REPLY

Hi Dave,

1. Check the spelling and syntax of your alias command with `$ cat ~/.bashrc` Make sure that alias is spelled correctly, and that there are no spaces around the equals sign (see here for more info: https://askubuntu.com/questions/391518/bash-alias-alias-not-found).
2. You can make Git Bash the default terminal. Press `F1` and type "Terminal: Select Default Profile." From the drop-down list, select Git Bash to be your default terminal. You can see screenshots of this done here: https://stackoverflow.com/questions/44435697/vscode-change-default-terminal

Hope that helps!

***
CMake Warning at D:/users/vivek/work/RPi-PICO/projects/pico-sdk/cmake/preload/toolchains/find_compiler.cmake:22 (message
):
PICO_TOOLCHAIN_PATH specified (D:\"Program Files (x86)\GNU Tools ARM
Embedded\10 2021.10''\bin), but arm-none-eabi-gcc not found there
...
CMake Error at D:/users/vivek/work/RPi-PICO/projects/pico-sdk/cmake/preload/toolchains/find_compiler.cmake:28 (message)

Compiler 'arm-none-eabi-gcc' not found, you can specify search path with
"PICO_TOOLCHAIN_PATH".
***

1

Oke Petersen on March 24, 2022 at 10:22 pm REPLY

Hi I encountered the same issue. The solution forme was to add another System variable:
Name :PICO_TOOLCHAIN_PATH
Path: C:\VSARM\armcc\102021.10\bin (or wherever you put your gcc)

Oke

William on January 26, 2022 at 10:36 pm REPLY

Fantastic tutorial! I'm so looking forward to this working, but I get the following error….

CMake Error at C:/Program Files/CMake/share/cmake-3.22/Modules/CMakeTestCCompiler.cmake:69 (message):
The C compiler
"C:/build/17.0/arm/sysroots/i686-nilrtsdk-mingw32/usr/libexec/arm-nilrt-linux-gnueabi/gcc/arm-nilrt-linux-gnueabi/4.9.2/gcc.exe"
is not able to compile a simple test program.

I've followed you instructions and all went well up to the point of compiling blink… any help appreciated

Sebastian Abril on January 31, 2022 at 4:54 am REPLY

Thanks for the tutorial!!! I have a question that I still can't solve, how can I set in visual code a different RP2040 based board, such as the adafruit

Thank you very much!

Thomas on February 22, 2022 at 12:02 am REPLY

It appears that there is a very recent issue with the Ming-w online installer. The error says “The file has been downloaded incorrectly!” It seems to be very recent and many have commented on Source Forge. Is there any solutions/ways around this?

Juan David Medina Tobon on March 4, 2022 at 5:46 pm REPLY

Hi,

I followed the answer from this post: https://stackoverflow.com/questions/46455927/mingw-w64-installer-the-file-has-been-downloaded-incorrectly. Apparently, there’s a problem with the installer.

Download the MinGW archive from https://sourceforge.net/projects/mingw-w64/files/mingw-w64/. I downloaded the one named “x86_64-posix-sjlj”. Extract the file and

1

William Stark on March 6, 2022 at 4:38 am REPLY

This Worked for me MingGw 64 was a problem, had to do same and use “x86_64-posix-sjlj”. also and changing the path as such, path was a problem in test and rebooting changed that for sdk example dir. blink and the path for mingw32. Using the new version ? just confused me. blink worked. moving forward to other and RTOS.
more BEER.

2

Lucas on March 24, 2022 at 4:02 pm REPLY

This workaround worked for me. Only had one more minor setback after this while running make for the blink example. I was prompted with the following error:

mingw32-make[2]: ***
[blink\CMakeFiles\blink.dir\build.make:812: blink/blink.elf]
Error -1073741511
mingw32-make[2]: *** Deleting file ‘blink/blink.elf’
mingw32-make[1]: *** [CMakeFiles\Makefile2:3723:

the build/elf2uf2 folder.

Michael Filgate on March 2, 2022 at 12:02 pm REPLY

Thanks for the tutorial, I now have a blinky pico !
Just a heads-up, I had to type “make.bat” on windows, for some reason just typing “make” resulted in nothing and had me chasing around my PATH variable.

JK_student on March 21, 2022 at 9:04 am REPLY

Hello,

I have question. I would like write permamently program on my RP2040. I wrote code in circuitpython. I need RP2040 which when it will be plug in to computer, It send data from RP2040 to computer by UART.
How Can I do it?

I think it’s the same work as https://www.youtube.com/watch?v=IMZUZuytt7o , but in circuitpython and to RP20404.

$ cmake -G “MinGW Makefiles” ..
CMake Error at pico_sdk_import.cmake:44 (message):
SDK location was not specified. Please set PICO_SDK_PATH or set PICO_SDK_FETCH_FROM_GIT to on to fetch from git.
Call Stack (most recent call first):
CMakeLists.txt:4 (include)

— Configuring incomplete, errors occurred!

Nino on April 1, 2022 at 11:55 am REPLY

Hey Shawn

Followed your tutorial… all worked seemlessly but as i try to run the command (cmake -G “MinGW Makefiles” ..) i get these errors:
CMake Error: CMAKE_C_COMPILER not set, after EnableLanguage
CMake Error: CMAKE_CXX_COMPILER not set, after EnableLanguage
CMake Error: CMAKE_ASM_COMPILER not set, after EnableLanguage
— Configuring incomplete, errors occurred!

do you know what the problem could be?

H' Shawn\

I tracked and used step that you mentioned. I get error about below.
Could you please help me
[cmake] PICO_SDK_PATH is C:/Pico/pico-sdk
[cmake] PICO platform is rp2040.
[cmake] CMake Error at CMakeLists.txt:8 (project):
[cmake] Running
[cmake]
[cmake] 'nmake' '-?'
[cmake]
[cmake] failed with:
[cmake]
[cmake] The system cannot find the file specified
[cmake]
[cmake]
[cmake] — Configuring incomplete, errors occurred!
[cmake] See also "C:/Pico/blink/build/CMakeFiles/CMakeOutput.log".

my cmakelists.txt

# Set minimum required version of CMake
cmake_minimum_required(VERSION 3.12)

```
set(CMAKE_CXX_STANDARD 17)

# Creates a pico-sdk subdirectory in our project for the libraries
pico_sdk_init()

# Tell CMake where to find the executable source file
add_executable(${PROJECT_NAME}
main.c
)

# Create map/bin/hex/uf2 files
pico_add_extra_outputs(${PROJECT_NAME})

# Link to pico_stdlib (gpio, time, etc. functions)
target_link_libraries(${PROJECT_NAME}
pico_stdlib
)

# Enable usb output, disable uart output
pico_enable_stdio_usb(${PROJECT_NAME} 1)
pico_enable_stdio_uart(${PROJECT_NAME} 0)
```

[cmake] PICO_SDK_PATH is C:/Pico/pico-sdk
[cmake] PICO platform is rp2040.
[cmake] CMake Error at CMakeLists.txt:8 (project):
[cmake] Running
[cmake]
[cmake] 'nmake' '-?'
[cmake]
[cmake] failed with:
[cmake]
[cmake] The system cannot find the file specified
[cmake]
[cmake]
[cmake] — Configuring incomplete, errors occurred!
[cmake] See also "C:/Pico/blink/build/CMakeFiles/CMakeOutput.log".

my cmakelists.txt

```
# Set minimum required version of CMake
cmake_minimum_required(VERSION 3.12)

# Include build functions from Pico SDK
include($ENV{PICO_SDK_PATH}/external/pico_sdk_import.cmake)

# Set name of project (as PROJECT_NAME) and C/C++ standards
project(blink C CXX ASM)
```

```
# Tell CMake where to find the executable source file
add_executable(${PROJECT_NAME}
main.c
)

# Create map/bin/hex/uf2 files
pico_add_extra_outputs(${PROJECT_NAME})

# Link to pico_stdlib (gpio, time, etc. functions)
target_link_libraries(${PROJECT_NAME}
pico_stdlib
)

# Enable usb output, disable uart output
pico_enable_stdio_usb(${PROJECT_NAME} 1)
pico_enable_stdio_uart(${PROJECT_NAME} 0)
```

Ibrahim on April 4, 2022 at 10:51 pm REPLY

To fix this error:
mingw32-make[2]: *** [blink\CMakeFiles\blink.dir\build.make:812:
blink/blink.elf] Error -1073741511
mingw32-make[2]: *** Deleting file ‘blink/blink.elf’

“C:\VSARM\mingw”. There you will have a folder called “mingw64”. You either set the environment variable to “C:\VSARM\mingw\mingw64\bin”, or keep it as is “C:\VSARM\mingw\mingw32\bin” but rename unpacked folder to mingw32 (a hack !)
It took me many days before I found a solution. Thanks God !

Ibrahim on April 5, 2022 at 8:24 pm REPLY

You will need to append the following line to your ./.vscode/settings.json file:
“cmake.generator”: “MinGW Makefiles”,

HuyLE on April 27, 2022 at 5:24 pm REPLY

I tested arm-non-eabi-gdb.exe of the latest version Arm GNU Embedded Toolchain 11.2-2022.02 and it doesn't work.
I have no problem on gdb with Arm GNU Embedded Toolchain 10.3-2021.10.

Mohamed on May 1, 2022 at 5:42 pm | REPLY |

Hi
Very good tutorial, I did like it… But I couldn't succeed in trying it.
Here is the the error output I get:

moham@DESKTOP-OTSLFLL MINGW64 /c/VSARM/sdk/pico/pico-examples (master)
$ cd build

moham@DESKTOP-OTSLFLL MINGW64 /c/VSARM/sdk/pico/pico-examples/build (master)
$ cmake -G "MinGW Makefiles" ..
PICO_SDK_PATH is C:/VSARM/sdk/pico/pico-sdk
PICO platform is rp2040.
PICO target board is pico.
Using board configuration from C:/VSARM/sdk/pico/pico-sdk/src/boards/include/boards/pico.h
TinyUSB available at C:/VSARM/sdk/pico/pico-sdk/lib/tinyusb/src/portable/raspberrypi/rp2040; enabling build support for USB.
— Configuring done
— Generating done
— Build files have been written to: C:/VSARM/sdk/pico/pico-examples/build

$ make
process_begin:
CreateProcess(C:\Users\moham\AppData\Local\Temp\make2468-1.bat, C:\Users\moham\AppData\Local\Temp\make2468-1.bat, …) failed.
make (e=2): The system cannot find the file specified.
mingw32-make: *** [Makefile:1211: cmake_check_build_system] Error 2

moham@DESKTOP-OTSLFLL MINGW64 /c/VSARM/sdk/pico/pico-examples/build/blink (master)
$

Of course the folders structure and the variables environment are set as described by you.

Any advice on the issue?

ShawnHymel on May 2, 2022 at 2:00 am REPLY

Hi Mohamed,

It looks like your system cannot find mingw32-make. I recommend checking that C:\VSARM\mingw\mingw32\bin is on your PATH

Shubham on May 23, 2022 at 5:15 am REPLY

I deleted the mingw path from user and saved the path in system variables
then restarted the system and it started working and making the build files

Shubham on May 23, 2022 at 5:20 am REPLY

I moved the path of mingw from user variables to system variables'
then I restarted my system
and 'make' started working

Ade on May 10, 2022 at 4:43 pm REPLY

With the current (3.23.1, as of May 2022) version of CMake, on Win10, I had to add
cmake_policy(SET CMP0057 NEW)
at the top of my CMakeLists.txt file to get it to work.

Hope this might help others....

thanks for the great detailed tutorial.

I have an problem. After typing typing the last command or step with "make" into the git bash terminal of VS i get the following message:

@user_pc /c/VSARM/sdk/pico/pico-examples/build/blink $ make
bash: make: command not found

Do you have an idea whats wrong with this? I followed every step from your tutorial, I think.
Would be greatful if you help me here out

ShawnHymel on May 27, 2022 at 2:49 pm REPLY

This is likely because you did not create make.bat in C:\VSARM\mingw\mingw32\bin or you forgot to put C:\VSARM\mingw\mingw32\bin on your PATH in environment variables. You might want to try the "make" command in other terminals to see if Windows really does see it on your path.

apfelschorle on May 27, 2022 at 4:10 pm REPLY

source ~/.bashrc

Started directly with next steps

apfelschorle on May 27, 2022 at 9:01 pm REPLY

What have I to do if I want to add a new project into th pico-examples folder? I created a folder into the pico-examples folder. But git bash cant find the folder.

bash: cd: test: No such file or directory

Leave a Comment

Your email address will not be published. Marked fields are required.

Name *

Email *

Website

I'm not a robot

reCAPTCHA

Privacy - Terms

SUBMIT