PICO SDK On Windows Guide

This project aims to create an easy-to-use installer to get started on Windows (using the C/C++ SDK) with Raspberry Pi Pico and other RP2040-based boards. It is inspired by, and is roughly equivalent to, the pico-setup project for Linux systems.

Once you have finished configuring the PICO SDK, you can recursively clone our product Repo to get an almost complete experience, but since some of the code doesn't compile properly on Windows, there are still features that are missing, pending further maintenance of the code by the Raspberry Pi Foundation.

The installer automates the prerequisite installation on Windows, as explained in the official Getting started with Raspberry Pi Pico guide.

The rest of this README document is about developing the installer itself. If you just want to install and use the compilers and toolchain, download the release linked to above. Further information for users is in the tutorial.

Included software

- Arm GNU Toolchain
- CMake
- Ninja

- Python 3.9
- Git for Windows
- Visual Studio Code
- OpenOCD

Building

You will need PowerShell 7.2 or newer in order to run the build script. See Installing PowerShell on Windows for download links and instructions. Note that this is needed only for building the installer itself, and not for end-user machines where the installer will be run.

The installers are built with NSIS 3. The NSIS script is generated by build.ps1 when provided with a JSON configuration file. The build script automatically downloads a local copy of NSIS to use for the build.

Only x86-64 builds are supported at this time.

Compiling OpenOCD and other tools (picotool, pioasm, elf2uf2) requires an installation of MSYS2. The build script automatically downloads and installs a local copy of MSYS2. You can specify a path to an existing copy of MSYS2 using the `-MSYS2Path` option. The build script will install a copy of MSYS2 at this path if it doesn't find an existing copy.
It is highly recommended to use a dedicated copy of MSYS2 for this build.

To build:

```
.\build.ps1 .\config\x64-standalone.json -MSYS2Path ~\Downloads\msys64
```

The built installers will be saved to the `bin` directory.

Tests

There are tests for some parts of the build scripts. You can run them like this:

```
Install-Module Pester -Force
Import-Module Pester -PassThru # Check the version of the imported module -
- we need v5 or greater
Invoke-Pester -Output Detailed
```

Installing the tools

Download the latest release and run it.

Starting Visual Studio Code

In your Start Menu, look for the *Pico - Visual Studio Code* shortcut, in the *Raspberry Pi Pico SDK <version>* folder. The shortcut sets up the needed environment variables and then launches Visual Studio Code.

Opening the examples

The first time you launch Visual Studio Code using the Start Menu shortcut, it will open the pico-examples repository.

To re-open the examples repository later, you can open the copy installed at `C:\Users\<user>\Documents\Pico-<version>\pico-examples`.

Building an example

Visual Studio Code will ask if you want to configure the pico-examples project when it is first opened; click *Yes* on that prompt to proceed. (If you miss the prompt look for the 'bell' icon in the bottom right.) You will then be prompted to select a kit -- select the *Pico ARM GCC - Pico SDK Toolchain with GCC arm-none-eabi* entry. If the *Pico ARM GCC* entry is not present, select *Unspecified* to have the SDK auto-detect the compiler.

To build one of the examples, click the *CMake* button on the sidebar. You should be presented with a tree view of the example projects; expand the project you'd like to build, and click the small build icon to the right of the target name to build that specific project.

To build everything instead, click the *Build All Projects* button at the top of the CMake Project Outline view.

Debugging an example

The `pico-examples` repository comes with `.vscode\*.json` files configured for debugging with Visual Studio Code. You can copy these files into your own projects as well.

To start debugging an example, click the *Run and Debug* button on the sidebar. The *Pico Debug* launch configuration should be selected already. To start debugging, click the small 'play' icon at the top of the debug window, or press F5.

The first time you start debugging, you will be prompted to select a target. If you wish to later change the launch target, you can do so using the status bar button with the name of the target.

Assuming that you have a Picoprobe configured and connected to your target device, your selected target should now be built, uploaded, and started. The debugger interface will load, and will pause the execution of the code at the `main()` entry point.
At this point, you can use the usual debugging tools to step, set breakpoints, inspect memory, and so on.

Wiring up SWD and UART to Picoprobe

Picoprobe wiring is explained in the Getting started document, *Appendix A: Using Picoprobe*, under the heading *Picoprobe Wiring*.

The Raspberry Pi Pico board used as Picoprobe should be flashed with the latest `picoprobe.uf2` build available at Picoprobe releases. The OpenOCD build included with the SDK installer only supports the CMSIS-DAP version of Picoprobe.

Open serial monitor in VSCode

The SDK installer adds the *Serial Monitor* extension to Visual Studio Code. Picoprobe includes a USB-serial bridge as well; assuming that you have wired up the TX and RX pins of the target to Picoprobe as described previously, you should have an option to select *COMn - USB Serial Device (COMn)* in the *Serial Monitor* tab in the bottom panel.

The baud rate should be set to the default of 115200 in most cases. Click *Start Monitoring* to open the serial port.

Command-line usage

To build and debug projects using command-line tools, you can open a terminal window using the *Pico - Developer Command Prompt* or *Pico - Developer PowerShell* shortcuts.

Start OpenOCD and gdb

```
openocd -f interface/cmsis-dap.cfg -f target/rp2040.cfg -c "adapter speed 5000"
```

If you wish to run gdb from the command line, you can invoke it like this:

```
arm-none-eabi-gdb
```

For example, to load and debug the `hello_serial` example, you might do:

(assuming that OpenOCD is already running as described above)

```
cd ${env:PICO_EXAMPLES_PATH}\build\hello_world\serial\
arm-none-eabi-gdb hello_serial.elf # hello_serial.elf is built at SDK install time by pico-setup.cmd
```

Then inside gdb:

```
(gdb) target remote localhost:3333
(gdb) load
(gdb) monitor reset init
(gdb) continue
```

Creating a new project

The commands below are for PowerShell, and will need to be adjusted slightly if you're using Command Prompt instead.

1. Copy pico_sdk_import.cmake from the SDK into your project directory:

   ```
   copy ${env:PICO_SDK_PATH}\external\pico_sdk_import.cmake .
   ```

2. Copy VS Code configuration from the SDK examples into your project directory:

   ```
   copy ${env:PICO_EXAMPLES_PATH}\.vscode . -recurse
   ```

3. Setup a `CMakeLists.txt` like:

4. ```
   cmake_minimum_required(VERSION 3.13)
   
   # initialize the SDK based on PICO_SDK_PATH
   # note: this must happen before project()
   include(pico_sdk_import.cmake)
   
   project(my_project)
   
   # initialize the Raspberry Pi Pico SDK
   pico_sdk_init()
   
   # rest of your project
   ```

16. Write your code (see pico-examples or the Raspberry Pi Pico C/C++ SDK documentation for more information)
   ```

About the simplest you can do is a single source file (e.g. hello_world.c)

```
#include <stdio.h>
#include "pico/stdlib.h"

int main() {
    setup_default_uart();
    printf("Hello, world!\n");
    return 0;
}
```

And add the following to your `CMakeLists.txt`:

```
add_executable(hello_world
    hello_world.c
)

# Add pico_stdlib library which aggregates commonly used features
target_link_libraries(hello_world pico_stdlib)

# create map/bin/hex/uf2 file in addition to ELF.
pico_add_extra_outputs(hello_world)
```

Note this example uses the default UART for *stdout*; if you want to use the default USB see the hello-usb example.

17. Launch VS Code from the *Pico - Visual Studio Code* shortcut in the Start Menu, and then open your new project folder.

18. Configure the project by running the *CMake: Configure* command from VS Code's command palette.

19. Build and debug the project as described in previous sections.

Uninstalling

Open the *Apps and Features* section in Windows Settings, then select *Raspberry Pi Pico SDK <version>*. Click the *Uninstall* button and follow the prompts.