Running Two Independent Geany Configurations on One PC

A Clean Way to Keep Unrelated Toolchains From Fighting Over One Build-Commands Menu

Development Guide Summary

August 19, 2026

Contents

1 The Problem

Geany has exactly one *Build* → *Set Build Commands* menu, and it is shared by every file Geany has ever opened. That is fine as long as you only ever compile one kind of thing. It stops being fine the moment you have two unrelated toolchains on the same machine – an SDCC/8051 setup for the STC89C52RC on one project, and a GNU RISC-V GCC setup for the CH32V003F4P6 on another – because both want to own the same Compile/Build/Execute commands, and switching between them means retyping paths and flags every time you change projects.

The fix is not to cram both toolchains into one Build Commands menu with manual switching. It is to run **two independent copies of Geany's configuration**, each with its own Build Commands, filetype settings, and recent-files list, launched from two separate shortcuts. Geany itself was built to support exactly this, via one command-line switch.

2 How Geany's Configuration Directory Works

Everything that makes a Geany install "yours" – Build Commands, plugin state, recent files, keybindings, custom filetype definitions – lives in one folder, not in the installed program files. On Windows that folder is:

```
C:\Users\<you>\AppData\Roaming\geany\
```

`geany.exe` itself, sitting in `Program Files`, is completely generic – every user-specific setting is read from and written to that one `AppData` folder. That separation is what makes multiple configurations possible: point Geany at a different folder, and it behaves like a completely separate installation, with no shared state at all.

3 The Mechanism: `-c` / `--config`

Geany's command-line interface documents exactly this switch:

> `-c` *dir_name*, `--config=`*dir_name* – Use an alternate configuration directory. The default configuration directory is `%APPDATA%\geany\` on Windows.

Point two different shortcuts at two different directories, and each launch of Geany is functionally its own installation – own Build Commands, own everything – while both still run off the single `geany.exe` you already have installed. Nothing about the program itself needs duplicating.

4 Step-by-Step Setup

4.1 1. Close every open Geany window

Geany writes its configuration back to disk on exit. Setting this up with a copy already running risks it overwriting the very files you are about to create.

4.2 2. Pick a folder name per toolchain

A clear, descriptive folder name pays for itself the first time you forget which shortcut is which. This guide uses:

```
C:\Users\<you>\AppData\Roaming\Geany-STC-SDCC
C:\Users\<you>\AppData\Roaming\Geany-CH32V003
```

You do not need to create these folders by hand – Geany creates a configuration directory automatically (populating it with default template and filetype files) the first time it is launched against a path that does not yet exist.

4.3 3. Create one shortcut per configuration

Right-click on the Desktop (or in a folder of your choosing) → *New* → *Shortcut*, and set the target to:

Listing 1: Shortcut target for the STC/SDCC profile

```
"C:\Program Files\Geany\bin\geany.exe" -c "C:\Users\<you>\
    AppData\Roaming\Geany-STC-SDCC"
```

Listing 2: Shortcut target for the CH32V003/RISC-V profile

```
"C:\Program Files\Geany\bin\geany.exe" -c "C:\Users\<you>\
    AppData\Roaming\Geany-CH32V003"
```

Adjust the install path if Geany lives somewhere other than the default `Program Files` location. Quote both paths in full – the executable path and the `-c` argument each need their own quotes if either contains a space (as `Program Files` and `AppData\Roaming` both do).

Rename each shortcut immediately to something unambiguous – "Geany (STC/SDCC)" and "Geany (CH32V003)" – rather than leaving both as "Geany – Shortcut". A different icon color per shortcut (right-click → Properties → Change Icon) is a cheap extra safeguard against opening the wrong one on autopilot.

4.4 4. Launch each once and configure its Build Commands

Double-click the first shortcut. Geany starts with a fresh, default configuration – no Build Commands set yet for anything project-specific. Go to *Build → Set Build Commands* and enter the toolchain's commands, exactly as already documented on this site:

Table 1: Build Commands per profile (see each project's own getting-started guide for full detail)

Profile	**Compile command**	**Build command**
Geany (STC/SDCC)	`sdcc -c "%f"`	`sdcc "%f"`
Geany (CH32V003)	*(single-file compile is less useful here – see below)*	`cmd /c build_ch32v003.bat`

The CH32V003 project is multi-file (main.c, debug.c, the peripheral drivers, the startup file), so its "Build" command reasonably just re-runs the whole batch file / Makefile rather than compiling one file in isolation – unlike the STC89C52RC's single-file SDCC workflow. Both are legitimate patterns; Geany does not care which one a given profile uses.

Repeat for the second shortcut. Because each profile's configuration lives in its own folder, setting Build Commands in one window has no effect whatsoever on the other – there is nothing to accidentally overwrite.

4.5 5. Verify the isolation

With both shortcuts configured, a quick sanity check is worth the thirty seconds:

1. Open **Geany (STC/SDCC)**, open any `.c` file, confirm *Build → Set Build Commands* shows the SDCC commands.
2. Close it. Open **Geany (CH32V003)**, confirm its Build Commands show the RISC-V batch file, *not* the SDCC commands.
3. Check *File → Recent Files* in each – the two lists should be completely independent, since each profile logs its own history.

If either check fails, the most common cause is a typo in the `-c` path on one of the two shortcuts, causing both to resolve to the same (usually the default) configuration directory.

5 A Note on the Windows Taskbar

Because both shortcuts launch the identical `geany.exe`, Windows may group their taskbar icons together despite the shortcuts being named differently

– Windows' taskbar grouping keys off the executable's identity, not the shortcut used to launch it, and `geany.exe` does not set a distinct Application User Model ID per configuration. This is cosmetic only; both windows still run fully independently. If it is worth solving, the two common fixes are pinning each shortcut separately (rather than pinning a running instance) and relying on the differently-colored icons from Step 3 to tell them apart at a glance, or wrapping each shortcut in its own `.bat` launcher, which Windows will sometimes group separately from a plain `.exe` shortcut.

6 Why Not Just Use Command-Line Flags Each Time?

Geany does not offer a way to select Build Commands from the command line – they are read from the configuration directory at startup, not passed as arguments per invocation. A single shared configuration therefore forces a choice: either re-enter Build Commands by hand every time you switch projects (error-prone, and exactly the kind of thing that goes stale and quietly breaks a build), or maintain two configurations that never need touching once set up. The `-c` switch is Geany's own supported mechanism for the latter, not a workaround.

7 Extending This to a Third Toolchain

Nothing about this approach is limited to two profiles. A third project – say, an ARM Cortex-M target using `arm-none-eabi-gcc` – gets its own folder (`Geany-ARM`, or similar) and its own shortcut, with zero interaction with the two configurations already in place. The pattern scales to as many independent toolchains as a given machine ends up needing.