

[PRODUCTS](#) [DOWNLOADS](#) [TUTORIALS](#) [SUPPORT](#) [CONTACT](#)[Tutorials](#) > [Embedded](#) > [ESP8266/ESP32](#) > Switching Advanced ESP-IDF Projects Between Different IDF Versions

Switching Advanced ESP-IDF Projects Between Different IDF Versions

📅 March 26, 2018 🔖 esp-idf, esp32

This tutorial shows how to install several independent ESP-IDF checkouts into your ESP32 toolchain and switch the advanced ESP-IDF projects between those IDF versions. We will create a basic project based on the ESP-IDF 3.0 release and show how to switch it to the experimental master branch of ESP-IDF from github.

Before you begin, install VisualGDB 5.4 or later.



Categories

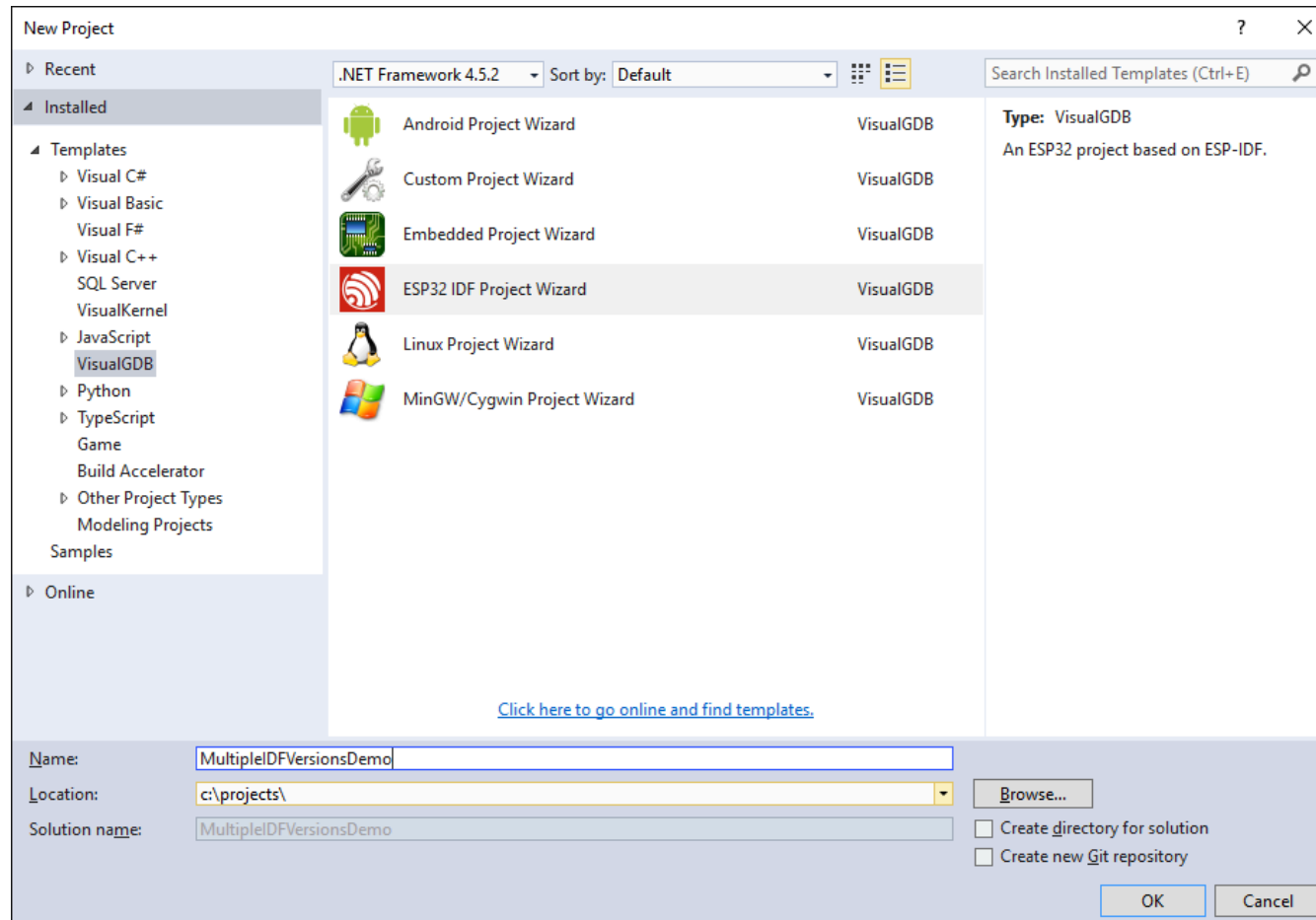
Documentation

[Advanced](#)[Embedded Projects](#)[General](#)[IntelliSense](#)[Linux](#)[Profiler](#)[Project Types](#)[Tests](#)[Troubleshooting](#)

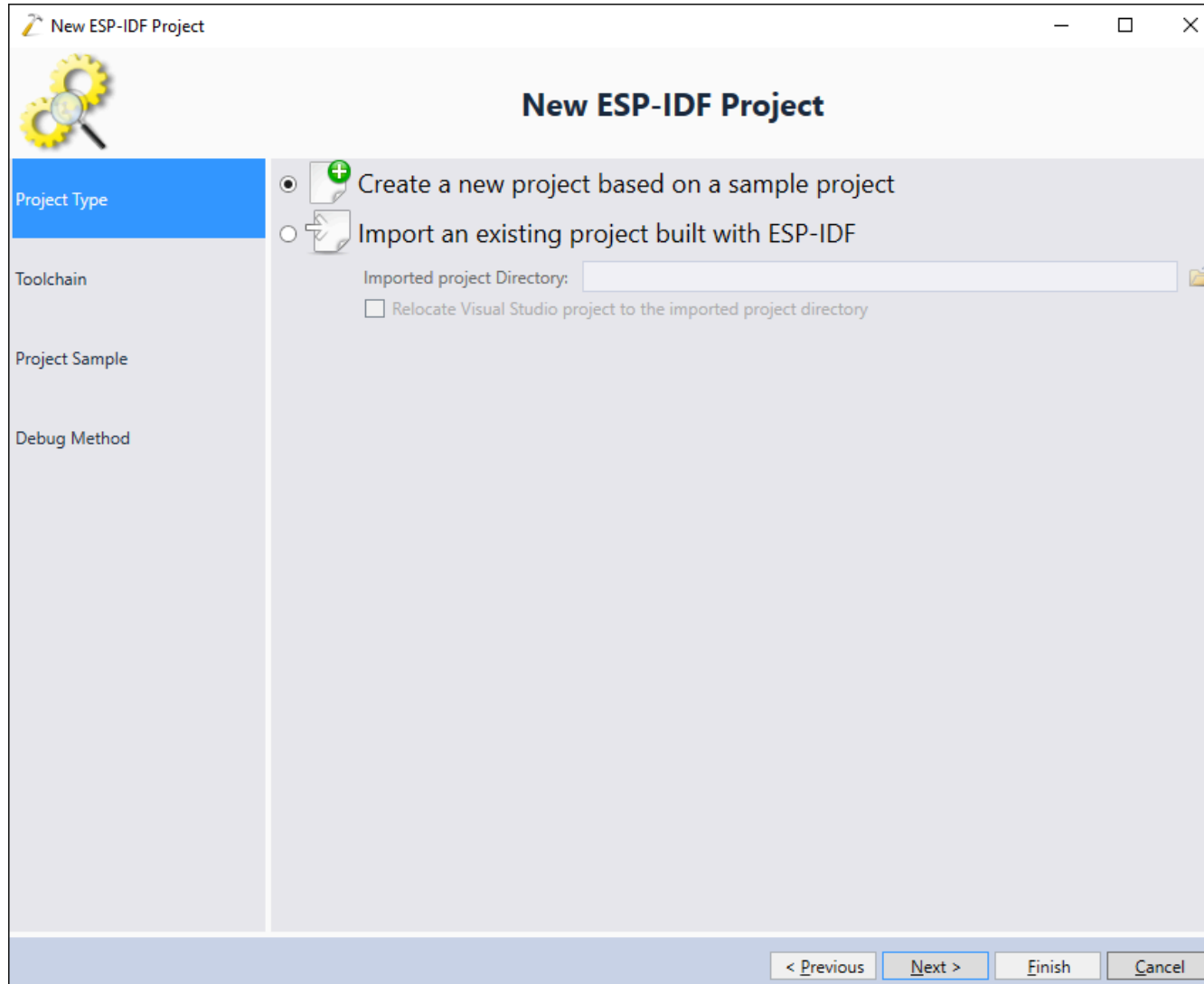
Tutorials

[Android](#)[Cocos2d-x](#)[Integration with other tools](#)[Archive](#)[Android](#)

1. Start Visual Studio and open the Advanced ESP-IDF Project Wizard:

[Embedded](#)[Linux](#)[Arduino](#)[CMake](#)[Continuous Integration](#)[Customization](#)[FreeBSD](#)[SDK](#)[Embedded](#)[ARM Features](#)[CMake](#)[ESP8266/ESP32](#)[Getting Started with Boards](#)[Internet of Things](#)[mbed](#)[MSP430](#)[RTOS](#)[STM32 Boards & Tools](#)[STM32 Peripherals](#)[IntelliSense](#)[Linux](#)[Beaglebone](#)[Cubieboard](#)[Linux Frameworks & Tools](#)

2. On the first page of the wizard select “Create a new project” and click “Next”:



3. On the next page select your ESP32 toolchain. VisualGDB will display the list of ESP-IDF checkouts installed into that toolchain. Select the default checkout and click “Next”:

Raspberry Pi

Live Tracing

Porting

Profiler

Embedded

Linux

Real-Time Watch

Unit Tests

Windows

Cygwin

MinGW

Uncategorised

Tags

android android samples arduino

arm beaglebone bluetooth cmake

CodeExplorer cross-compile custom

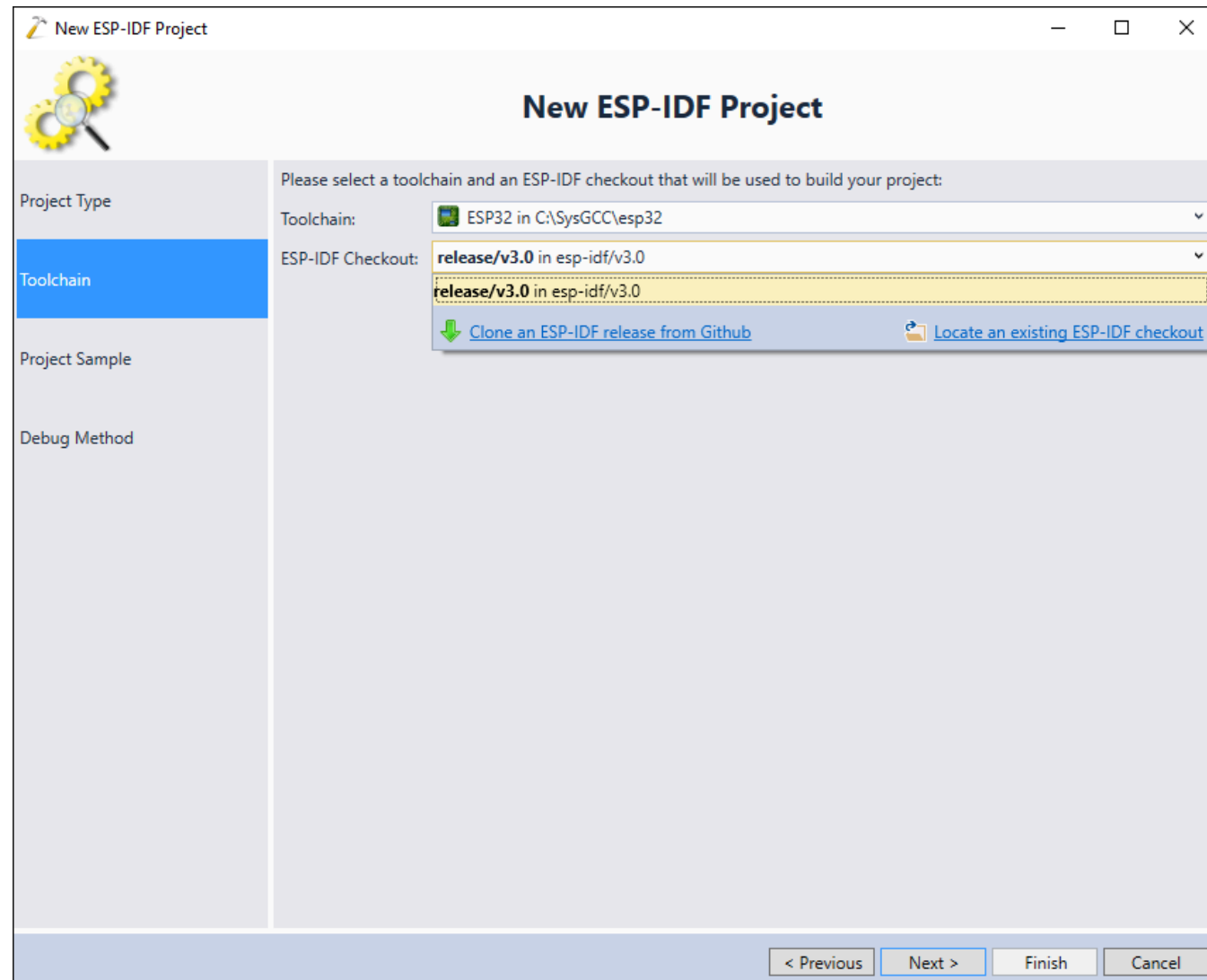
embedded embedded cmake esp32

esp8266 freertos HTTP import

intellisense IoT keil kinetis lcd led library

linux live tracing mbed msbuild non-

intrusive debugging nrf51 nxp openocd

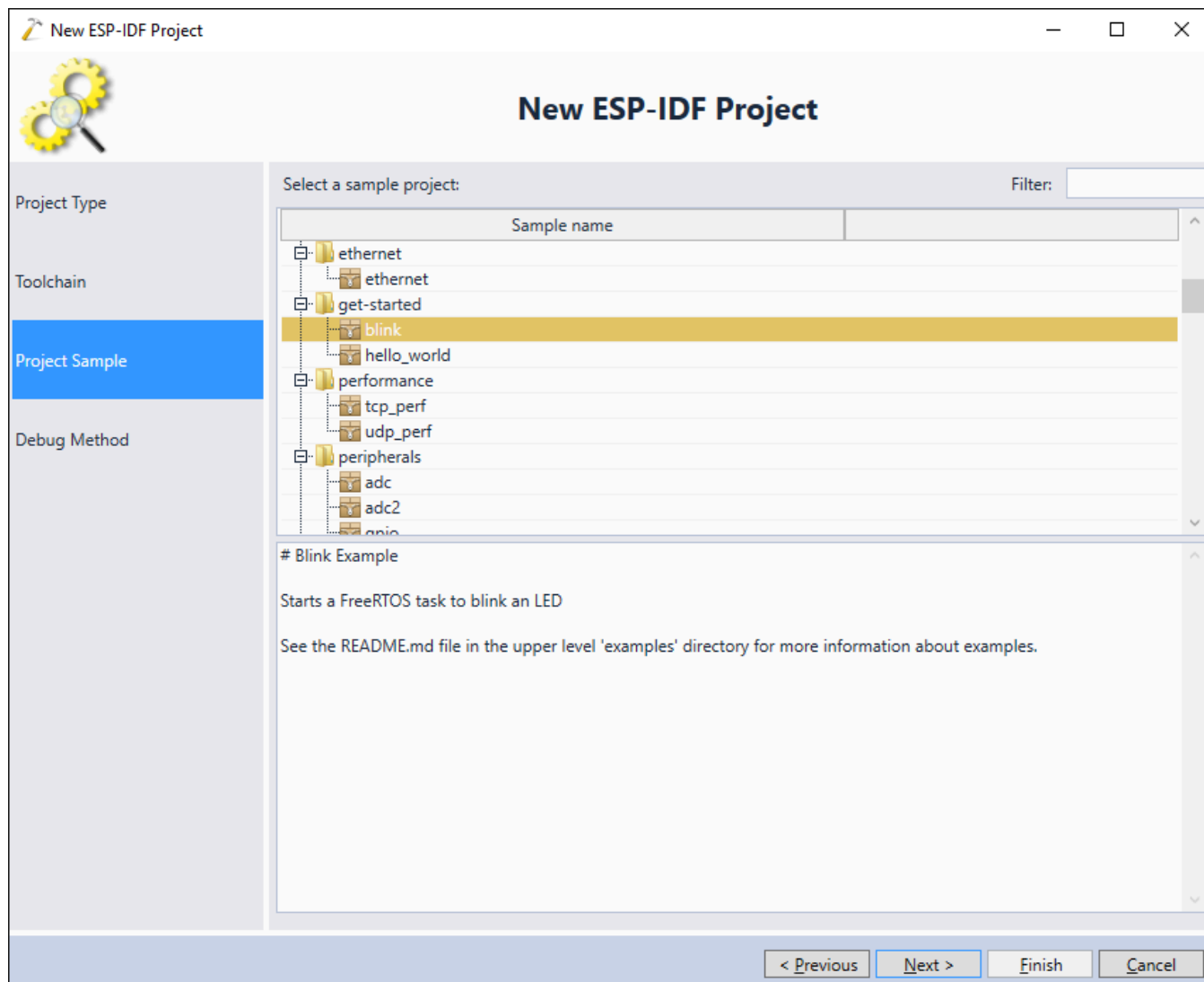


[porting profiler qt quickstart raspberry](#)
[raspberry pi rp2040 stm32 tests](#)
[troubleshooting uart WiFi win32](#)

4. As different ESP-IDF versions may not be 100% compatible with each other, we will use the most basic “blink” project to demonstrate switching between IDF versions. Select it on the



“Project Sample” page and click “Next”:



5. Finally select the debugging settings that match your project and click “Finish” to create it.

New ESP-IDF Project

Project Type

Toolchain

Project Sample

Debug Method

Debug using: Olimex ARM-USB-OCD-H Test

☒ Set JTAG/SWD frequency to: 3000 KHz

Program FLASH memory: Always Never If rebuilt since last load

Program FLASH using: OpenOCD (via JTAG) esptool.py (via a COM port)

Use the ESP-IDF Settings page to specify the COM port for FLASH programming.

FLASH settings will be automatically imported from ESP-IDF project settings.

[Show a tutorial on troubleshooting ESP32 FLASH programming](#)

☒ Show FreeRTOS threads in the 'threads' window

☐ Additional FLASH resources to program

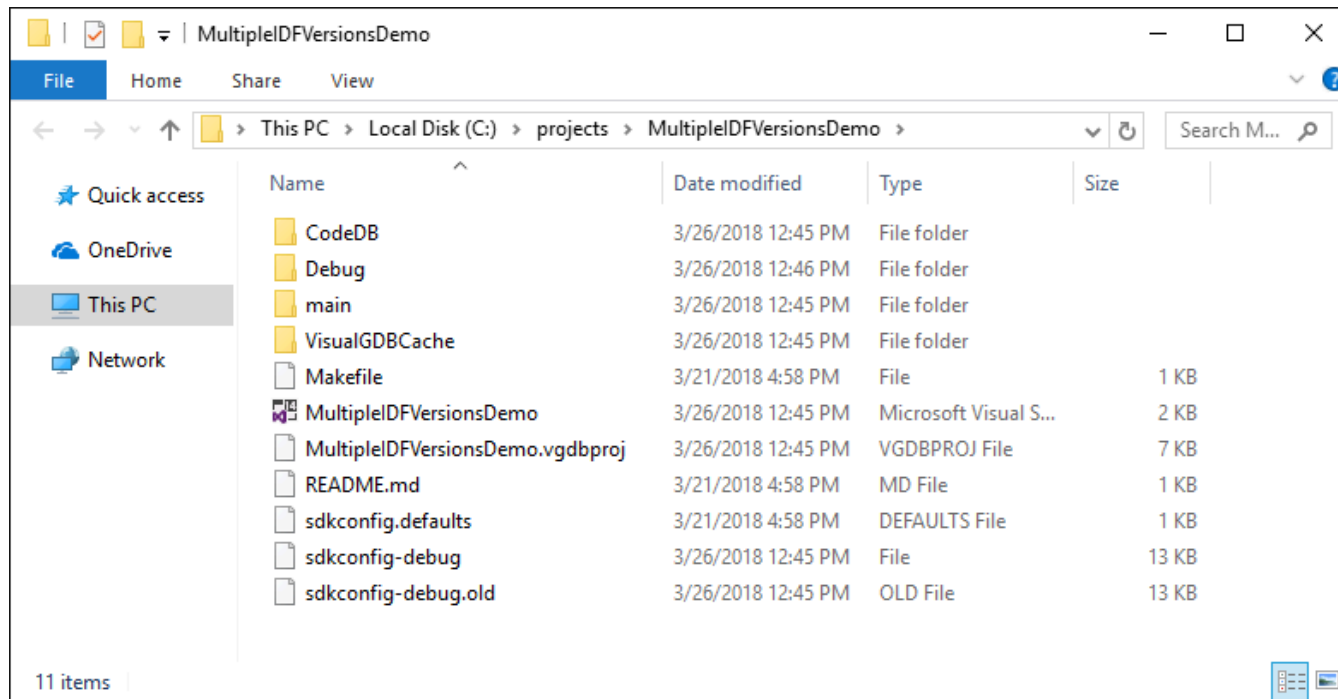
☐ Advanced settings

< Previous Next > Finish Cancel

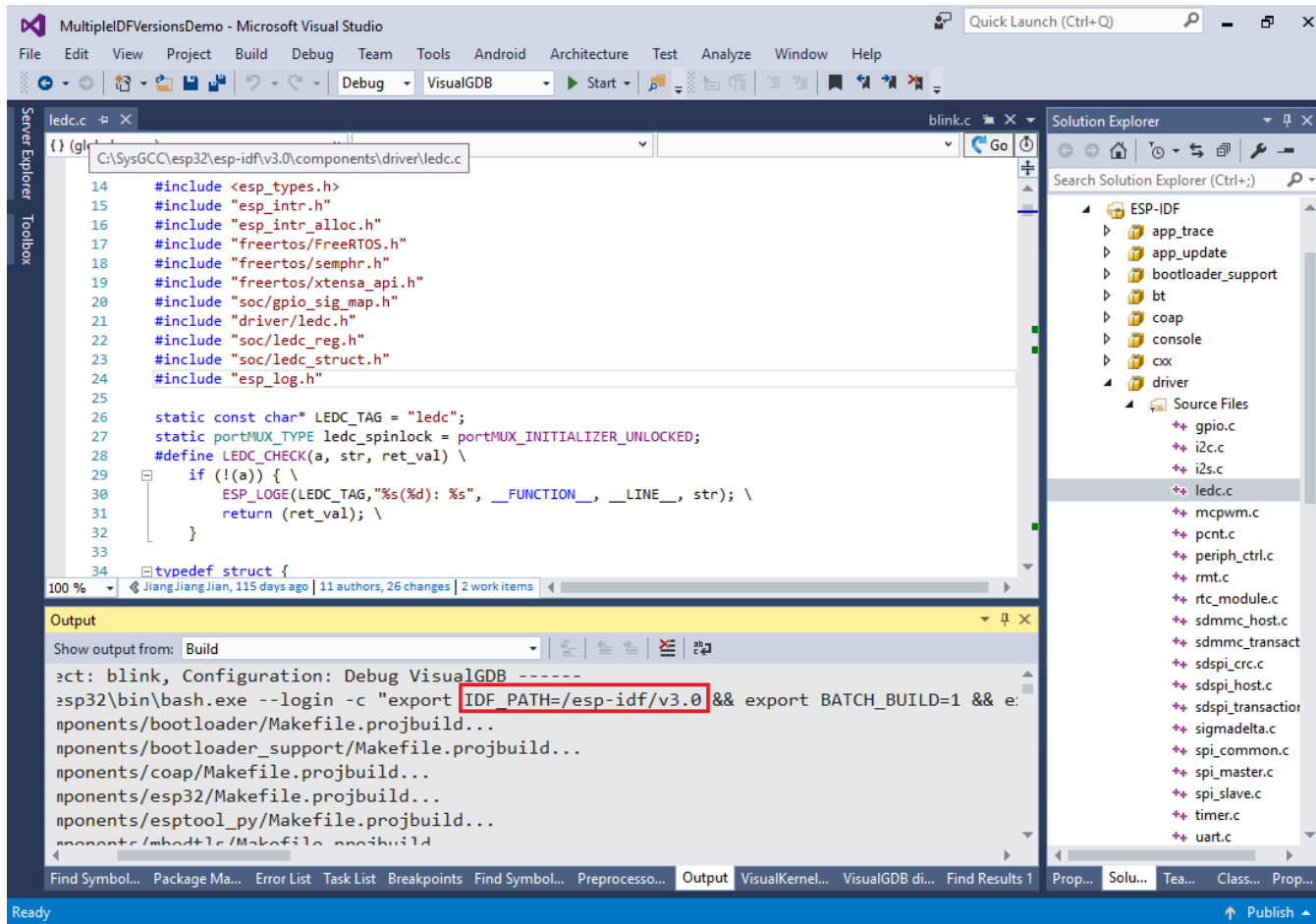
6. VisualGDB will create a basic project. Note that the project itself doesn't include a copy of ESP-IDF and only includes the following files:

File	Description
Makefile	Project-level Make file defining project name and other options.

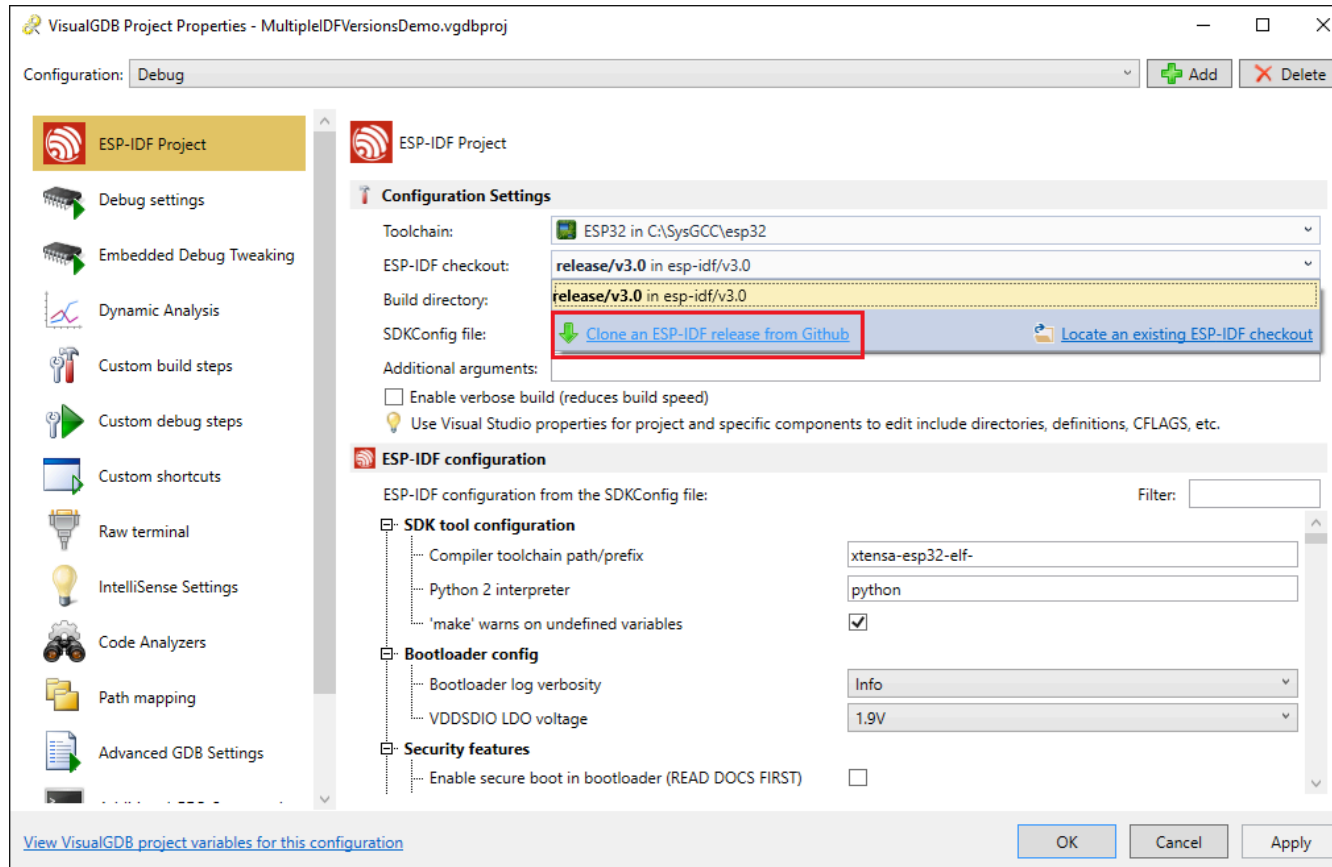
<project>.vgdbproj	VisualGDB project file containing debugging settings, basic build settings and the list of configurations.
sdkconfig-<configuration>	Per-configuration sdkconfig files. They will be created first time you open the corresponding configuration of the project.
main\component.mk	Configuration file for the main project component.



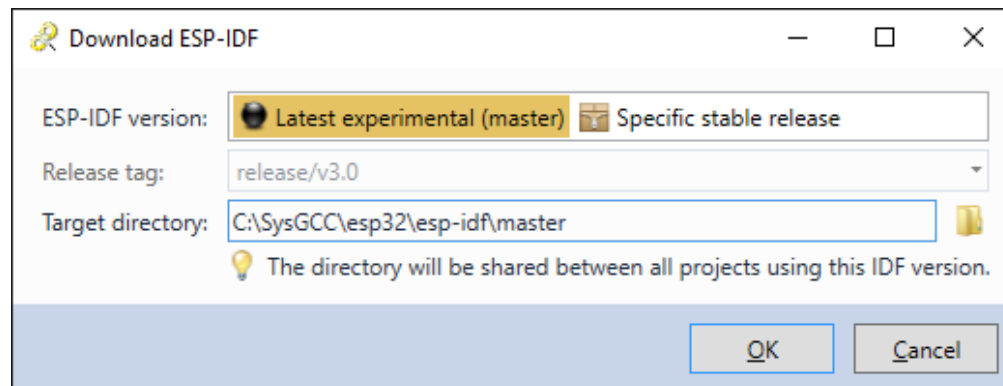
7. Note that the ESP-IDF path is not stored in any of the project-specific Makefiles. Instead VisualGDB will locate and substitute it dynamically when you open or build the project:



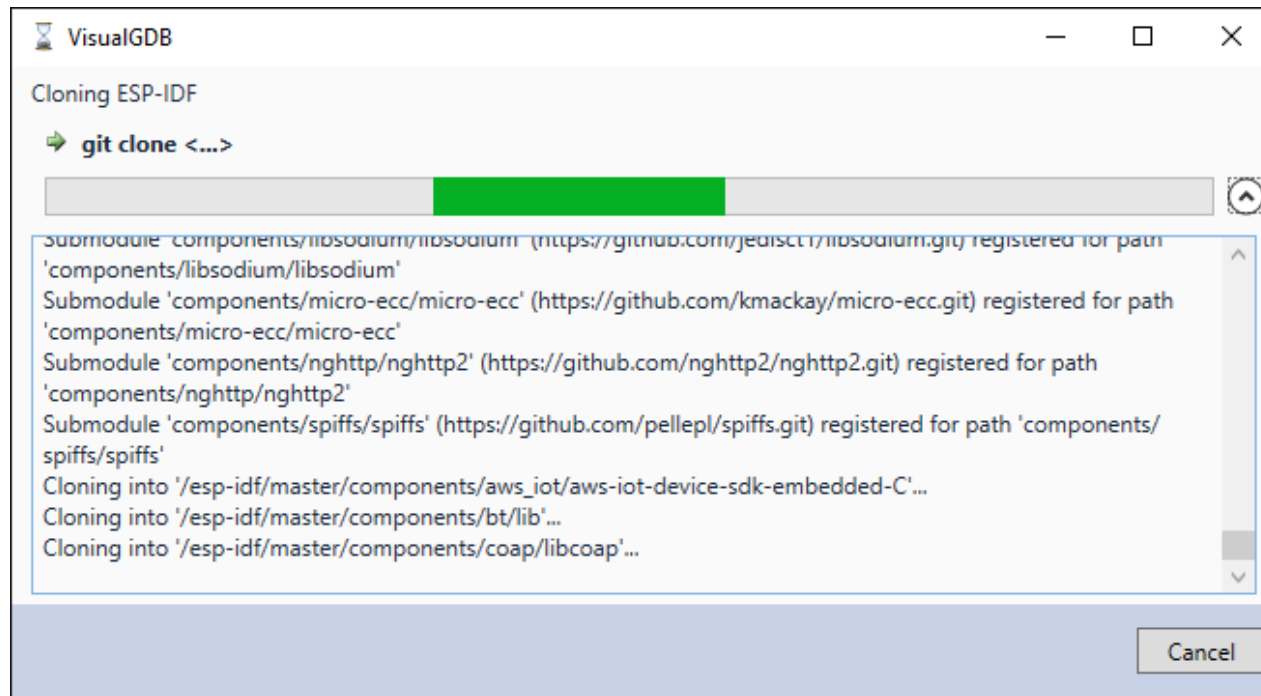
8. Now we will show how to change the ESP-IDF version. Open VisualGDB Project Properties on the first page, select "Build Directory" -> "Clone an ESP-IDF release":



9. Pick the version you want to clone (VisualGDB will automatically load the list of release tags from the github repository) and click OK:

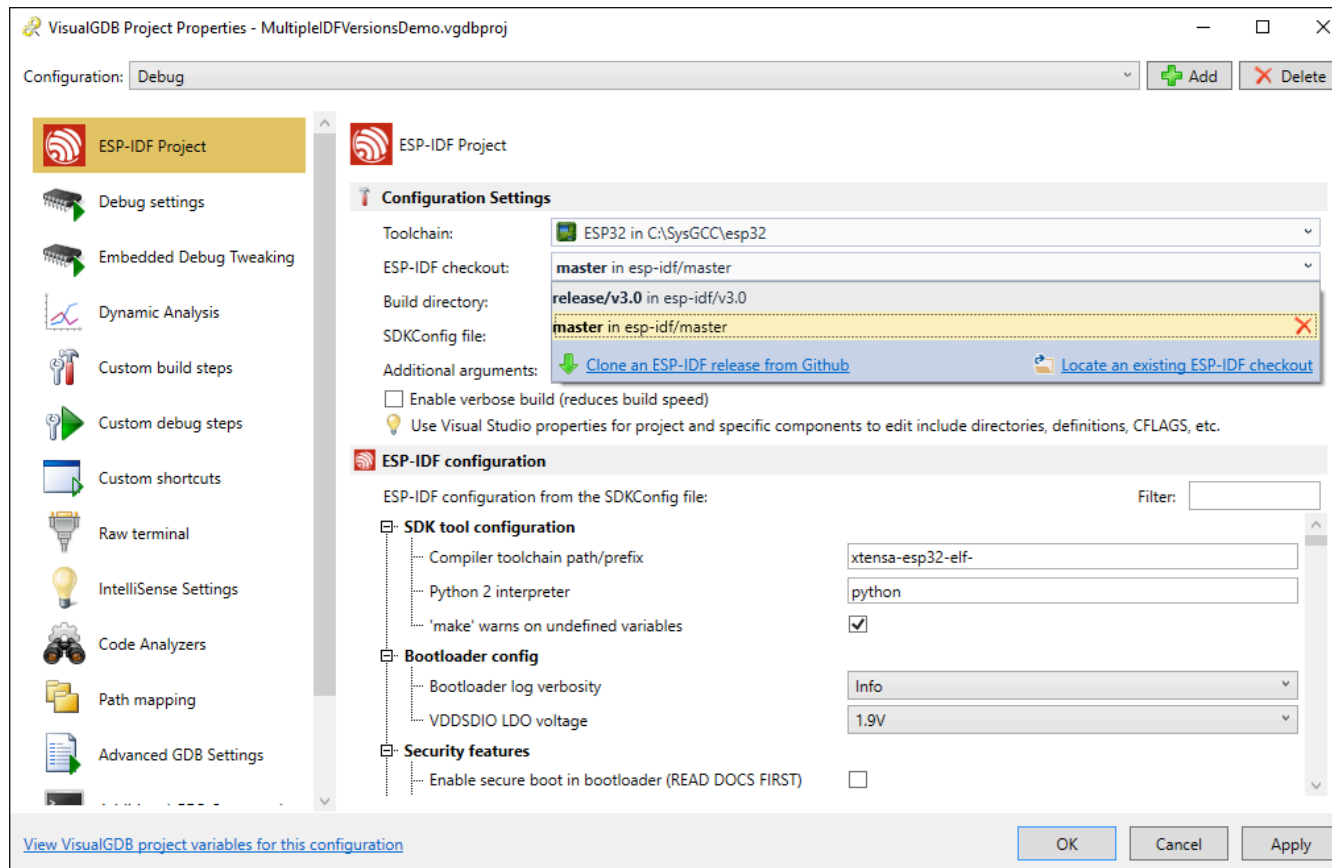


10. VisualGDB will automatically launch git and create a recursive checkout of ESP-IDF and all related libraries:

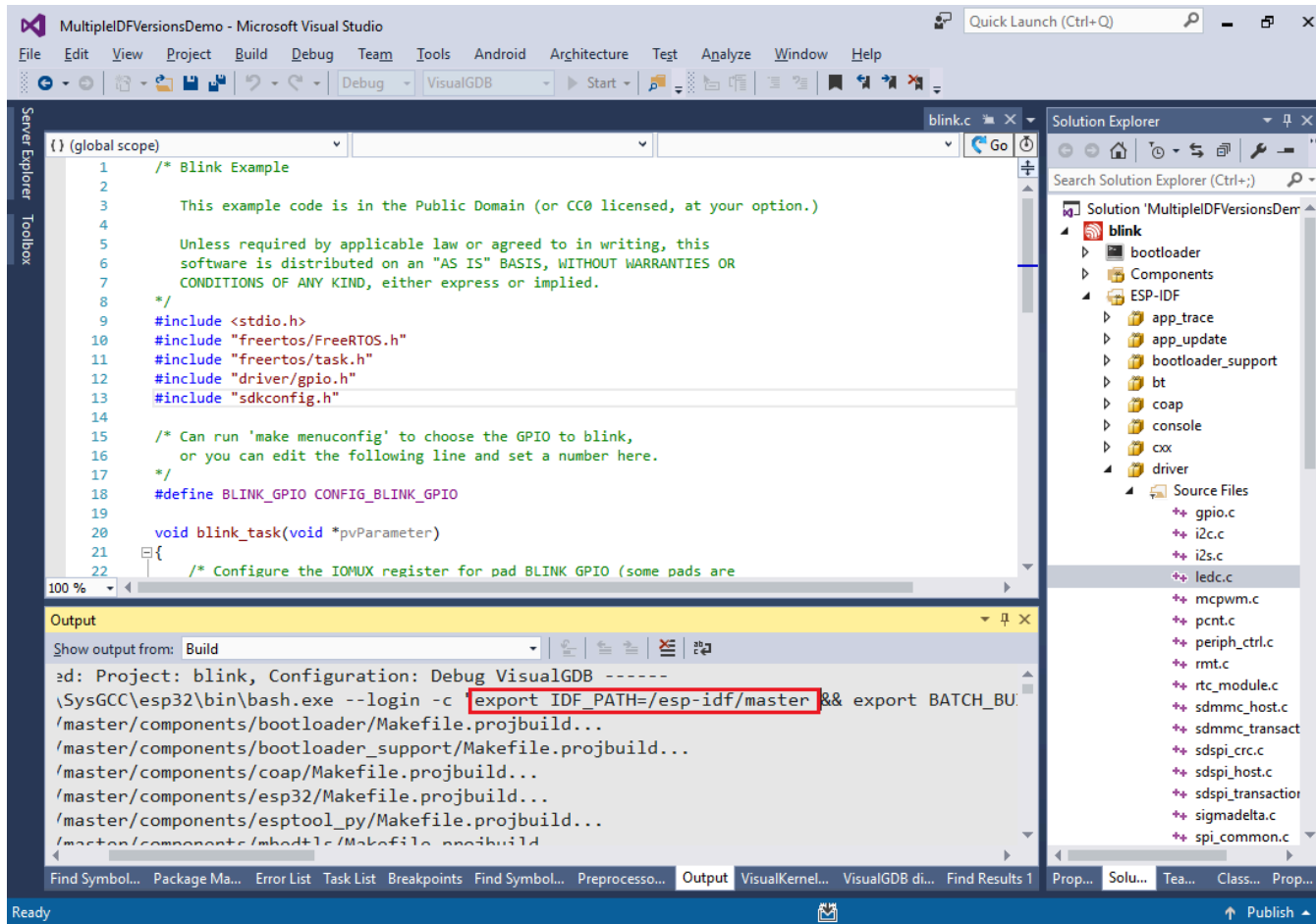


11. Once the checkout is complete, you will be able to select it in VisualGDB Project Properties or in the ESP-IDF project wizard:

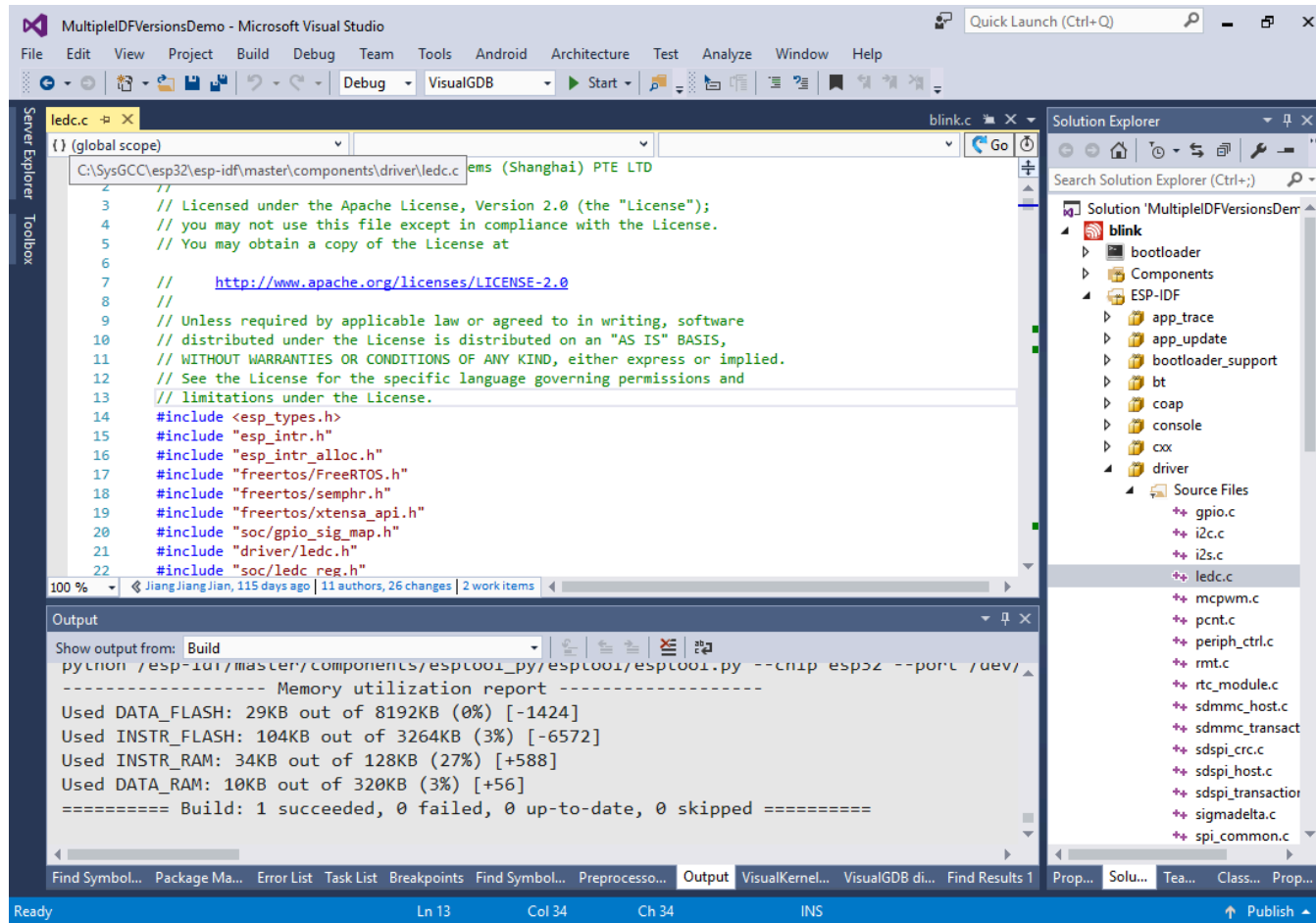




12. Build the project again. Note how VisualGDB now sets IDF_PATH to match the new checkout location:



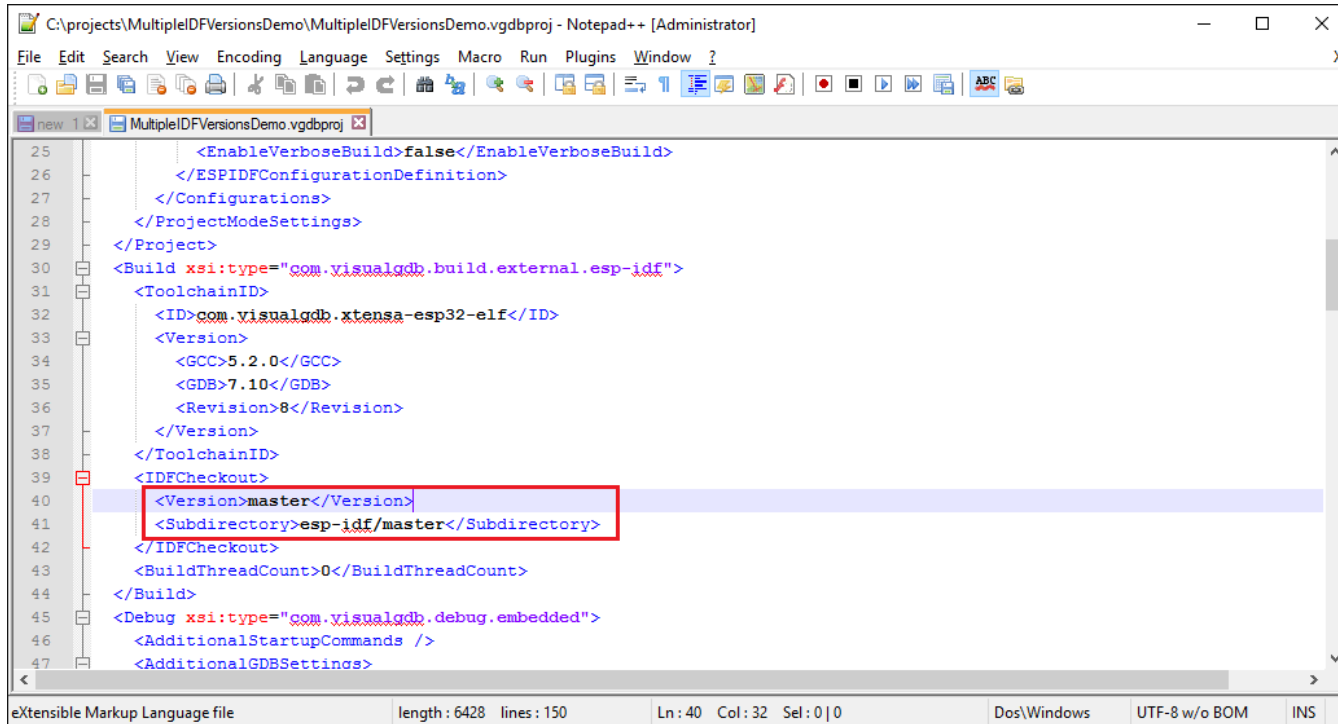
13. Once VisualGDB updates the code model, Solution Explorer will switch to showing files from the new ESP-IDF checkout as well:



You can always switch the checkout back and forth via VisualGDB Project Properties. As long as it is present in your toolchain directory, you don't need to re-clone it each time.

14. VisualGDB doesn't hardcode the absolute path of the checkout in the project properties file. Instead it stores the git branch name and the subdirectory in the toolchain folder (in case

multiple copies of the same branch are checked out to different locations):



```
<?xml version="1.0" encoding="UTF-8" ?>
<ESPIDFConfigurationDefinition>
  <EnableVerboseBuild>false</EnableVerboseBuild>
</ESPIDFConfigurationDefinition>
</Configurations>
</ProjectModeSettings>
</Project>
<Build xsi:type="com.visualgdb.build.external.esp-idf">
  <ToolchainID>
    <ID>com.visualgdb.xtensa-esp32-elf</ID>
    <Version>
      <GCC>5.2.0</GCC>
      <GDB>7.10</GDB>
      <Revision>8</Revision>
    </Version>
  </ToolchainID>
  <IDFCheckout>
    <Version>master</Version>
    <Subdirectory>esp-idf/master</Subdirectory>
  </IDFCheckout>
  <BuildThreadCount>0</BuildThreadCount>
</Build>
<Debug xsi:type="com.visualgdb.debug.embedded">
  <AdditionalStartupCommands />
  <AdditionalGDBSettings>

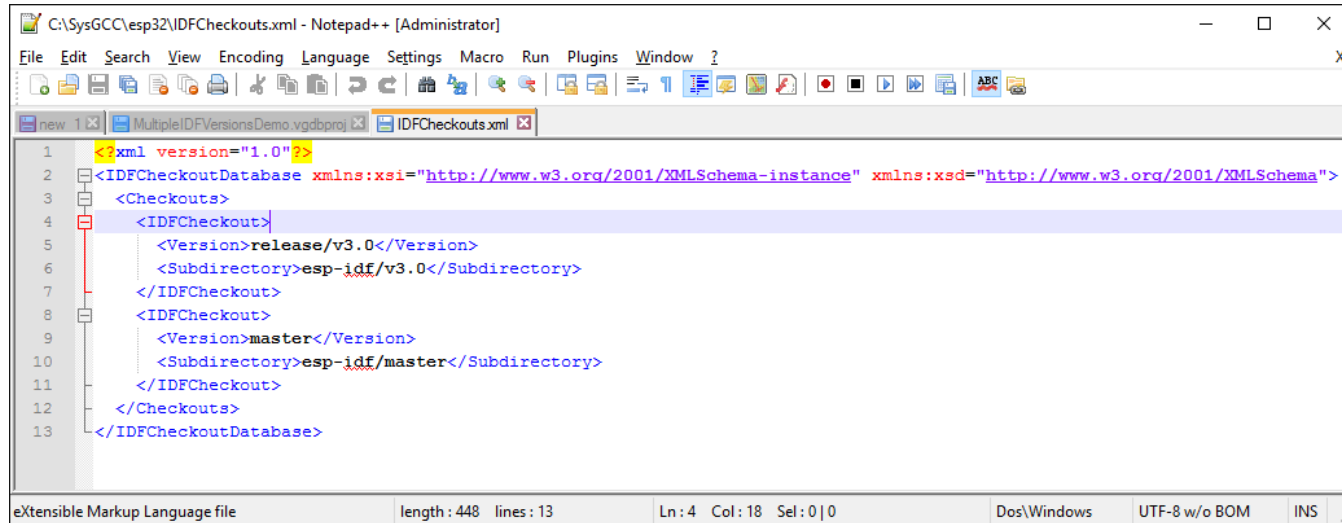
```

If you open the same project on another machine, VisualGDB will first try locating a checkout by both version and subdirectory, and if none is found, will fall back to using version only.

15. The list of ESP-IDF checkouts is stored in the <toolchain directory>\IDFCheckouts.xml file. If you decide to work on a private fork of ESP-IDF, ensure you distribute the same checkout



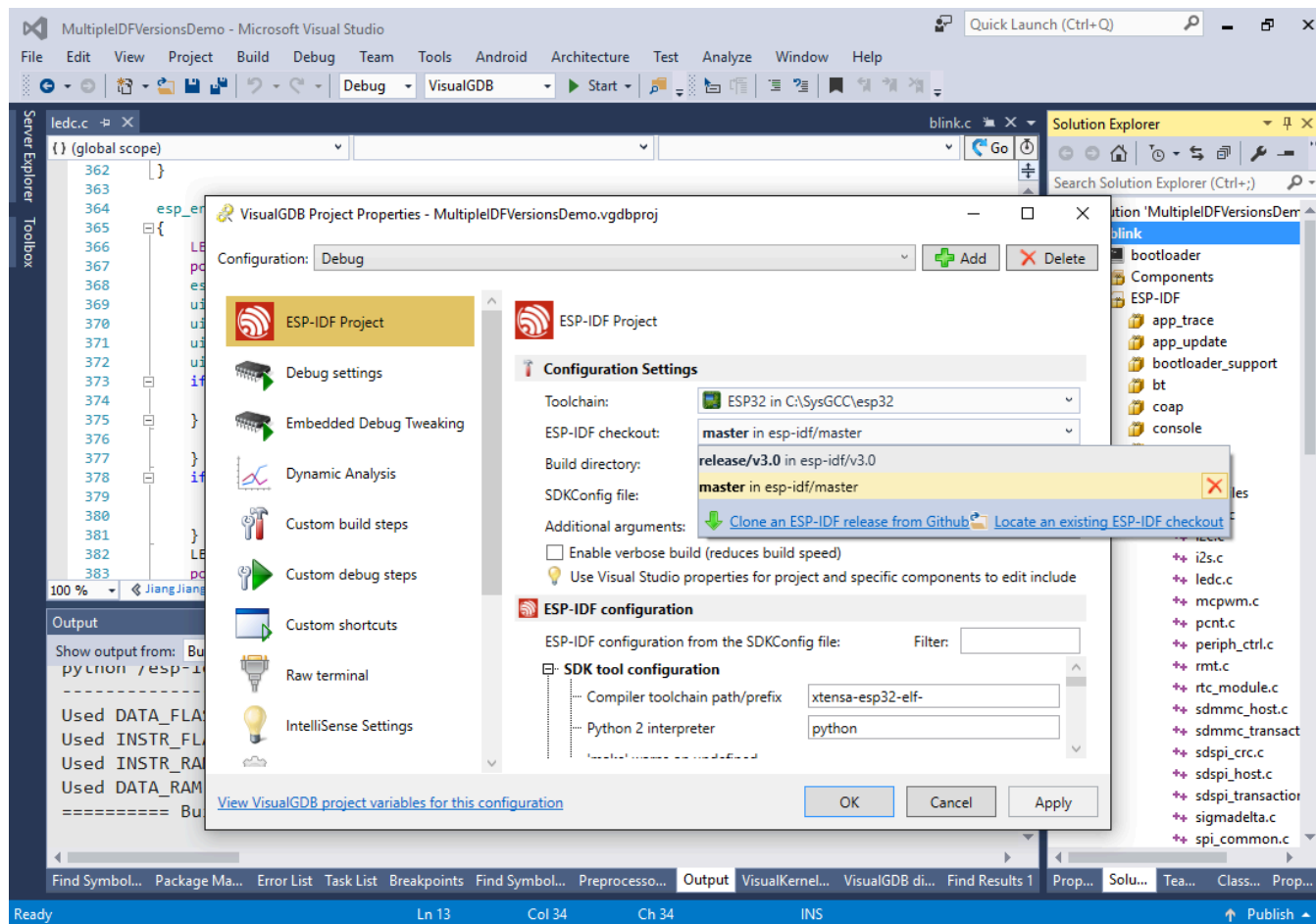
(and the corresponding IDFCheckouts.xml file) to all your development machines:



```
1  <?xml version="1.0"?>
2  <IDFCheckoutDatabase xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema">
3    <Checkouts>
4      <IDFCheckout>
5        <Version>release/v3.0</Version>
6        <Subdirectory>esp-idf/v3.0</Subdirectory>
7      </IDFCheckout>
8      <IDFCheckout>
9        <Version>master</Version>
10       <Subdirectory>esp-idf/master</Subdirectory>
11     </IDFCheckout>
12   </Checkouts>
13 </IDFCheckoutDatabase>
```

16. You can delete the checkouts you don't need anymore directly from the checkout selector in VisualGDB Project Properties and the ESP-IDF project wizard:





© 2012-2022 Sysprogs OÜ. All rights reserved.

[Terms of Use](#) | [Copyright](#) | [Privacy Policy](#)