Visual Studio | Marketplace Sign in

Visual Studio Code > Programming Languages > ESP-IDF New to Visual Studio Code? Get i

ESP-IDF

Espressif Systems espressif.com | 621,676 installs | (104) | Free

Develop and debug applications for Espressif ESP32, ESP32-S2 chips with ESP-IDF

Install Trouble Installing?

Overview Version History Q & A Rating & Review

ESP-IDF Visual Studio Code Extension

Tutorials Documentation Troubleshooting Supported Chips version v1.7.1 Github Releases Forum esp32.com

Develop, build, flash, monitor, debug and more with Espressif chips using Espressif IoT Development Framework (ESP-IDF)

Nightly builds for Visual Studio Code. You can use this VSIX to test the current github master of the extension by pressing F1 or click menu `View` -> `Command Palette...`, type `Install from VSIX` and then select the previously downloaded `.vsix` file to install the extension.

Make sure to review our documentation first to properly use the extension.

Tutorials

1. Install and setup the extension.
2. Create a project from ESP-IDF examples, Build, flash and monitor.
3. Debugging with steps to configure OpenOCD and debug adapter.
4. Heap tracing
5. Code coverage
6. Developing on Docker Container
7. Developing on WSL

Check all the tutorials here.

Table of content

Check all the documentation.

How to use

- Download and install Visual Studio Code.

Categories

Programming Languages Snippets Debuggers

Tags

arduino-esp32 bluetooth debuggers ESP ESP3 esp32c2 ESP32-C2 esp32c3 ESP32-C3 esp32c esp32h2 ESP32-H2 esp32p4 ESP32-P4 esp32s ESP32-S2 esp32s3 ESP32-S3 ESP-ADF ESP-IDF ESP-matter ESP-MDF Espressif iot kconfig keybindings matter snippet soc wifi

Works with

Universal

Resources

Issues
Repository
Homepage
License
Changelog
Download Extension

Project Details

espressif/vscode-esp-idf-extension
Last Commit: 3 days ago
8 Pull Requests
17 Open Issues

More Info

Version	1.7.1
Released on	31/12/2019 06:54:02
Last updated	22/03/2024 06:47:31
Publisher	Espressif Systems
Unique Identifier	espressif.esp-idf-extension
Report	Report a concern

- Then
  - Either open Visual Studio Code and create a workspace folder.
  - Run `code ${YOUR_PROJECT_DIR}` from a command line terminal.
- Install this extension in your Visual Studio Code.

There are few dependencies required in your system and available in environment variable PATH before installing this extension. Please review the following documentation.

- Requirements for Linux
- Requirements for MacOS
- For Windows the C++ Build Tools might be required.

Install ESP-IDF and ESP-IDF Tools by following the install tutorial.

> **NOTE:** Please note that this extension **only supports** the release versions of ESP-IDF, you can use the extension on `master` branch or other branch, but certain features might not properly work.

> **NOTE:** If you are using Windows Subsystem for Linux (WSL) 2, please take a look at the WSL tutorial for an step by step instruction or check the requirements in WSL Documentation needed in the WSL distribution.

- (OPTIONAL) Press F1 and type **ESP-IDF: Select where to Save Configuration Settings**, which can be User settings, Workspace settings or workspace folder settings.

  > **NOTE:** Please take a look at Working with multiple projects for more information. Default is User settings.

- On the first time using the extension, press F1 to show the Visual Studio Code Command Palette and type **ESP-IDF: Configure ESP-IDF Extension** to open the extension configuration wizard. This will install ESP-IDF, ESP-IDF tools, create a virtual python environment with ESP-IDF and this extension python packages and configure the extension settings with these values. **NOTE: Make sure that there is no spaces in any configured path since ESP-IDF build system doesn't support spaces yet.**.

  > **NOTE:** Please take a look at Install tutorial documentation or the Setup documentation for details about extension setup and configuration.

- Press F1 and type **ESP-IDF: Show Examples Projects** to create a new project from ESP-IDF examples.
- Configure the `.vscode/c_cpp_properties.json` as explained in C/C++ Configuration. There is a default configuration that should work when you create a new project with the extension commands but you might want to modify it to your needs.

  > **Note:** For code navigation the Microsoft C/C++ Extension or Clangd extension can be used to resolve header/source links. By default, projects created with **ESP-IDF: Create Project from Extension Template** or **ESP-IDF: Show Examples Projects** include a template for Microsoft C/C++ extension `c_cpp_properties.json` configuration file. This can be optimized by building the project first and configure your project to use `build/compile_commands.json` as explained in C/C++ Configuration.

- Do some coding!
- Check you set the correct port of your device by pressing F1, typing **ESP-IDF: Select Port to Use:** and choosing the serial port your device is connected.
- Select an Espressif target (esp32, esp32s2, etc.) with the **ESP-IDF: Set Espressif Device Target** command.
- Use the **ESP-IDF: Select OpenOCD Board Configuration** to choose the openOCD configuration files for the extension openOCD server.
- When you are ready, build your project by pressing F1 and typing **ESP-IDF: Build your Project**.
- Flash to your device by pressing F1 and typing **ESP-IDF: Select Flash Method and Flash** to select either UART or JTAG. You can also use the **ESP-IDF: Flash (UART) your Project** or **ESP-IDF: Flash (with JTag)** directly.

  > **NOTE:** When using the **ESP-IDF: Select Flash Method and Flash** command, your choice will be saved in the `idf.flashType` configuration setting in the current workspace folder's settings.json.

- You can later start a monitor by pressing F1 and typing **ESP-IDF: Monitor your Device** which will log the device activity in a Visual Studio Code terminal.
- To make sure you can debug your device, select the your board by pressing F1 and typing **ESP-IDF: Select OpenOCD Board Configuration** or manually define the openOCD configuration files with `idf.openOcdConfigs` configuration in your settings.json.

- If you want to start a debug session, just press F5 (make sure you had at least build and flash once before so the debugger works correctly). Check the Troubleshooting section if you have any issues.

Available commands

Click F1 to show Visual studio code actions, then type **ESP-IDF** to see possible actions.

Command Description	Keyboard Shortcuts (Mac)	Keyboard Shortcuts (Windows/ Linux)
Add Arduino ESP32 as ESP-IDF Component		
Add Docker Container Configuration		
Add Editor coverage		
Add OpenOCD rules file (For Linux users)		
Add vscode configuration folder		
Build, Flash and start a monitor on your device	⌘ I D	Ctrl E D
Build your project	⌘ I B	Ctrl E B
Clear eFuse Summary		
Clear ESP-IDF Search Results		
Clear Saved ESP-IDF Setups		
Configure ESP-IDF extension		
Configure Paths		
Configure Project SDKConfig for Coverage		
Create project from Extension Template	⌘ I C	Ctrl E C
Create New ESP-IDF Component		
Device configuration		
Dispose Current SDK Configuration Editor Server Process		
Doctor Command		
Encrypt and Flash your Project		
Erase Flash Memory from Device	⌘ I R	Ctrl E R
Execute Custom Task	⌘ I J	Ctrl E J
Flash your project	⌘ I F	Ctrl E F
Flash (DFU) your project		
Flash (UART) your project		
Flash (with JTag)		
Full Clean Project	⌘ I X	Ctrl E X
Get eFuse Summary		
Get HTML Coverage Report for project		
Import ESP-IDF Project		
Install ESP-ADF		
Install ESP-IDF Python Packages		
Install ESP-MDF		
Install ESP-Matter		
Install ESP-Rainmaker		

Install ESP-HomeKit-SDK		
Launch IDF Monitor for CoreDump / GDB-Stub Mode		
Launch QEMU Server		
Launch QEMU Debug Session		
Monitor your device	⌘ I M	Ctrl E M
Monitor QEMU Device		
New Project	⌘ I N	Ctrl E N
Open ESP-IDF Terminal	⌘ I T	Ctrl E T
Open NVS Partition Editor		
Pick a workspace folder		
Remove Editor coverage		
Save Default SDKCONFIG file (save-defconfig)		
SDK Configuration editor	⌘ I G	Ctrl E G
Search in documentation...	⌘ I Q	Ctrl E Q
Select Flash Method		
Select port to use	⌘ I P	Ctrl E P
Select OpenOCD Board Configuration		
Select where to save configuration settings		
Select output and notification mode		
Set default sdkconfig file in project		
Set Espressif device target		
Set ESP-MATTER Device Path (ESP_MATTER_DEVICE_PATH)		
Show Examples Projects		
Show Ninja Build Summary		
Size analysis of the binaries	⌘ I S	Ctrl E S
Unit Test: Build and flash unit test app for testing		
Unit Test: Install ESP-IDF PyTest requirements		
Remove Editor coverage		
Run ESP-IDF-SBOM vulnerability check		

About commands

1. The **Add Arduino-ESP32 as ESP-IDF Component** command will add Arduino-ESP32 as a ESP-IDF component in your current directory (`${CURRENT_DIRECTORY}/components/arduino`).

   > **NOTE:** Not all versions of ESP-IDF are supported. Make sure to check Arduino-ESP32 to see if your ESP-IDF version is compatible.

2. You can also use the **ESP-IDF: Create Project from Extension Template** command with `arduino-as-component` template to create a new project directory that includes Arduino-ESP32 as an ESP-IDF component.

3. The **Install ESP-ADF** will clone ESP-ADF inside the selected directory and set `idf.espAdfPath` (`idf.espAdfPathWin` in Windows) configuration setting.

4. The **Install ESP-Matter** will clone ESP-Matter inside the selected directory and set `idf.espMatterPath` configuration setting. The **ESP-IDF: Set ESP-MATTER Device Path (ESP_MATTER_DEVICE_PATH)** is used to define the device path for ESP-Matter. **ESP-Matter is not supported in Windows**.
5. The **Install ESP-MDF** will clone ESP-MDF inside the selected directory and set `idf.espMdfPath` (`idf.espMdfPathWin` in Windows) configuration setting.
6. The **Install ESP-HomeKit-SDK** will clone ESP-HomeKit-SDK inside the selected directory and set `idf.espHomeKitSdkPath` (`idf.espHomeKitSdkPathWin` in Windows) configuration setting.
7. The **Show Examples Projects** command allows you create a new project using one of the examples in ESP-IDF, ESP-ADF, ESP-Matter, ESP-HomeKit-SDK or ESP-MDF directory if related configuration settings are correctly defined.

Commands for tasks.json and launch.json

We have implemented some utilities commands that can be used in tasks.json and launch.json that can be used like:

```
"miDebuggerPath": "${command:espIdf.getXtensaGdb}"
```

- `espIdf.getExtensionPath`: Get the installed location absolute path.
- `espIdf.getOpenOcdScriptValue`: Return the value of OPENOCD_SCRIPTS from `idf.customExtraVars` or from system OPENOCD_SCRIPTS environment variable.
- `espIdf.getOpenOcdConfig`: Return the openOCD configuration files as string. Example `-f interface/ftdi/esp32_devkitj_v1.cfg -f board/esp32-wrover.cfg`.
- `espIdf.getProjectName`: Return the project name from current workspace folder `build/project_description.json`.
- `espIdf.getXtensaGcc`: Return the absolute path of the toolchain gcc for the ESP-IDF target given by `idf.adapterTargetName` configuration setting and `idf.customExtraPaths`.
- `espIdf.getXtensaGdb`: Return the absolute path of the toolchain gdb for the ESP-IDF target given by `idf.adapterTargetName` configuration setting and `idf.customExtraPaths`.

See an example in the debugging documentation.

Available Tasks in tasks.json

A template Tasks.json is included when creating a project using **ESP-IDF: Create Project from Extension Template**. These tasks can be executed by running F1, writing `Tasks: Run task` and selecting one of the following:

1. `Build` - Build Project
2. `Set Target to esp32`
3. `Set Target to esp32s2`
4. `Clean` - Clean the project
5. `Flash` - Flash the device
6. `Monitor` - Start a monitor terminal
7. `OpenOCD` - Start the openOCD server
8. `BuildFlash` - Execute a build followed by a flash command.

Note that for OpenOCD tasks you need to define `OPENOCD_SCRIPTS` in your system environment variables with openocd scripts folder path.

Troubleshooting

If something is not working please check for any error on one of these:

> **NOTE:** Use `idf.openOcdDebugLevel` configuration setting to 3 or more to show debug logging in OpenOCD server output.

> **NOTE:** Use `logLevel` in your /.vscode/launch.json to 3 or more to show more debug adapter output.

1. In Visual Studio Code select menu **View** -> **Output** -> **ESP-IDF**. This output information is useful to know what is happening in the extension.
2. In Visual Studio Code select menu **View** then click **Command Palette...** and type `ESP-IDF: Doctor Command` to generate a report of your environment configuration and it will be copied in your clipboard to

paste anywhere.

3. Check log file which can be obtained from:

- Windows: `%USERPROFILE%\.vscode\extensions\espressif.esp-idf-extension-VERSION\esp_idf_vsc_ext.log`
- Linux & MacOSX: `$HOME/.vscode/extensions/espressif.esp-idf-extension-VERSION/esp_idf_vsc_ext.log`

4. In Visual Studio Code, select menu **Help** -> `Toggle Developer Tools` and copy any error in the Console tab related to this extension.

5. Make sure that your extension is properly configured as described in JSON Manual Configuration. Visual Studio Code allows the user to configure settings at different levels: **Global (User Settings)**, **Workspace** and **Workspace Folder** so make sure your project has the right settings. The `ESP-IDF: Doctor command` result might give the values from user settings instead of the workspace folder settings.

6. Review the OpenOCD troubleshooting FAQ related to the `OpenOCD` output, for application tracing, debug or any OpenOCD related issues.

If there is any Python package error, please try to reinstall the required python packages with the **ESP-IDF: Install ESP-IDF Python Packages** command. Please consider that this extension install ESP-IDF, this extension's and ESP-IDF Debug Adapter python packages when running the **ESP-IDF: Configure ESP-IDF Extension** setup wizard.

If the user can't resolve the error, please search in the github repository issues for existing errors or open a new issue here.

Code of Conduct

This project and everyone participating in it is governed by the Code of Conduct. By participating, you are expected to uphold this code. Please report unacceptable behavior to vscode@espressif.com.

License

This extension is licensed under the Apache License 2.0. Please see the LICENSE file for additional copyright notices and terms.