

Q "site:aithub.com docker vscode raspberrypi pico" を検索しました	10 結果
Q "Raspberry Pi Pico SDK setup in Docker with VSCode" を検索しました	10 結果
Q "DockerでVSCodeとPico SDK環境を構築する方法" を検索しました	10 結果

DockerでVSCodeとPico SDK環境を構築する方法

Raspberry Pi Picoの開発環境をDockerコンテナで構築し、VSCodeのDev Containers拡張機能を使って管理する方法が推奨されます。これにより、ホストOSの汚染を避け、クロスプラットフォームで一貫した環境を実現できます。主にC/C++開発を対象とし、Pico SDKを活用します。公式のRaspberry Pi Pico拡張機能も併用可能です。

前提条件

- **VSCode:** 最新版をインストール。
- **Docker:** Docker Desktop (Windows/macOS) またはDocker Engine (Linux) をインストール・起動。
- **VSCode拡張機能:**
 - Remote - Containers (必須: Dev Containersとしてインストール)。
 - Raspberry Pi Pico (公式: プロジェクト作成・ビルドを簡素化)。
 - C/C++ Extension Pack (IntelliSense・デバッグ用)。
 - CMake Tools (ビルド管理用)。

これらをVSCodeのExtensionsタブからインストールしてください。

推奨リポジトリの選択

GitHubに複数のDockerベースのDev Containerテンプレートがあります。初心者向けにシンプルでドキュメントが充実したものを選ぶと良いです。以下のおすすめ:

- **dodancs/pico-sdk-dev-container**: 基本的なC/C++環境。PDFドキュメントあり。
github.com
- **lukstep/raspberry-pi-pico-docker-sdk**: RP2040/RP2350対応。Docker Hubイメージあり。
github.com
- **AppacYazilim/PicoDevEnv**: 簡易セットアップ。C++特化。
github.com

ここでは**dodancs/pico-sdk-dev-container**を例に手順を説明します。他のリポジトリも似た流れです。

構築手順

1. リポジトリをクローン:

- ターミナルで以下を実行（任意のディレクトリに）：

text

×

≡

⊕

```
git clone https://github.com/dodancs/pico-sdk-dev-container.git  
my-pico-project  
cd my-pico-project
```

- これでDockerfile、devcontainer.json、Pico SDKの設定が揃います。Pico SDKはコンテナ内で自動クローンされます。

2. VSCodeでフォルダを開く:

- VSCodeで my-pico-project フォルダを開く（File > Open Folder）。
- 初回はコンテナがビルドされるので、数分かかります。コマンドパレット（Ctrl+Shift+P）で Dev Containers: Reopen in Container を選択。
 - これでDockerコンテナが起動し、VSCodeのワークスペースがコンテナ内に移行します。ステータスバーに（Dev Container: Pico-SDK Dev Container）と表示されます。

3. プロジェクト初期化:

- ・コンテナ内のターミナル (Ctrl+`) でPicoプロジェクトを作成:

```
text
```

×

≡

○

```
mkdir hello_world && cd hello_world
```

- ・CMakeLists.txtを作成（最小例）：

```
text
```

×

≡

○

```
cmake_minimum_required(VERSION 3.13)
include($ENV{PICO_SDK_PATH}/external/pico_sdk_import.cmake)
project(hello_world)
pico_sdk_init()
add_executable(hello_world hello_world.c) # または .cpp
target_link_libraries(hello_world pico_stl)
pico_add_extra_outputs(hello_world)
```

- ・サンプルソース hello_world.c を作成:

```
text
```

×

≡

○

```
#include <stdio.h>
#include "pico/stl.h"
int main() {
    const uint LED_PIN = PICO_DEFAULT_LED_PIN;
    gpio_init(LED_PIN);
    gpio_set_dir(LED_PIN, GPIO_OUT);
    while (true) {
        gpio_put(LED_PIN, 1);
        sleep_ms(250);
        gpio_put(LED_PIN, 0);
        sleep_ms(250);
    }
}
```

4. ビルドと実行:

- CMake Tools拡張で CMake: Configure を実行（コマンドパレット）。
- ビルド: CMake: Build またはターミナルで:

```
text
```

×

≡

○

```
mkdir build && cd build  
cmake ..  
make
```

- 生成された `hello_world.uf2` をPicoボードにドラッグ&ドロップ（BOOTSELモードでUSB接続）。

5. デバッグ設定（オプション）：

- PicoProbe（別Picoでデバッグ用）を使う場合、`devcontainer.json`の `runArgs` に USBデバイスを追加:

```
text
```

×

≡

○

```
"runArgs": ["--device=/dev/bus/usb", "--device=/dev/ttyACM0"]
```

- VSCodeの `.vscode/launch.json` でCortex-Debugを設定（拡張が自動提案）。
- コマンドパレットで Debug: Start Debugging を実行。

トラブルシューティング

問題

コンテナビルト失敗 Docker権限不足: sudo usermod -aG docker \$USER (Linux)。
再起動後再試行。 github.com

SDKクローンエラー GitHubトークン設定: VSCode設定で raspberry-pi-pico.githubToken にPersonal Access Tokenを入力。
github.com

USBデバイス未検出 devcontainer.jsonの runArgs に --privileged や --device=/dev/ttyUSB0 を追加。ホストで lsusb 確認。
winters.nz

OpenOCDエラー RP2040限定: 最新SDK使用。RP2350はOpenOCD未サポートの場合あり。 github.com

拡張機能インストール失敗 コンテナ再構築: Dev Containers: Rebuild Container。
github.com

追加Tips

- 公式拡張併用: インストール後、Raspberry Pi Pico: New Project でテンプレート生成。Docker内で動作。 github.com
- カスタムDockerfile: 必要に応じてDockerfileを編集 (例: GCC-ARM追加)。
- クロスプラットフォーム: Windows/macOS/Linuxで同一環境。WSL2 (Windows) 推奨。 niwat0r1.hateblo.jp
- 詳細ドキュメント: 各GitHubリポジトリのREADME参照。例: [dodancs](#)リポ。

この方法で、Pico SDKのクロスコンパイルからデバッグまでDocker内で完結します。問題があれば具体的なエラーを教えてください！

↳ PicoProbeデバッグの詳細手順

↳ Arduino IDEでのPico開発

↳ トラブルシューティングを詳細化