# VSCodeを利用した仮想化リアルタイムOS実演デモ

Athrill on TOPPERS SSP

# はじめに

V850の仮想環境athrill 上にてリアルタイムOS TOPPERS SSP シュリンク版のデモを行います。
ハンズオンセミナーの場合は、あらかじめ各自PCに環境をセットアップしておいてください。
内容的には20分程度のデモになります。

# おさらい

Athrill とは
PC上で動作するV850シミュレータになります。ターゲット(実機)がなくともPCだけで実行させることが可能であり、主に車載ソフトの試験に活用するために開発されたものです。
DockerHubにathrill本体（実行形式),サンプルソフトソースおよび実行形式、ビルド環境(V850ツールチェーン)が公開されており、非常に簡単に動作確認できる状態になっています。(2020年8月13日現在)

TOPPERS-SSP シュリンク版とは
TOPPERSの個人会員の髙橋が、TOPPERS-SSPの正規版から改造をしたもの。
C言語記述でのディスパッチャは新規に追加されたもので、C言語記述なので移植が容易になっている。
基本的には、タイマー割込みのベアメタルプログラムまでそれぞれのマイコンのプログラムから比較的容易に移植可能。コンフィグレータは対応していないためコンフィグのソースの手修正が必要になる。本デモでは実行可能なソースをGitHubに公開しているので、それを動作させる。

Visual Studio Code (Vscode)
Microsoftが無償で公開しているプログラミング環境(IDE)。今回Vscodeのリモート機能を使って、上記AthrillのDockerに接続、ビルドと実行を行います。

# 前提条件　(利用環境の条件)

1)Windows10pro 64bit バージョン 2004、ビルド 19041 以上
2)仮想化支援機能「Intel VT／AMD-V」対応プロセッサ
3)4GBメモリ以上
4)WSL2有効化済み
5)Docker For Windows インストール済み

おそらく、Dockerさえ動作すれば、上記環境以外でも可能とは思いますが、検証していないので、今回確認した環境を記載しています。

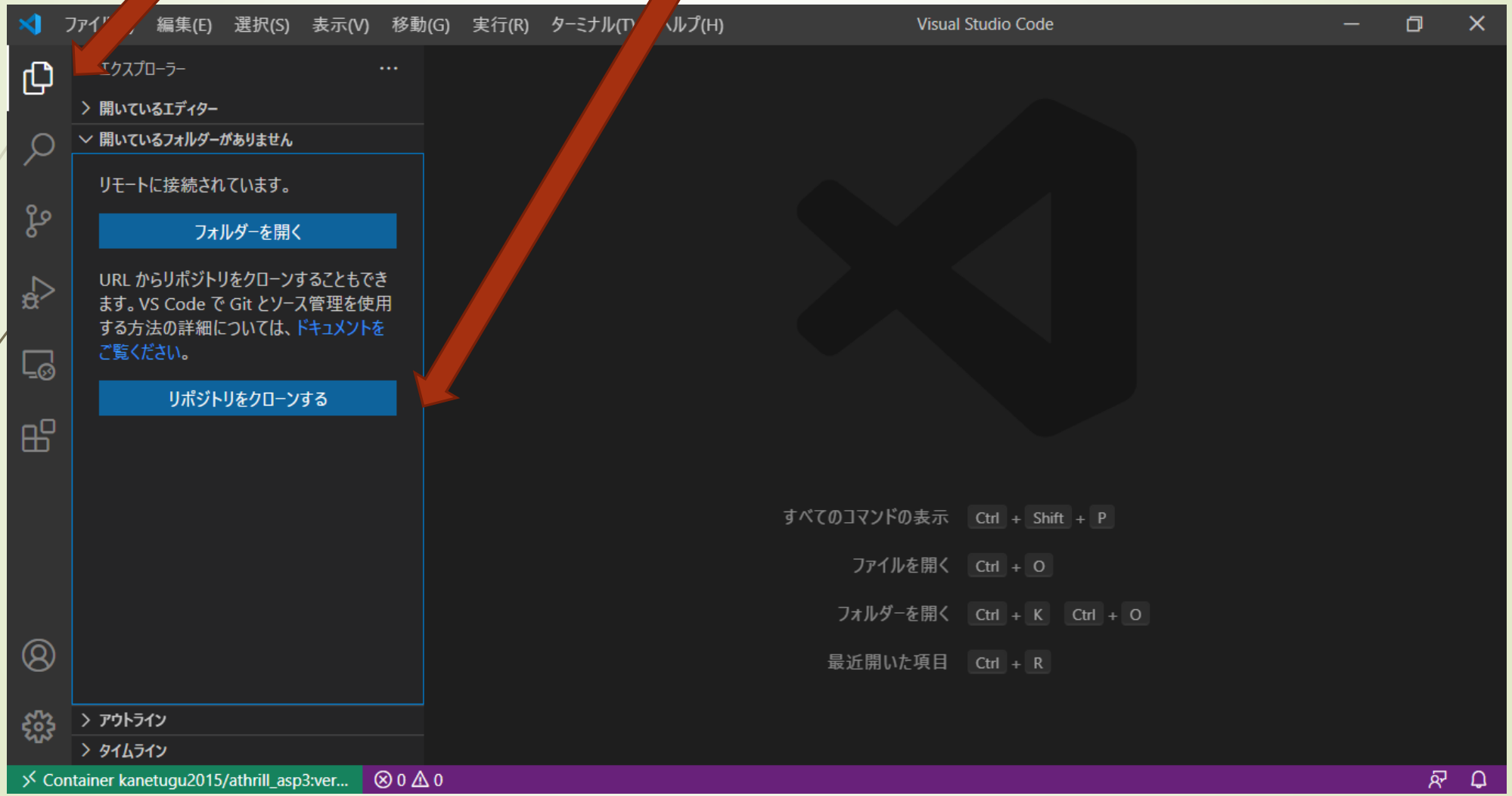
尚、事前準備として環境を設定する場合は以下の手順を案内します。
別途　「Athrill 実行環境事前設定の説明.docx」を参照ください。

# VSCodeのインストール

VScodeをネットからダウンロードしてインストールします。
https://code.visualstudio.com/
さらにリモート機能の拡張機能をインストールします。
Remote Development
ms-vscode-remote.vscode-remote-extensionpack
というのがパックになっているのでこれをインストールします。
さらにgitをインストールします。
Git Blame
GitLens

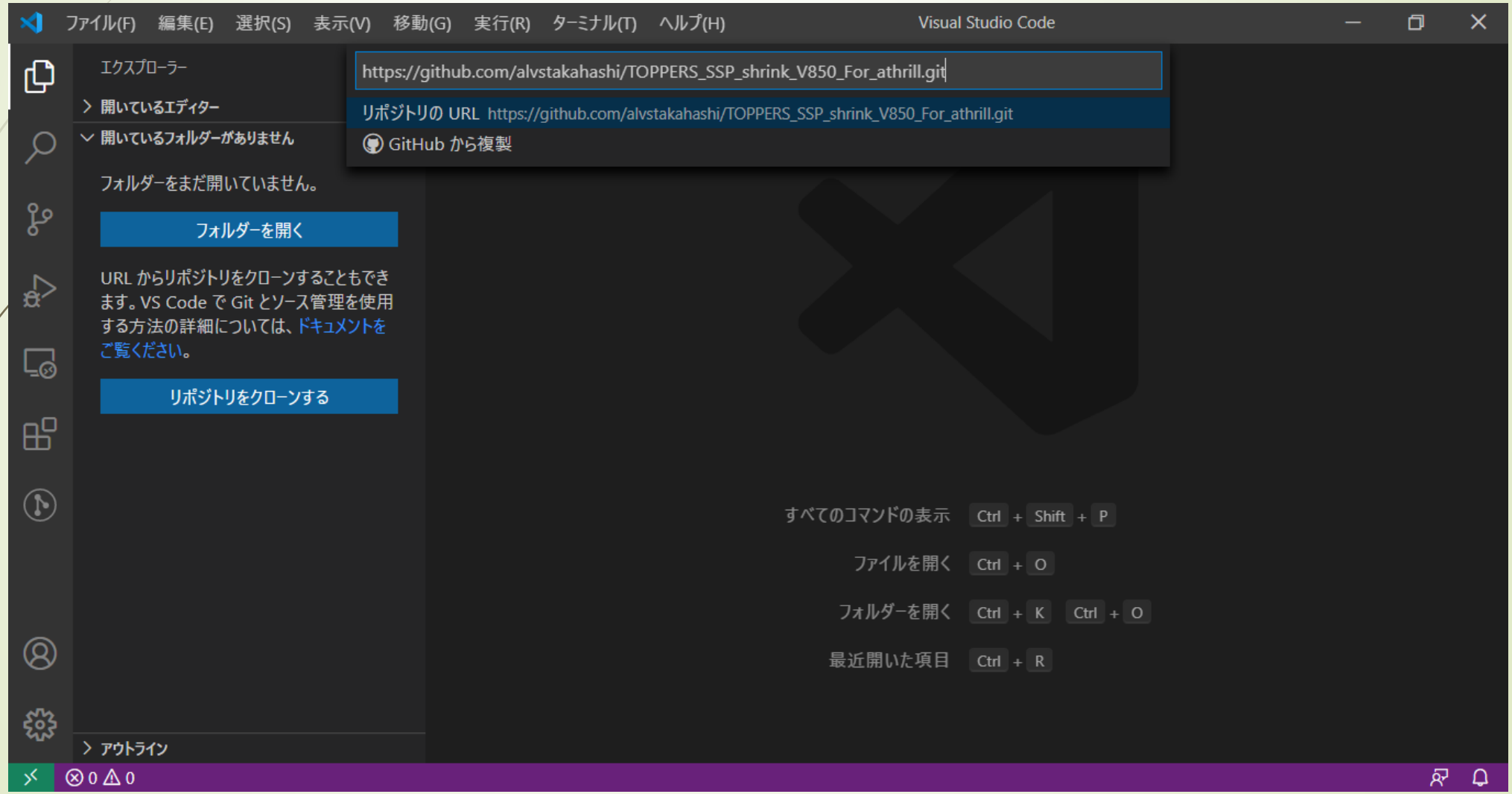
# 1)Vscodeを起動

ここをクリックし、リポジトリをクローンするボタンをクリックします。



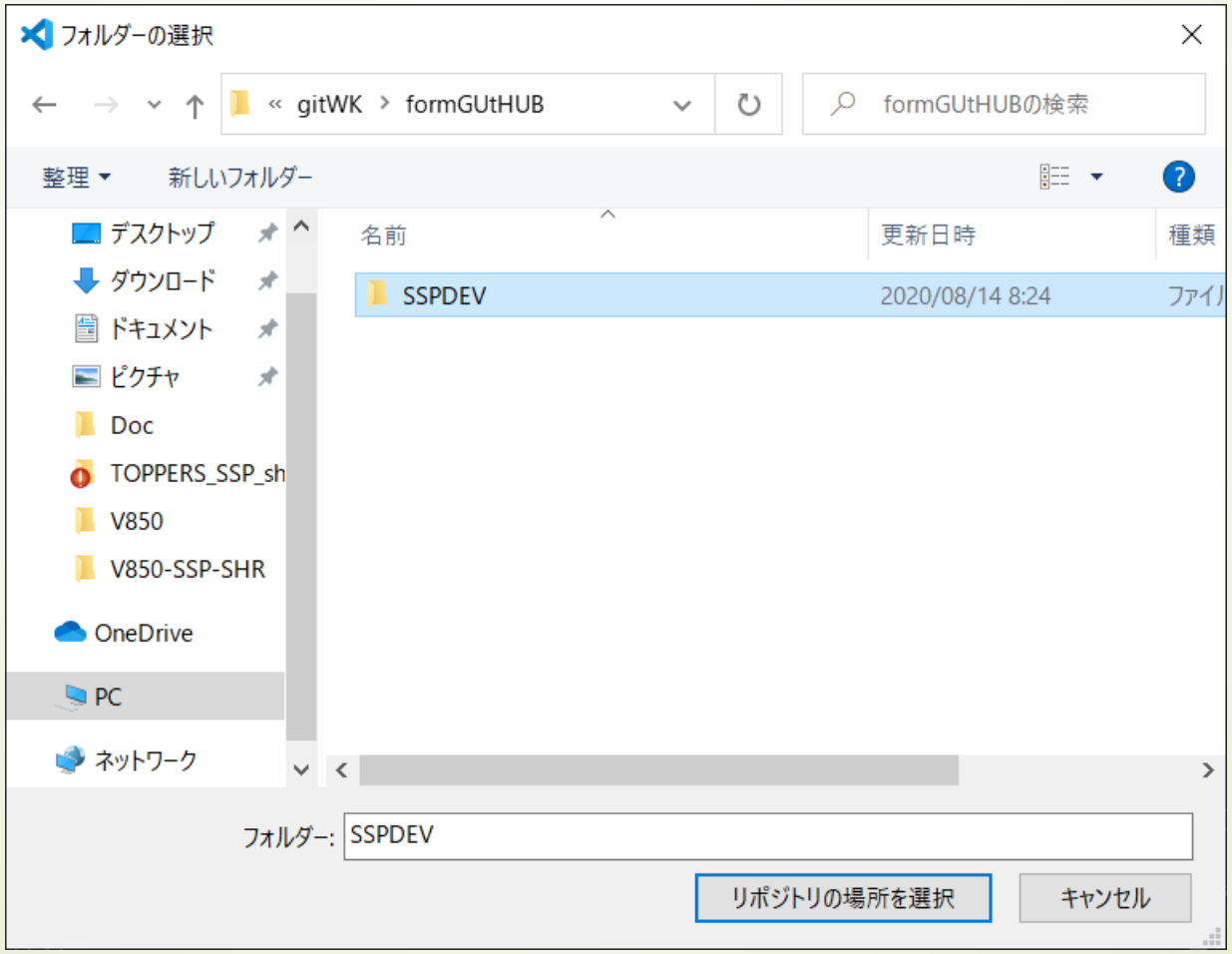
# 2)GithubからTOPPERS_SSP_shrink_V850_For_athrillを取得

https://github.com/alvstakahashi/TOPPERS_SSP_shrink_V850_For_athrill.git
を指定します。



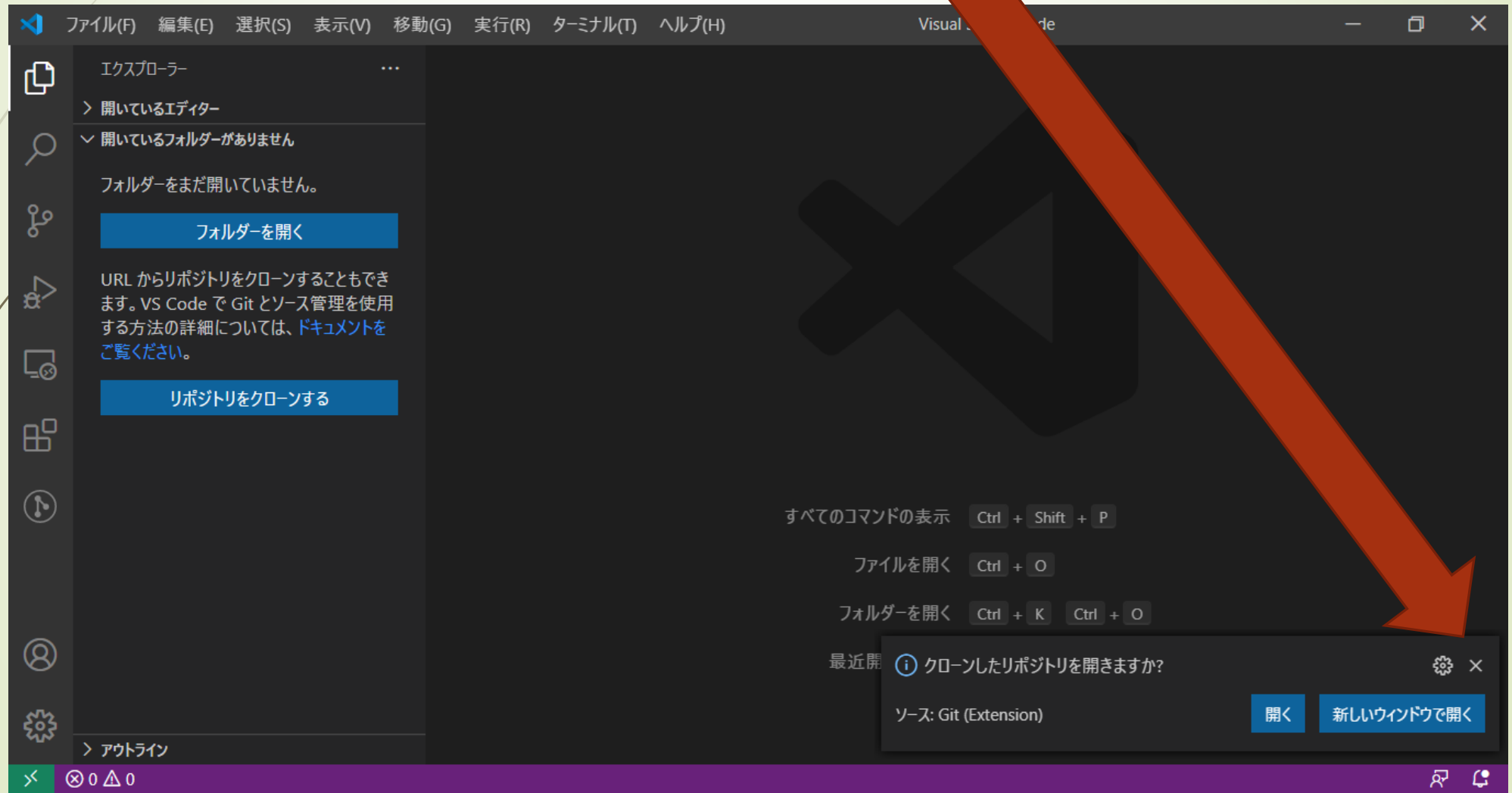
# 3)保存先を指定

PCローカルフォルダを指定します。　次回からここを開きます。



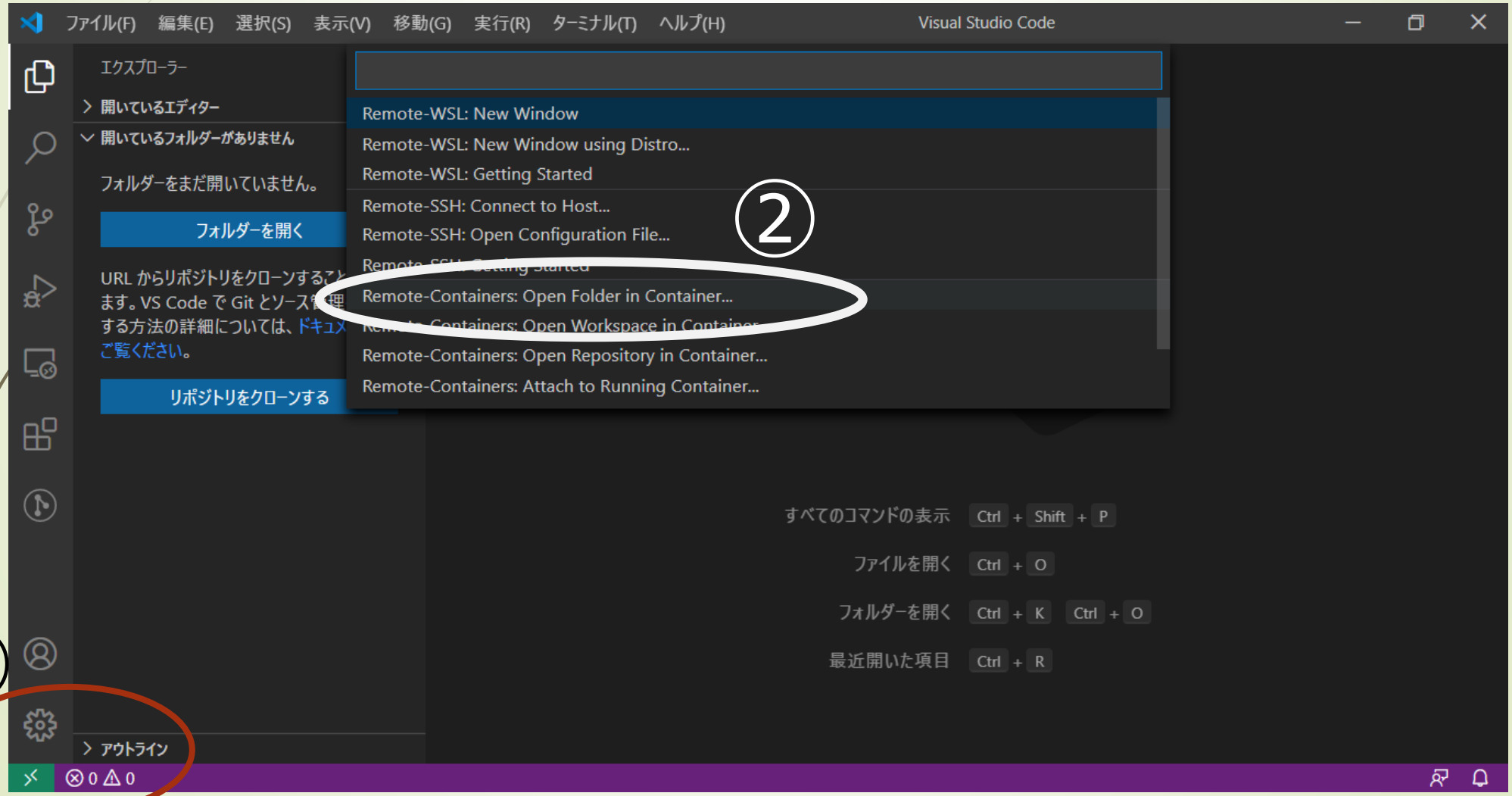
# 4)ここでは開きません

ここを開くのダイアログがでますが、開かず閉じてください。



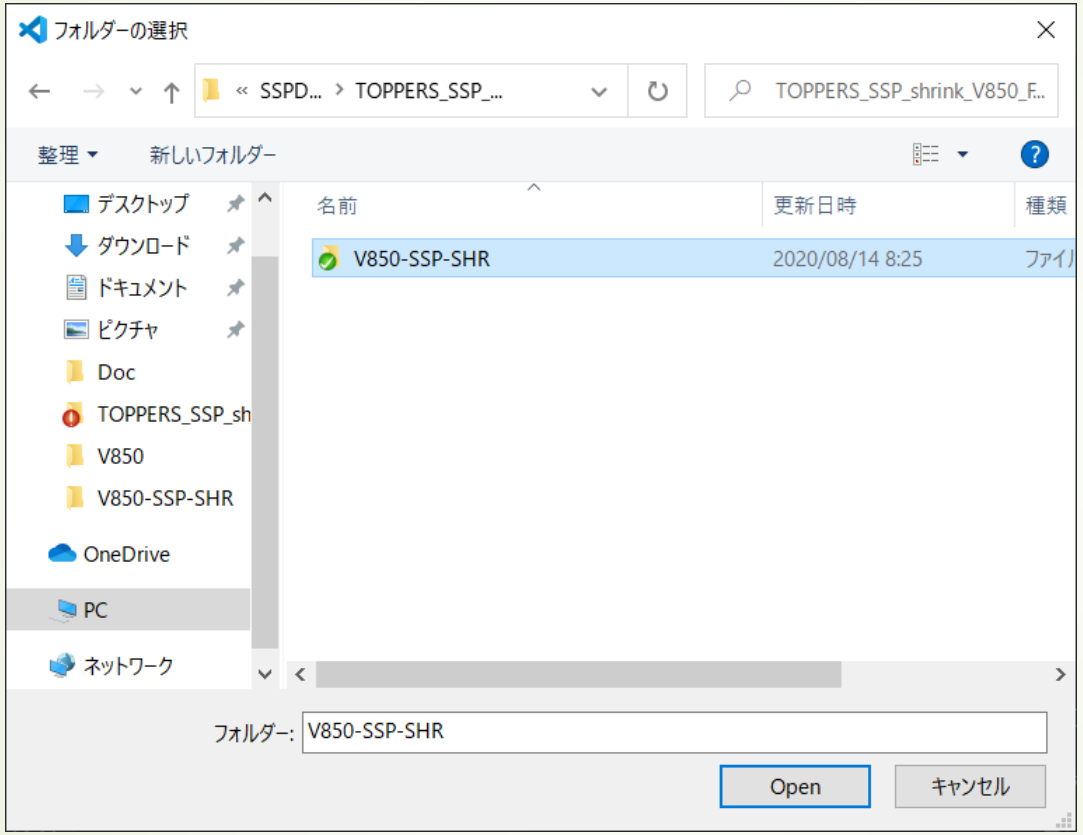
# 5)コンテナを開く

①左下の緑部分をクリック
②プルダウンから
Remore-Containers:Open Fo;der in Container..　を選択します。



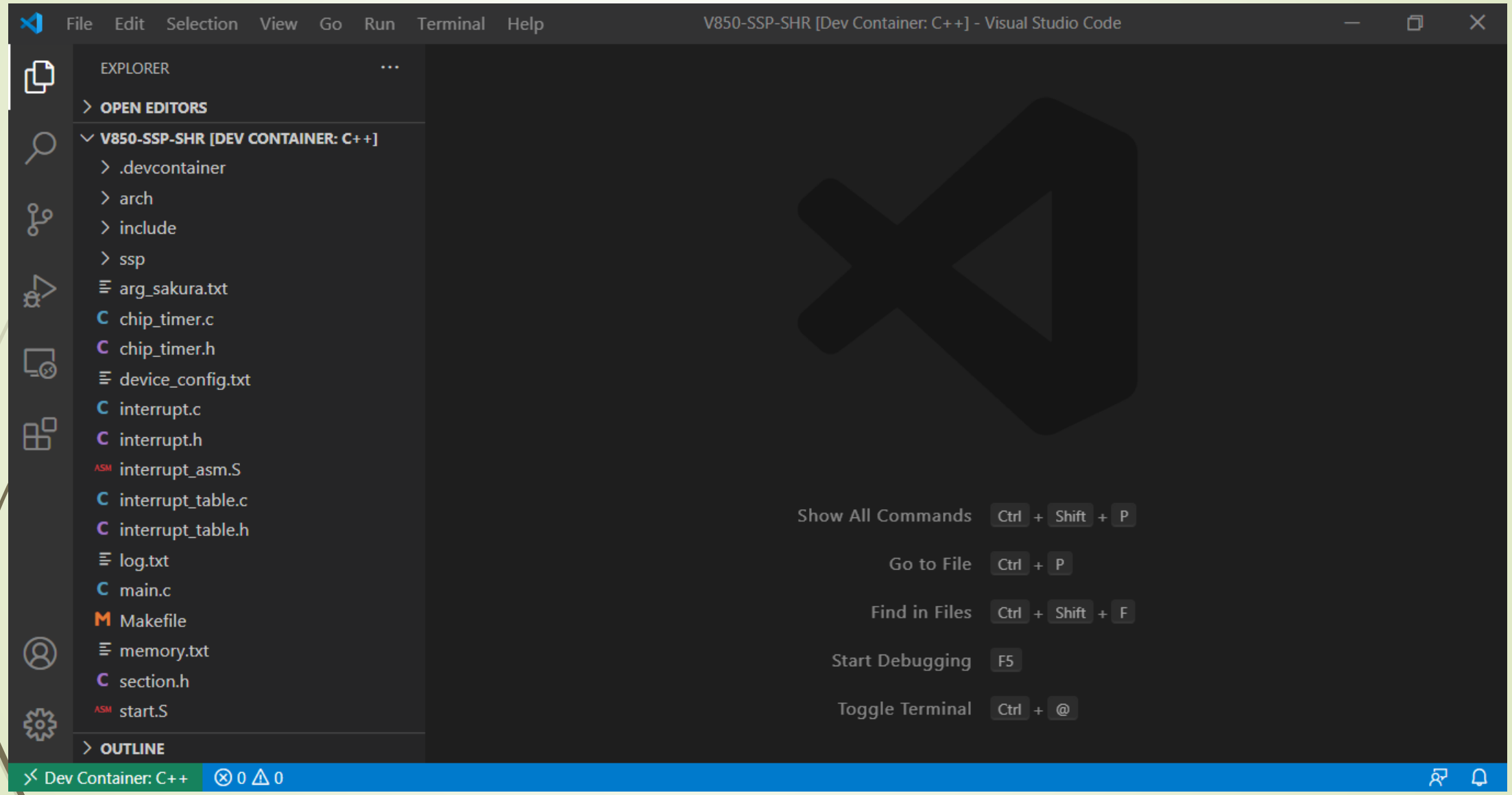
# 6)コンテナを開くフォルダ指定

Githubから取得したフォルダの1階層下のフォルダを指定します。



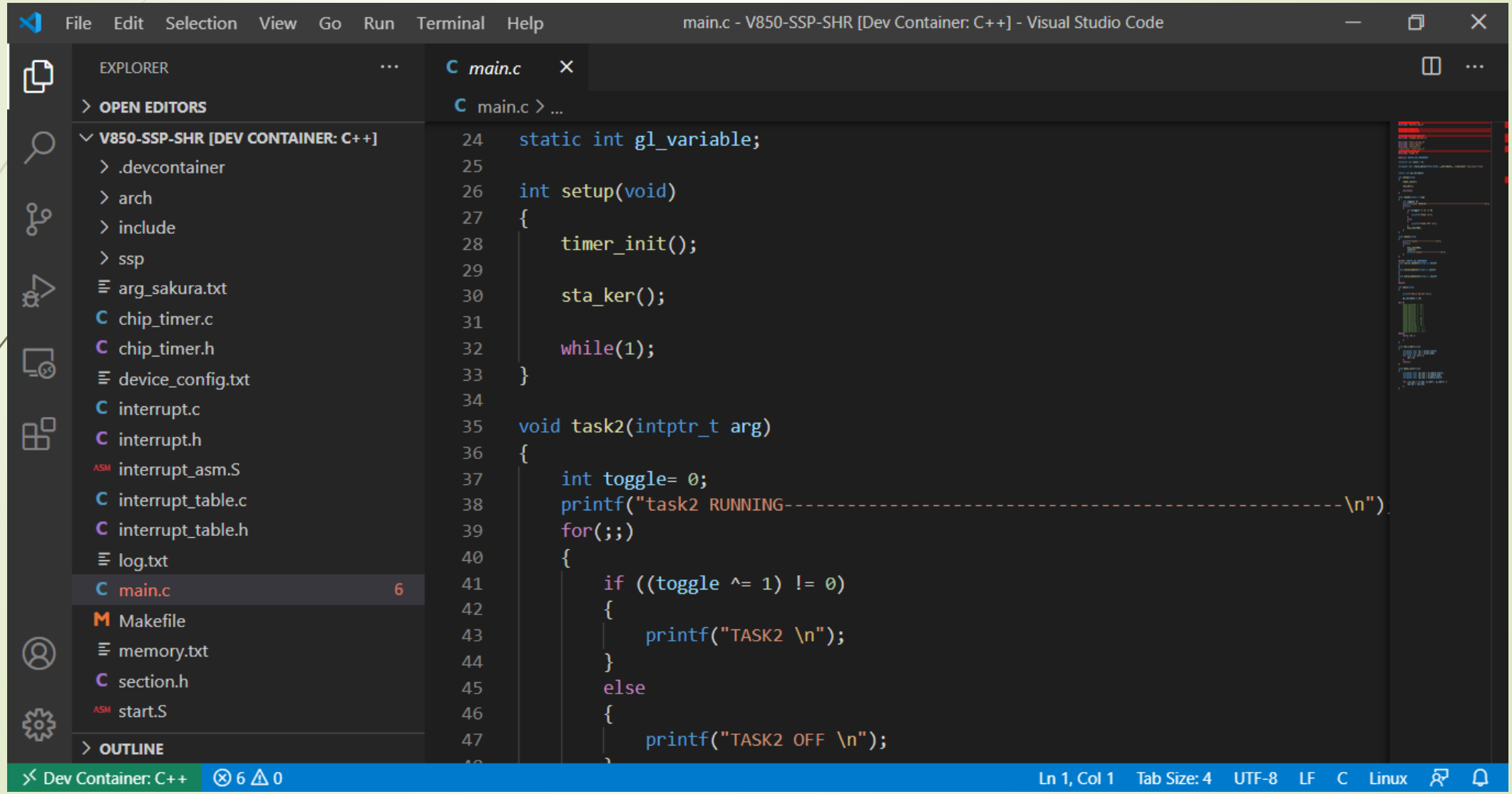

指定後、DockerHUBから読みこむために数分かかります。！

# 7)コンテナ接続完了、開いた状態



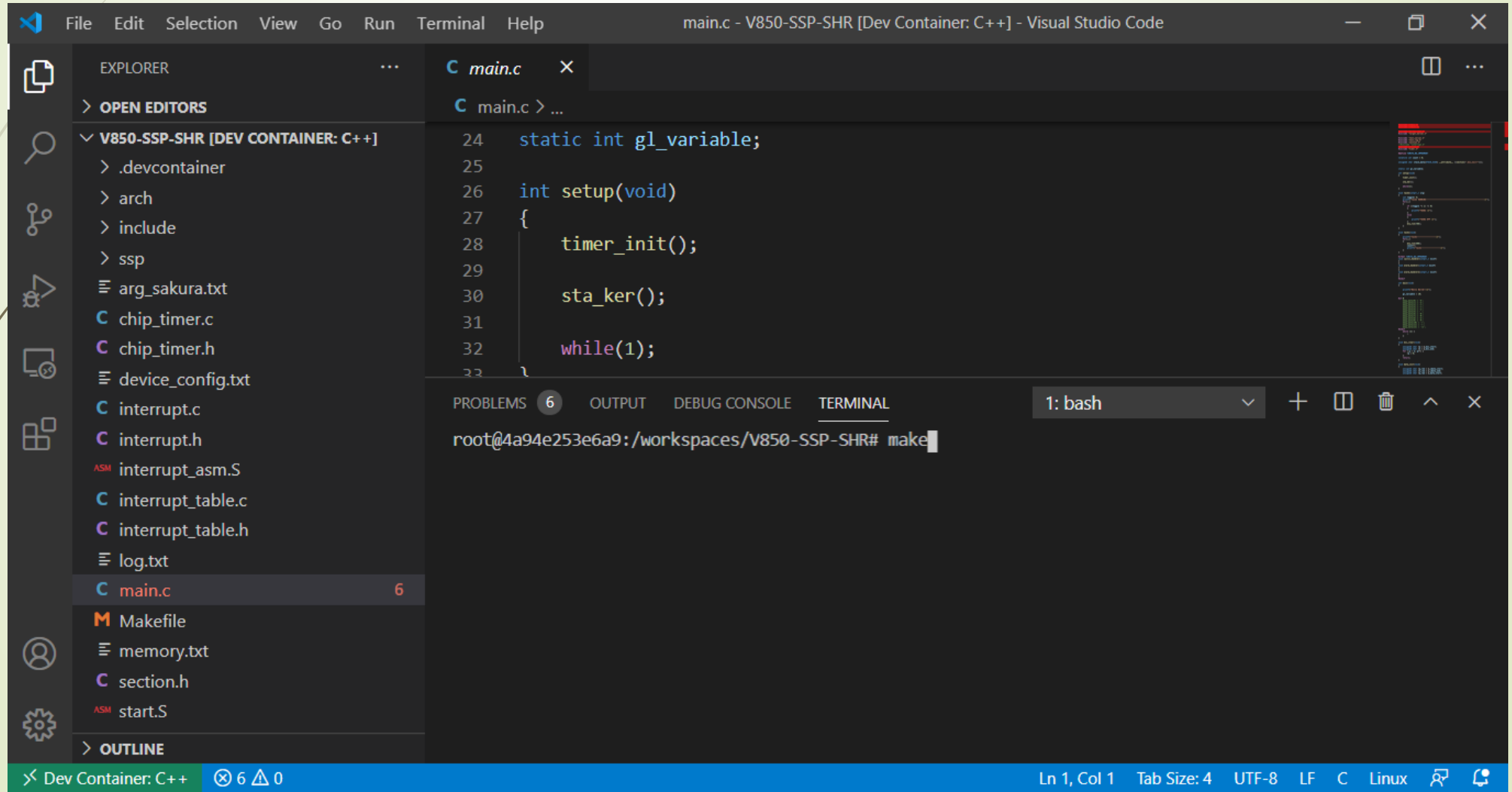
# 8)ソースを開いてみます

左側のファイルツリーからmain.cをクリックして開いてみます。
ここでソースの修正が可能です。

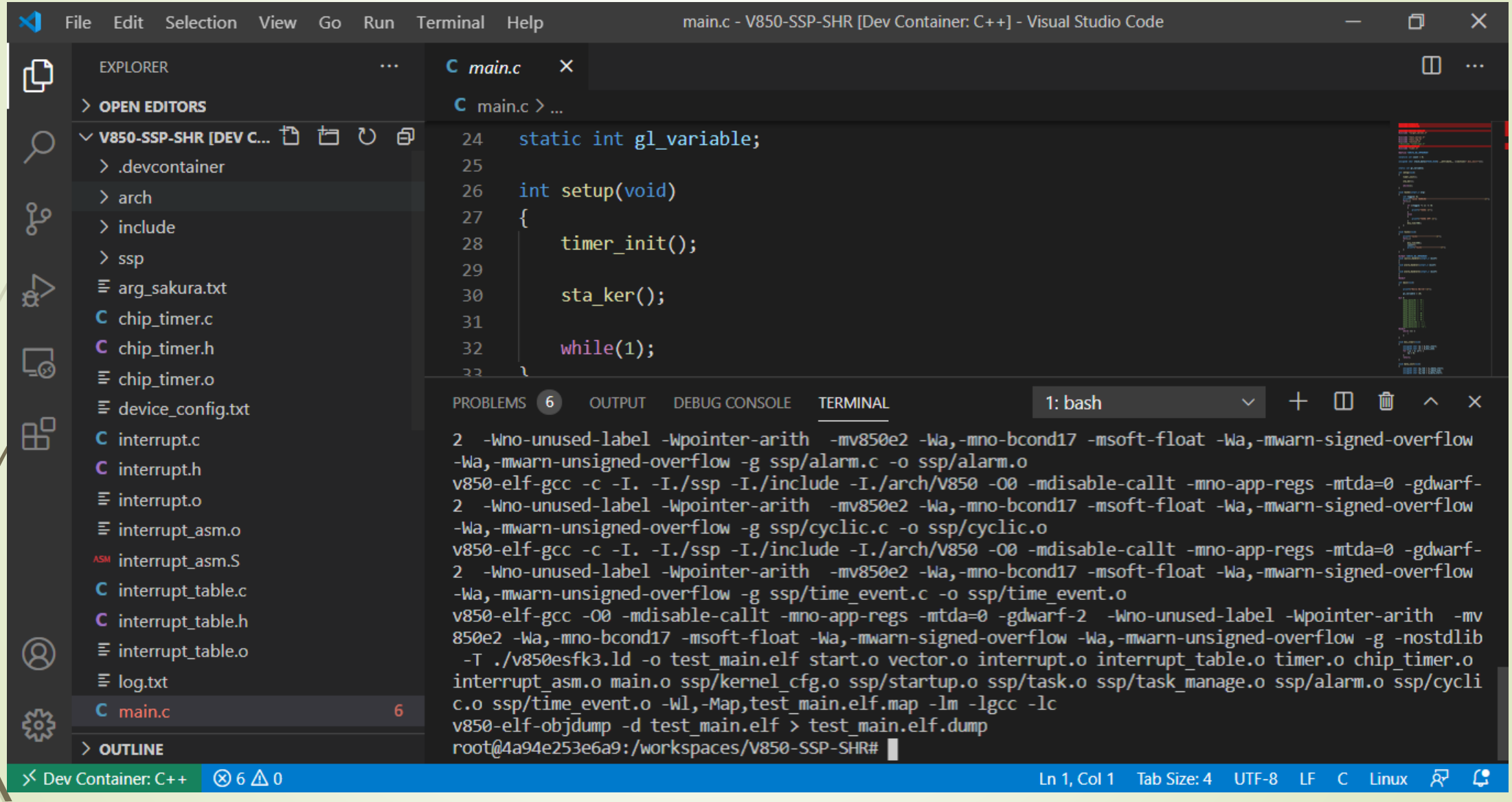


# 9)ビルドします。

ターミナルを新規に開き、make を実行します。
コンテナ側にすでにビルド環境が設定済みなのでmakeするだけです

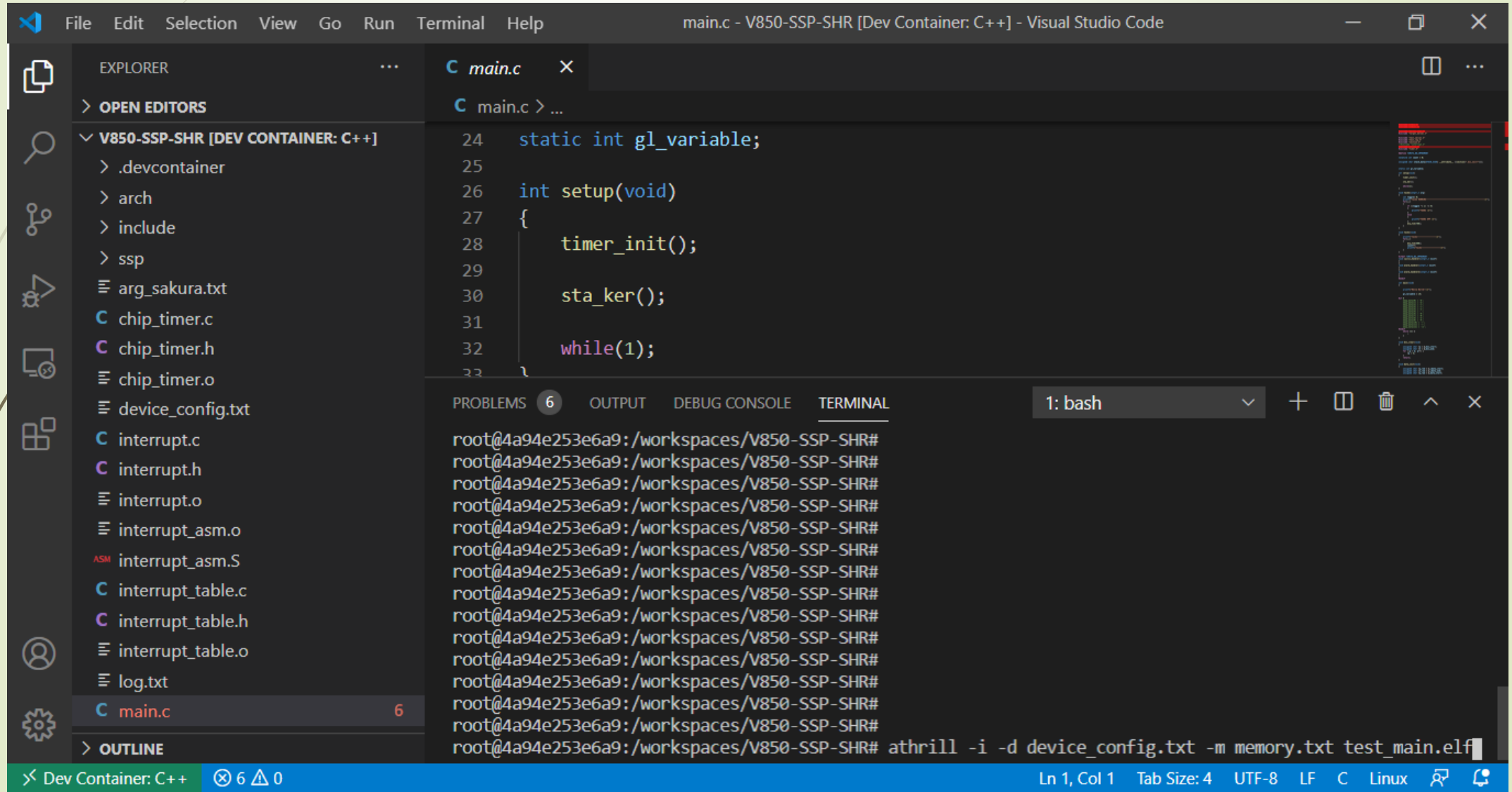


# 10)ビルド完了


File Edit Selection View Go Run Terminal Help
main.c - V850-SSP-SHR [Dev Container: C++] - Visual Studio Code
EXPLORER
OPEN EDITORS
V850-SSP-SHR [DEV C...
.devcontainer
arch
include
ssp
arg_sakura.txt
chip_timer.c
chip_timer.h
chip_timer.o
device_config.txt
interrupt.c
interrupt.h
interrupt.o
interrupt_asm.o
interrupt_asm.S
interrupt_table.c
interrupt_table.h
interrupt_table.o
log.txt
main.c 6
OUTLINE
main.c
main.c > ...
24 static int gl_variable;
25
26 int setup(void)
27 {
28 timer_init();
29
30 sta_ker();
31
32 while(1);
PROBLEMS 6
OUTPUT
DEBUG CONSOLE
TERMINAL
1: bash
2 -Wno-unused-label -Wpointer-arith -mv850e2 -Wa,-mno-bcond17 -msoft-float -Wa,-mwarn-signed-overflow -Wa,-mwarn-unsigned-overflow -g ssp/alarm.c -o ssp/alarm.o
v850-elf-gcc -c -I. -I./ssp -I./include -I./arch/V850 -O0 -mdisable-callt -mno-app-regs -mtda=0 -gdwarf-2 -Wno-unused-label -Wpointer-arith -mv850e2 -Wa,-mno-bcond17 -msoft-float -Wa,-mwarn-signed-overflow -Wa,-mwarn-unsigned-overflow -g ssp/cyclic.c -o ssp/cyclic.o
v850-elf-gcc -c -I. -I./ssp -I./include -I./arch/V850 -O0 -mdisable-callt -mno-app-regs -mtda=0 -gdwarf-2 -Wno-unused-label -Wpointer-arith -mv850e2 -Wa,-mno-bcond17 -msoft-float -Wa,-mwarn-signed-overflow -Wa,-mwarn-unsigned-overflow -g ssp/time_event.c -o ssp/time_event.o
v850-elf-gcc -O0 -mdisable-callt -mno-app-regs -mtda=0 -gdwarf-2 -Wno-unused-label -Wpointer-arith -mv850e2 -Wa,-mno-bcond17 -msoft-float -Wa,-mwarn-signed-overflow -Wa,-mwarn-unsigned-overflow -g -nostdlib -T ./v850esfk3.ld -o test_main.elf start.o vector.o interrupt.o interrupt_table.o timer.o chip_timer.o interrupt_asm.o main.o ssp/kernel_cfg.o ssp/startup.o ssp/task.o ssp/task_manage.o ssp/alarm.o ssp/cyclic.o ssp/time_event.o -Wl,-Map,test_main.elf.map -lm -lgcc -lc
v850-elf-objdump -d test_main.elf > test_main.elf.dump
root@4a94e253e6a9:/workspaces/V850-SSP-SHR#
Dev Container: C++
6 0
Ln 1, Col 1
Tab Size: 4
UTF-8
LF
C
Linux

# 11)実行

athrill -i -d device_config.txt -m memory.txt test_main.elf
で実行します。



# 11)実行終了

ロード後、ｃリターンで実行を始めます。
CTL-C で終了します。
Warring がでますが問題ないです。