

## **preamble**

This document is used to describe the installation and use of the GCC compilation environment for the PY32 microcontroller. The editing function is realized by using VSCode software; the compilation function is realized by using gcc-arm-none-eabi software; the download and burn function is realized by using CooFlash software and PY-LINK emulator, JFlash software and J-Link emulator.

PUYA CONFIDENTIAL

## Contents

<b>1</b>	<b>Installing GCC .....</b>	<b>3</b>
1.1	Download .....	3
1.2	Installation.....	3
<b>2</b>	<b>Installing mingw .....</b>	<b>5</b>
<b>3</b>	<b>Installing VSCode .....</b>	<b>6</b>
3.1	Download .....	6
3.2	Installation.....	6
<b>4</b>	<b>Adding Environment Variables .....</b>	<b>7</b>
<b>5</b>	<b>Makefile files.....</b>	<b>8</b>
<b>6</b>	<b>*.ld link file .....</b>	<b>9</b>
<b>7</b>	<b>Edit, compile, download.....</b>	<b>10</b>
7.1	VSCode Editor .....	10
7.2	GCC Compilation.....	11
7.3	CooFlash Software Download .....	11
7.4	JFlash Software Download.....	12
7.5	JFlash Command Line Download.....	13
7.6	OpenOCD Command Line Download .....	14
<b>8</b>	<b>Creating and Using VSCode Tasks .....</b>	<b>15</b>
8.1	Creating compile and download tasks - tasks.json .....	15
8.2	Creating a debug task - launch.json .....	18
8.3	Copy task folder - .vscode .....	20
<b>9</b>	<b>Version History .....</b>	<b>21</b>

## 1 Installing GCC

### 1.1 downloading

Latest gcc-arm-none-eabi compiler download link: <https://developer.arm.com/downloads/-/gnu-rm>

### 1.2 mounting

Figure 1.2-1. Double-click gcc-arm-none-eabi-10.3-2021.10-win32.exe to start installation



Figure 1.2-2. Select Chinese (Simplified) and click OK.

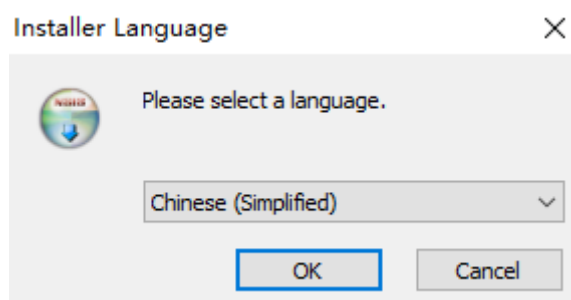


Figure 1.2-3. Click the "Next" button.



Figure 1.2-4. Click the "I Accept" button.

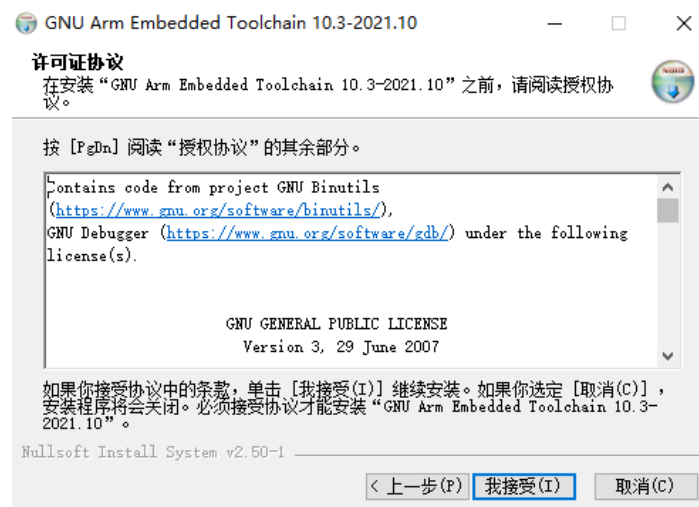


Figure 1.2-5. After selecting the path, click the "Install" button.

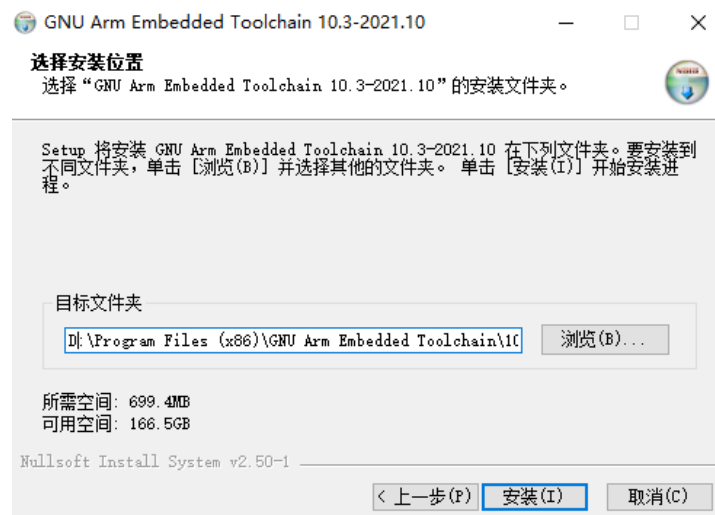


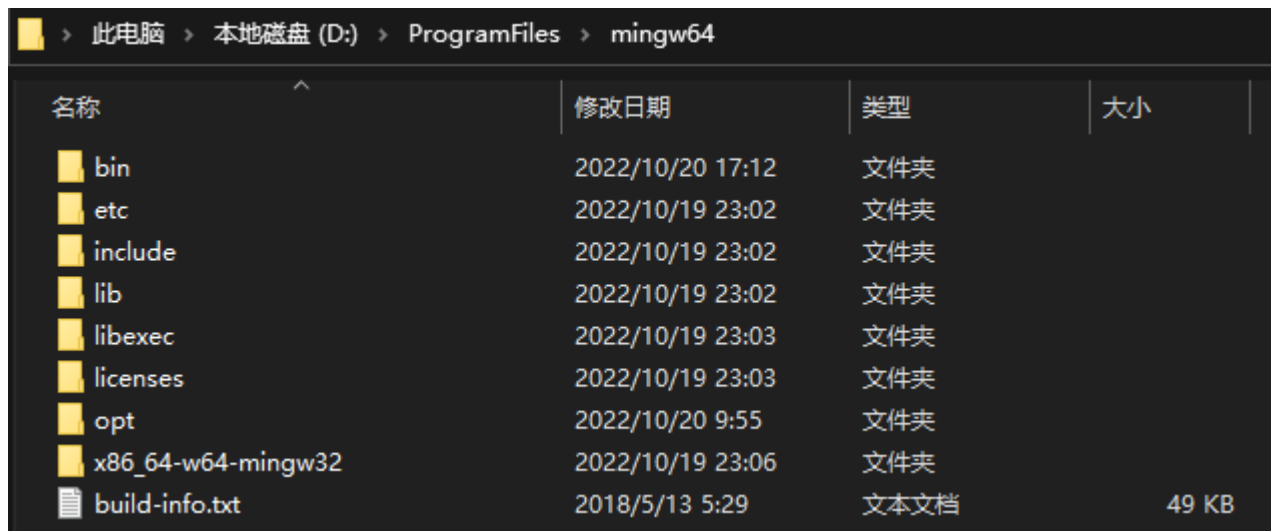
Figure 1.2-6. Check "Add path to environment variable", click "Finish" button.



## 2 Install mingw

This software is green installation-free software, can be used after unzipping, pay attention to unzip the path do not have spaces, Chinese and other special characters, such as D:\ProgramFiles.

Figure 2-1. Extracting mingw64.rar



名称	修改日期	类型	大小
bin	2022/10/20 17:12	文件夹	
etc	2022/10/19 23:02	文件夹	
include	2022/10/19 23:02	文件夹	
lib	2022/10/19 23:02	文件夹	
libexec	2022/10/19 23:03	文件夹	
licenses	2022/10/19 23:03	文件夹	
opt	2022/10/20 9:55	文件夹	
x86_64-w64-mingw32	2022/10/19 23:06	文件夹	
build-info.txt	2018/5/13 5:29	文本文档	49 KB

### 3 Installing VSCode

#### 3.1 downloading

Latest VSCode software download link: <https://code.visualstudio.com/Download>

#### 3.2 mounting

Figure 1.2-1. Double-click VSCodeUserSetup-x64-1.73.1.exe and follow the installation wizard to complete the installation.

名称	修改日期	类型	大小
 VSCodeUserSetup-x64-1.73.1.exe	2022/11/11 10:45	应用程序	90,429 KB

## 4 Adding Environment Variables

Figure 4-1. Select the user variable "Path" and click the Edit button.

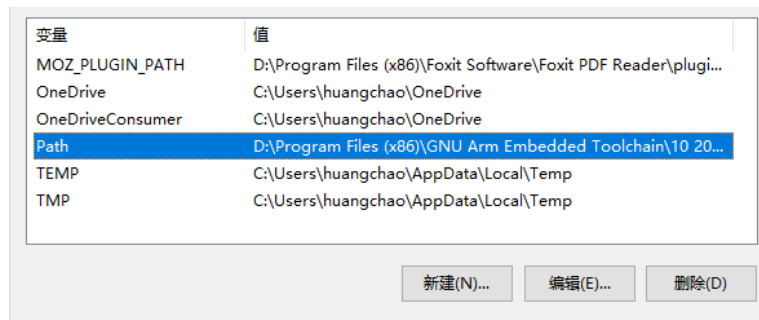


Figure 4-2. After adding the paths for gcc and mingw, click the "OK" button.

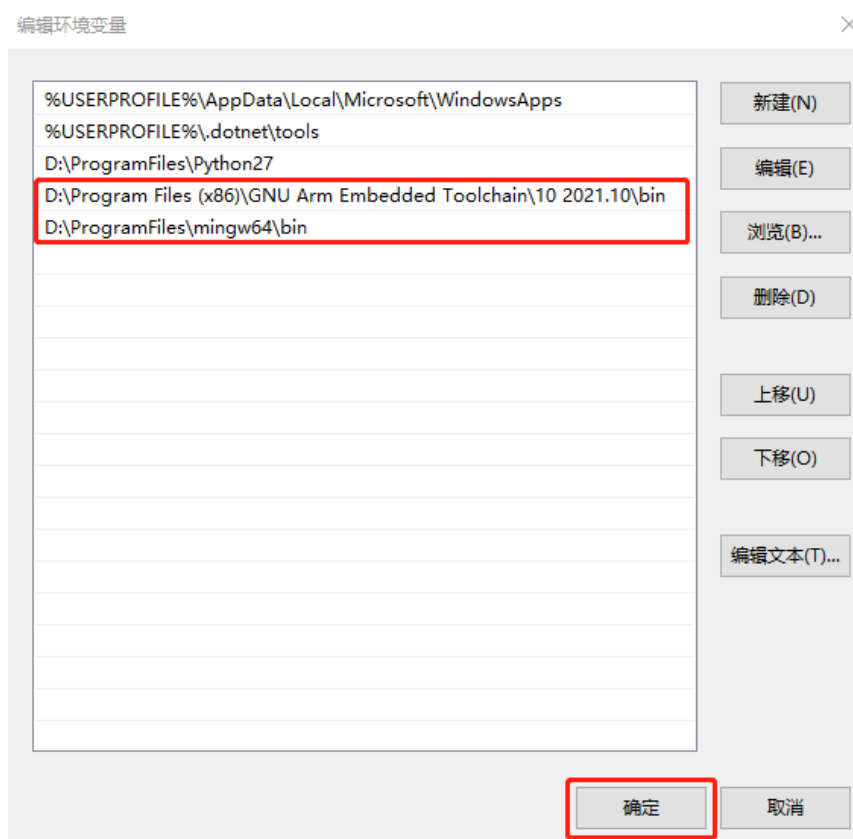
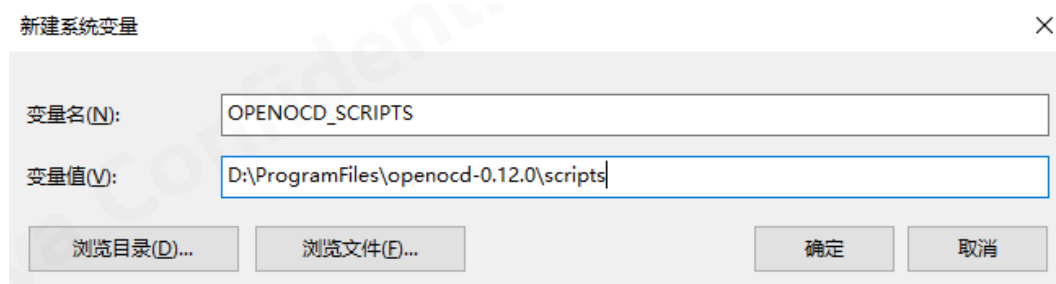


Figure 4-3. Adding System Variable, Name: OPENOCD\_SCRIPTS, Value: openocd scripts folder



## 5 Makefile

name (of a thing)	clarification
TARGET	Name of the generated target file (hex/bin/elf)
OPT	Compile Optimization Level
BUILD_DIR	Path to the generated target file
C_SOURCES	List of C source files to be compiled
ASM_SOURCES	List of S assembly files to be compiled
C_DEFS	Preprocessing Macro Definitions
AS_INCLUDES	Compilation files contain directories
C_INCLUDES	C source files contain directories



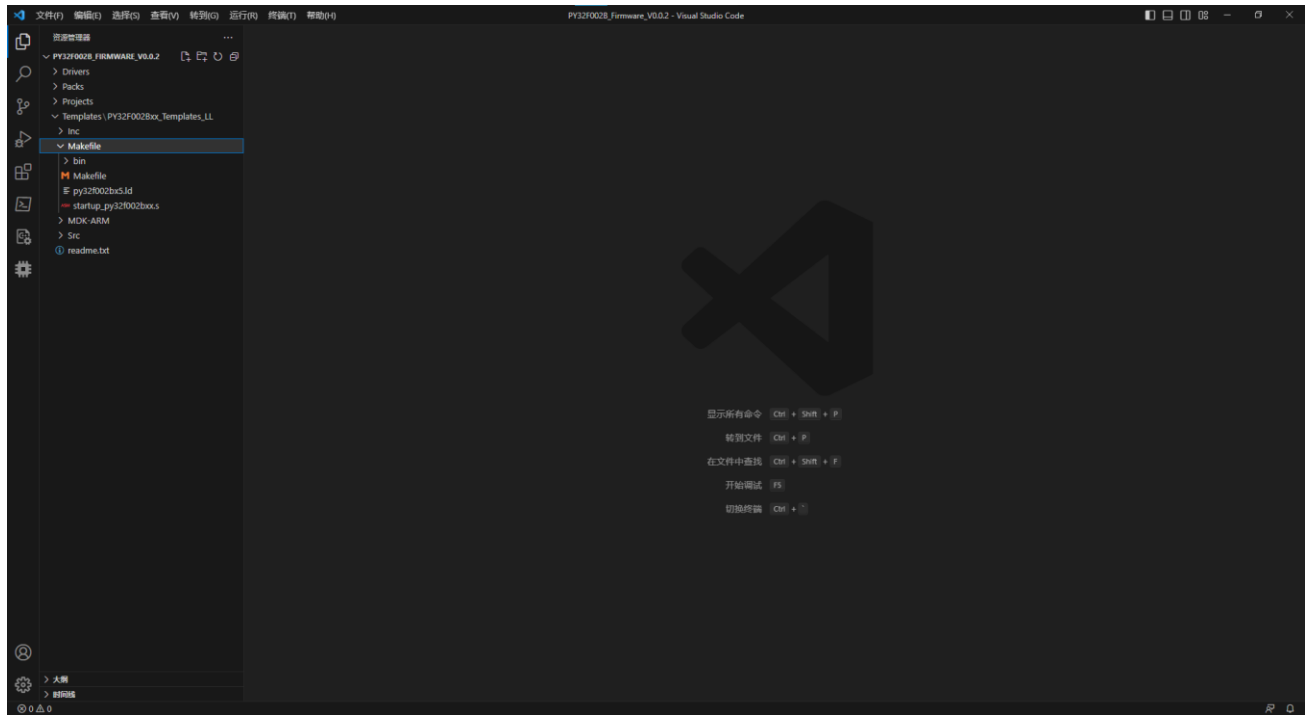
## 6 \*.ld link file

name (of a thing)	clarification
_Min_Heap_Size	Setting the heap size
_Min_Stack_Size	Setting the stack size
RAM (xrw)	SRAM starting address and size
FLASH (rx)	FLASH start address and size

## 7 Edit, compile, download

### 7.1 VSCode Editor

Figure 7.1-1. Unzip PY32F002B\_Firmware\_V0.0.2.rar first, and then open the unzipped folder using VSCode



## 7.2 GCC Compilation

Figure 7.2-1. Right-click on the Makefile folder to be compiled and click "Open in Integrated Terminal".

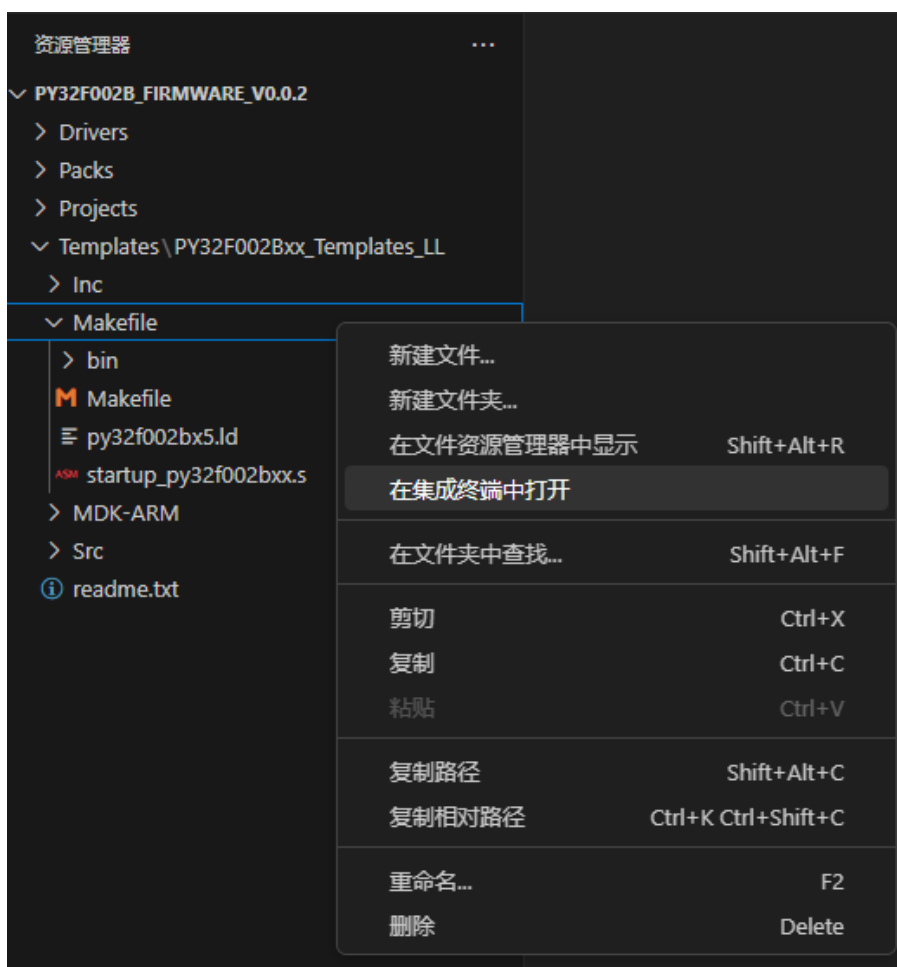


Figure 7.2-2. Type make in the open Powershell terminal and press "Enter".

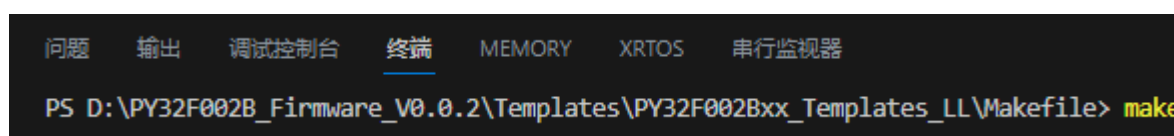
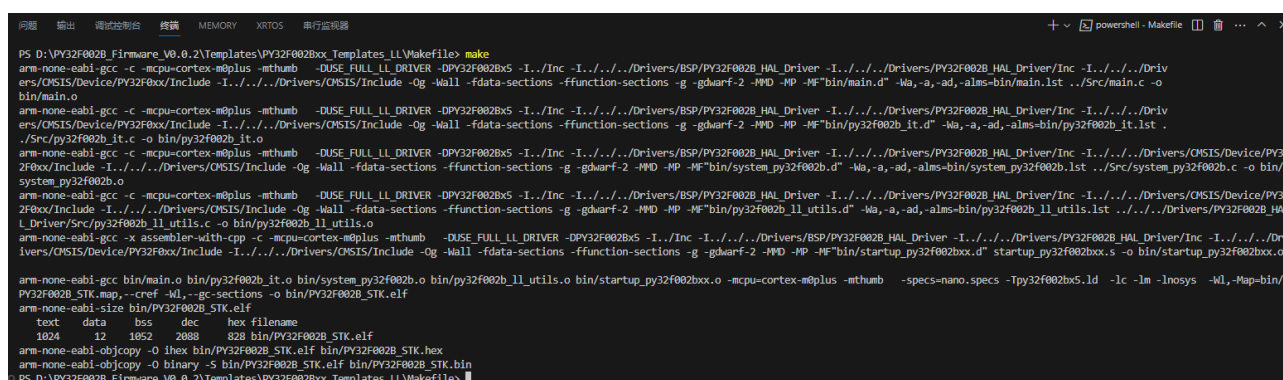


Figure 7.2-3. Compile the target file in hex, bin and elf formats.



## 7.3 CooFlash Software Download

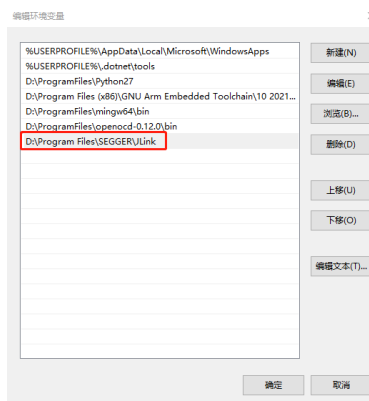
Please refer to the document PY32 Microcontroller CooFlash Download Software User Manual.

#### 7.4 JFlash Software Download

Please refer to the document PY32 Microcontroller JFlash Download Software User Manual.

## 7.5 JFlash Command Line Download

Figure 7.5-1. Adding the JFlash installation directory to the user environment variable.



- JFlash download command

```
JFlash.exe -openprjpy32f030x8.jflash -openbin\PY32F030_STK.elf,0x08000000 -auto -startapp -exit
```

Figure 7.5-2. JFlash download in progress

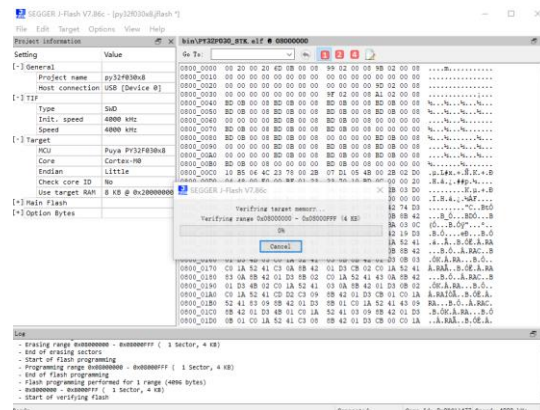
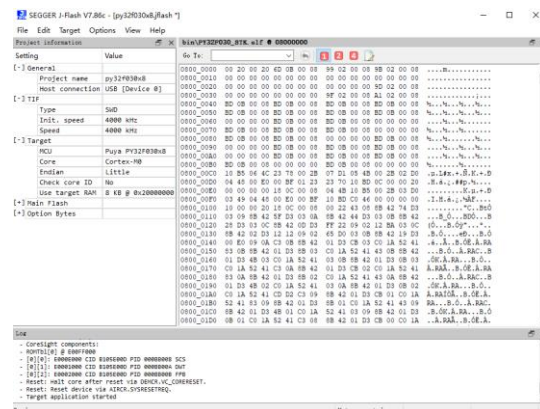
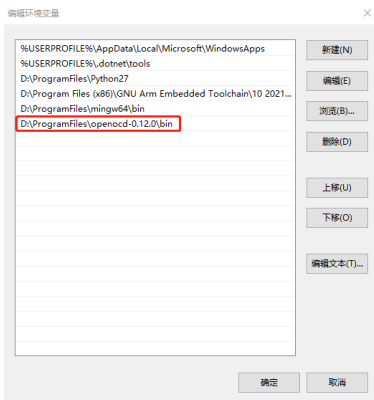


Figure 7.5-3. JFlash download complete.



7.6 OpenOCD Command Line Download

Figure 7.6-1. Adding the openocd bin installation directory to the user environment variable.



- OpenOCD download command  
openocd -s "D:/ProgramFiles/openocd-0.12.0/scripts" -f interface/cmsis-dap.cfg -f target/py32f002a.cfg -c "program bin/PY32F002A\_STK.elf verify reset exit"

Figure 7.6-2. VSCode powershell terminal entering the above command to start the download

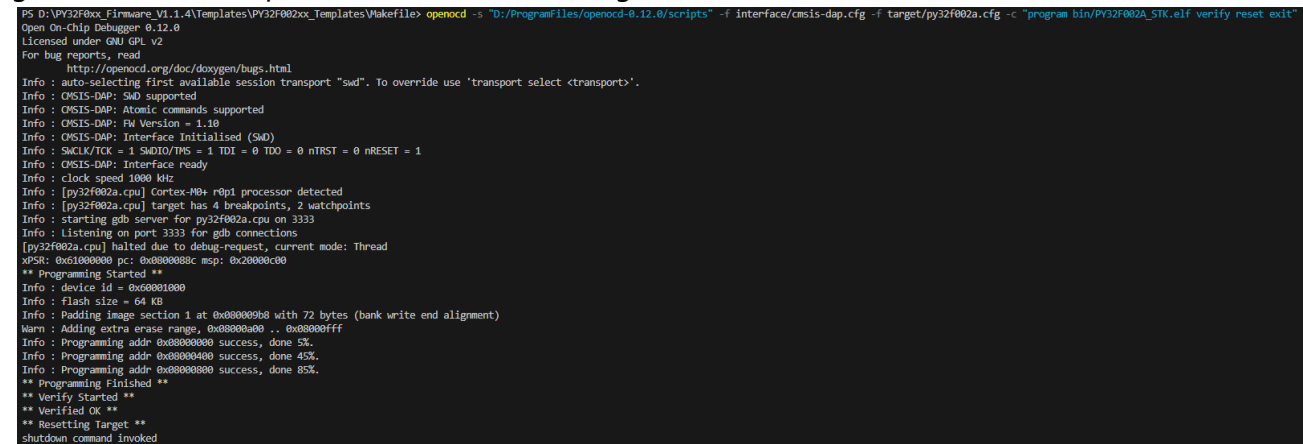


Table 7.6-1. list of openocd cfg files

chip series	target scripts	adapter	interface scripts
PY32F002A	py32f002a.cfg	DAP-LINK	cmsis-dap.cfg
PY32F002B	py32f002b.cfg	J-Link	jlink.cfg
PY32F003	py32f003.cfg	ST-LINK V2	stlink-v2.cfg
PY32F030	py32f030.cfg	ULINK	ulink.cfg
PY32F07X	py32f07x.cfg		
PY32F403	py32f403.cfg		
PY32L020	py32l020.cfg		

## 8 Creating and using VSCode tasks

### 8.1 Create compile and download tasks - tasks.json

Figure 8.1-1. Opening the Makefile folder using VSCode

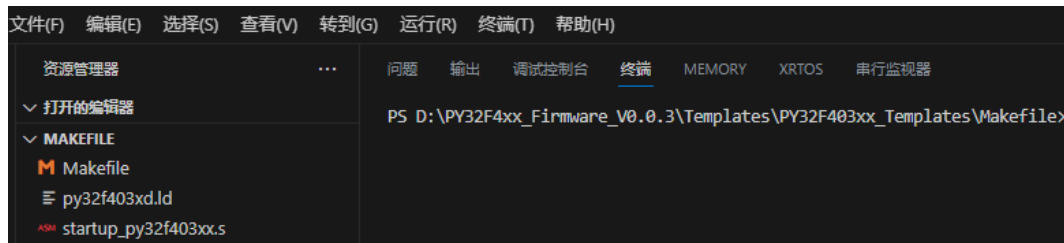


Figure 8.1-2. The menu bar selects "Terminal", "Configure Tasks", and "Create tasks.json file using template".

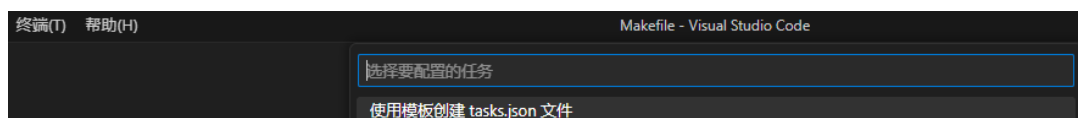


Figure 8.1-3. Select "Others to run an example of any external command".

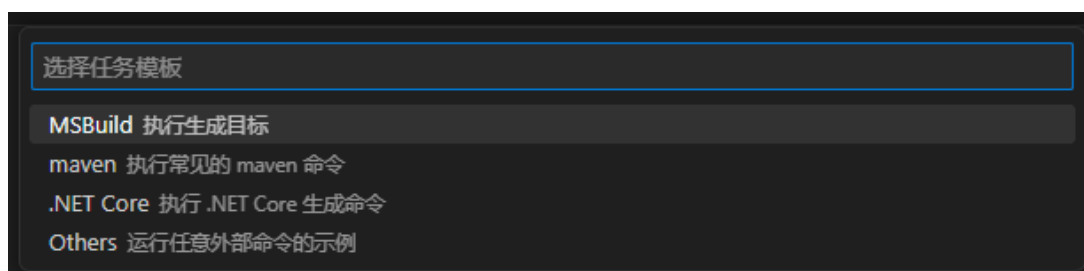


Figure 8.1-4. VSCode automatically generates tasks.json file

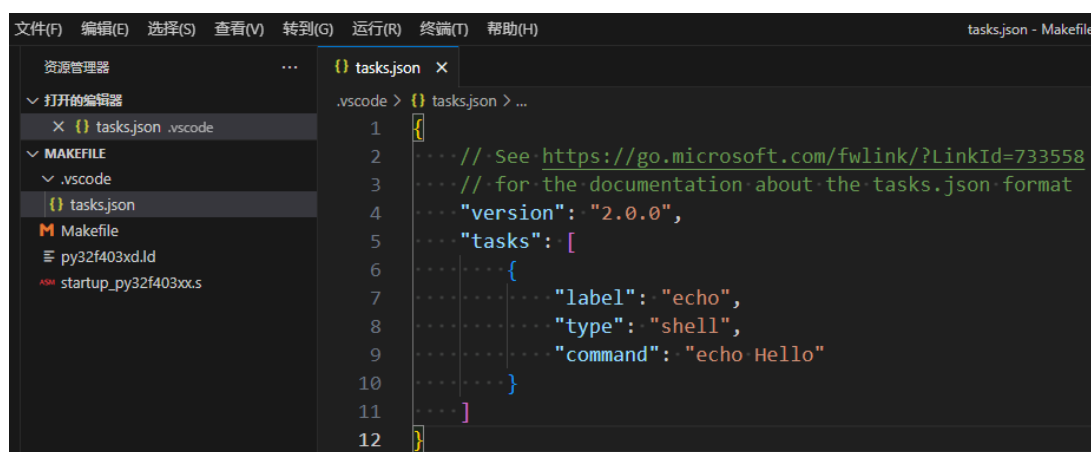


Figure 8.1-5. Modifying the tasks.json file and saving it

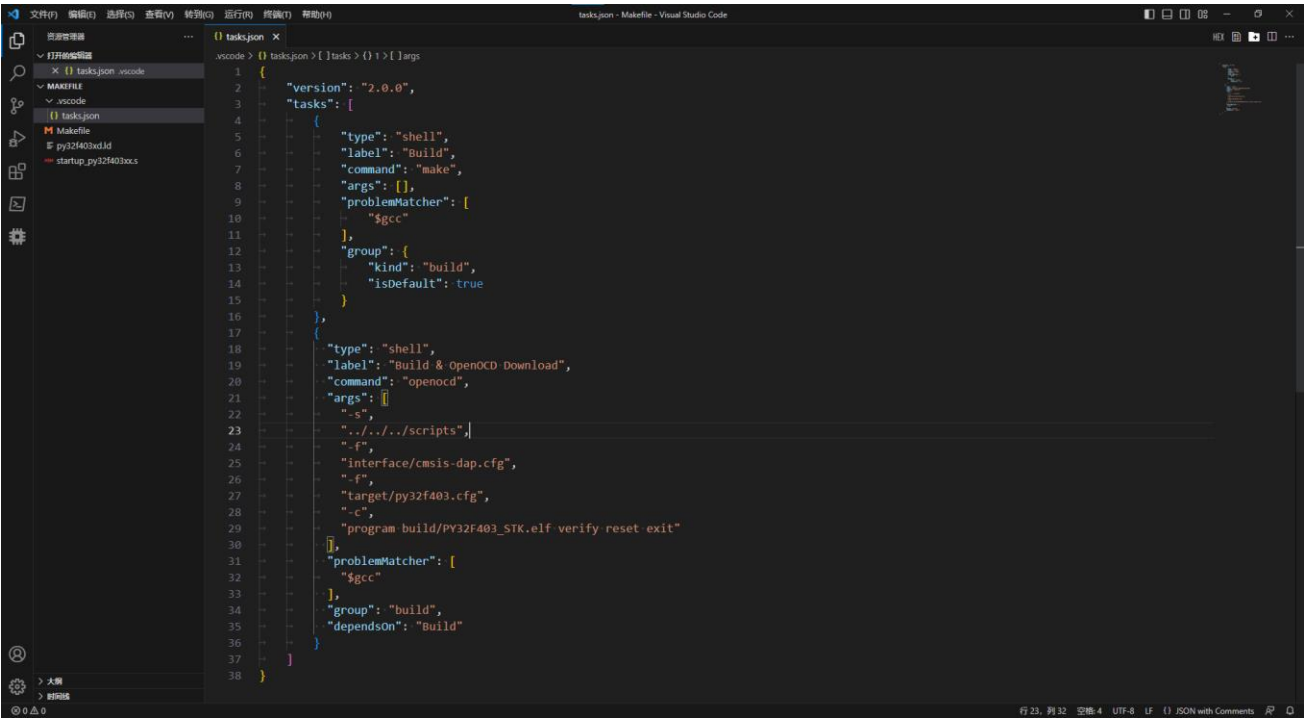


Figure 8.1-6. Select "Terminal" and "Run Tasks" from the menu bar.

Two new tasks appear: Build and Build & OpenOCD Download.

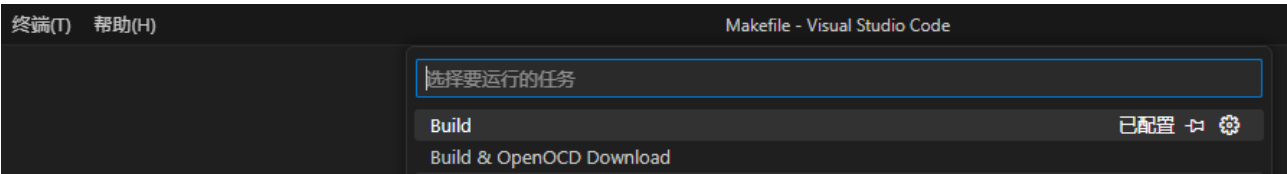
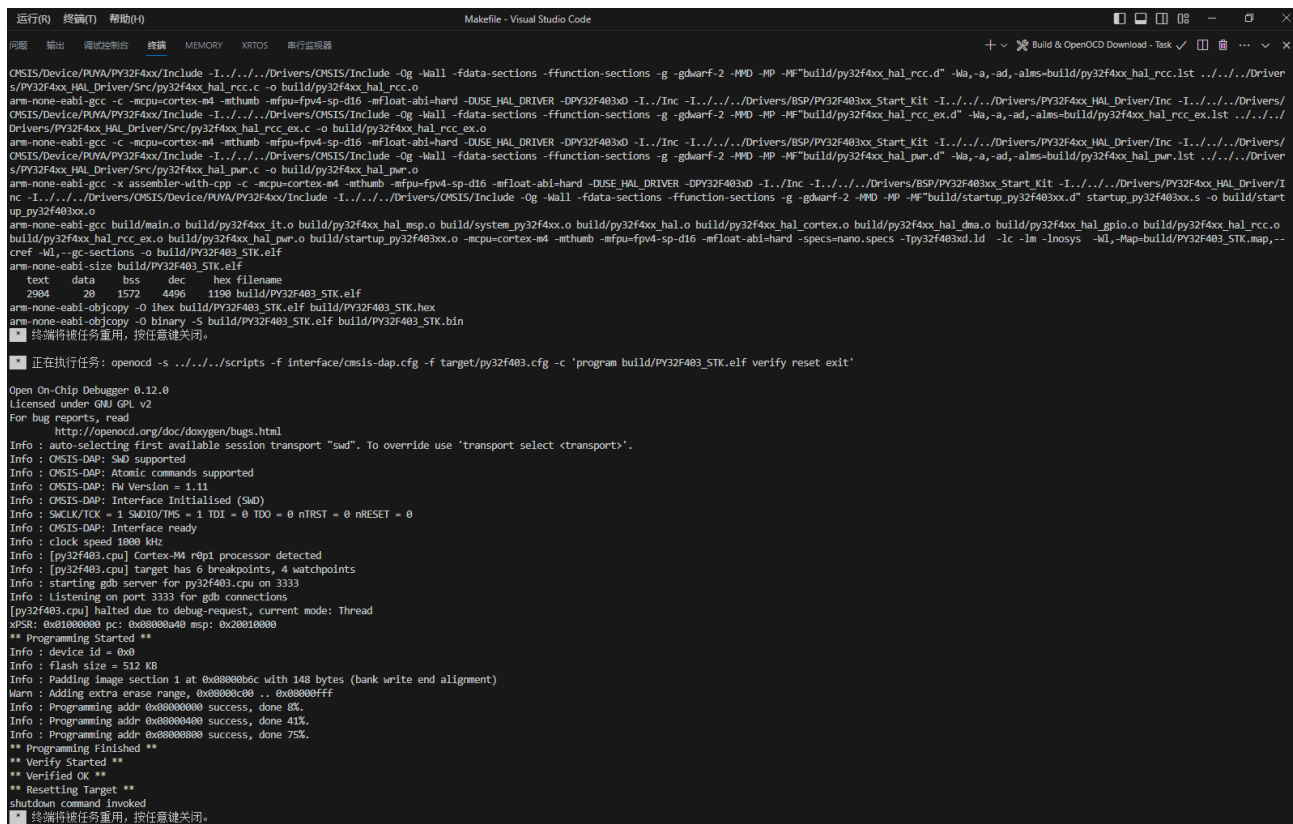


Figure 8.1-7. Copying the scripts script file

此电脑 > 本地磁盘 (D:) > PY32F4xx_Firmware_V0.0.3 > scripts			
名称	修改日期	类型	大小
interface	2023/6/8 10:22	文件夹	
target	2023/6/8 10:22	文件夹	
mem_helper.tcl	2022/12/11 14:39	Altium Script Do...	1 KB
py32f403xx.svd	2023/3/15 14:52	SVD 文件	524 KB



Figure 8.1-8. Run the Build & OpenOCD Download task to perform the compilation and download in turn



## 8.2 Creating a debug task - launch.json

Figure 8.2-1. Select "Run and Debug" in the leftmost toolbar.

Then click on the blue font "Create launch.json" and select "Cortex Debug".

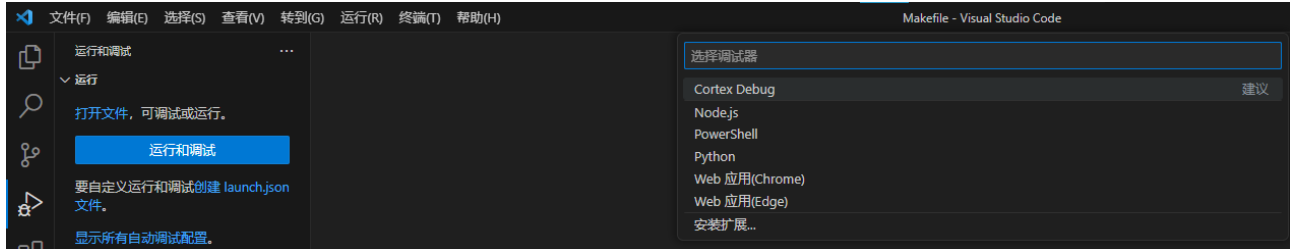


Figure 8.2-2. Modifying the launch.json file

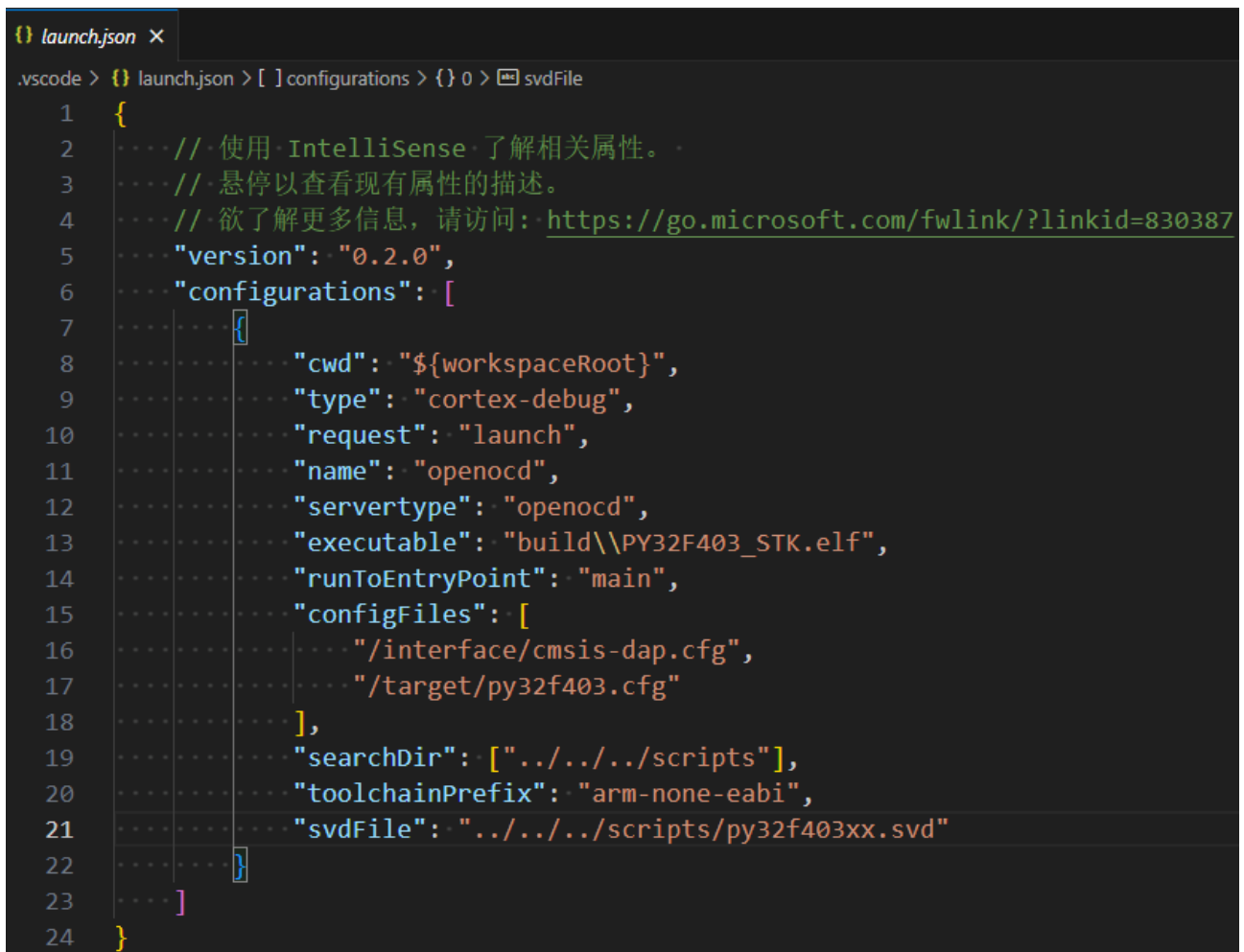
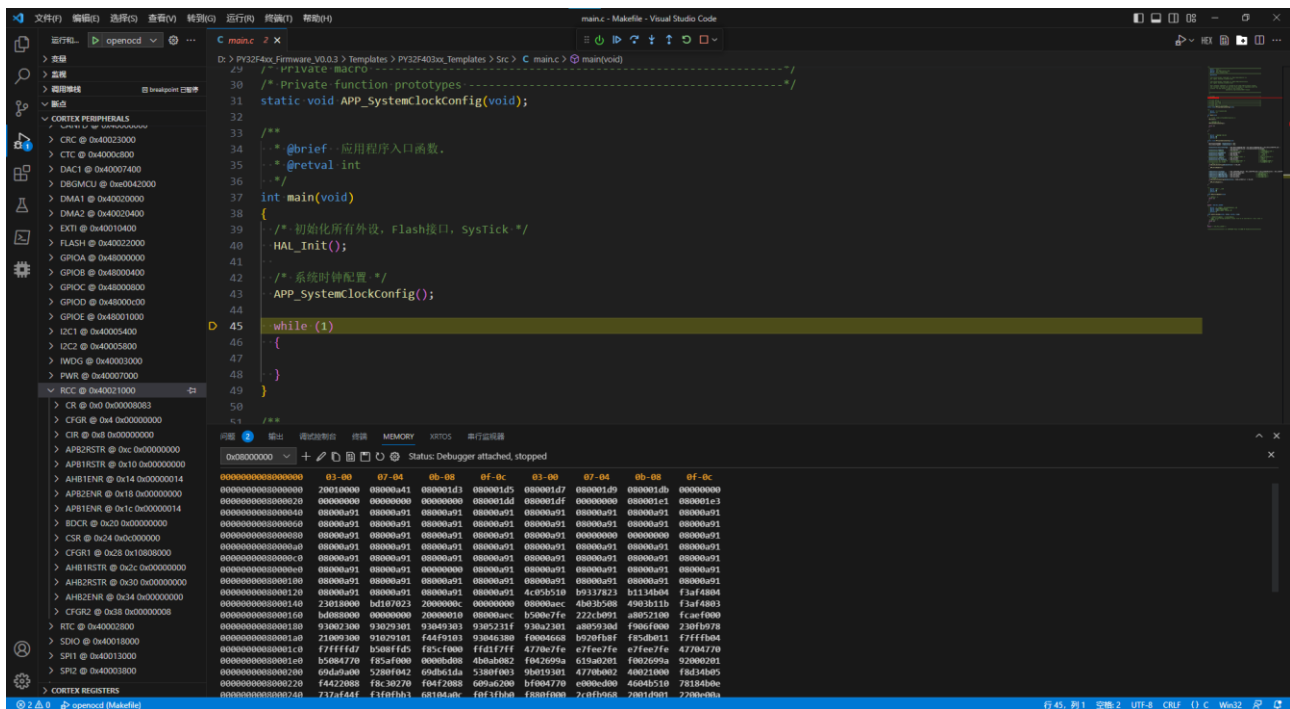
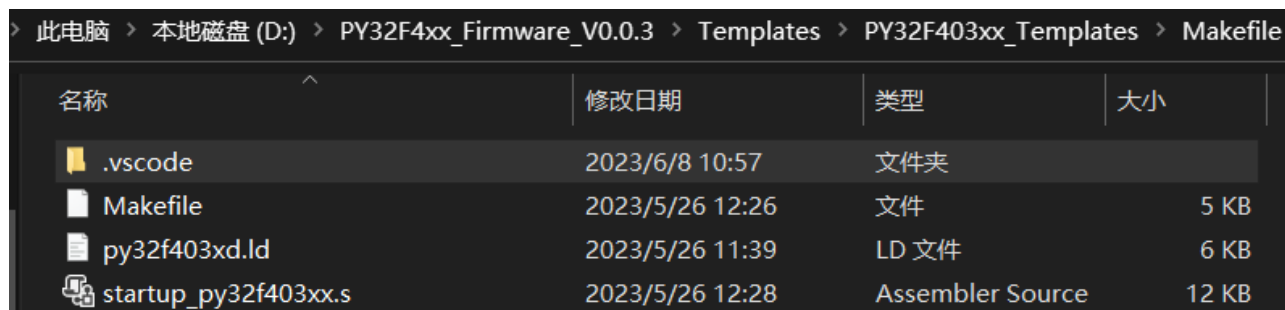


Figure 8.2-3. Click on the green triangle button to the left of "openocd" in the left task window to start Cortex-Debug debugging.



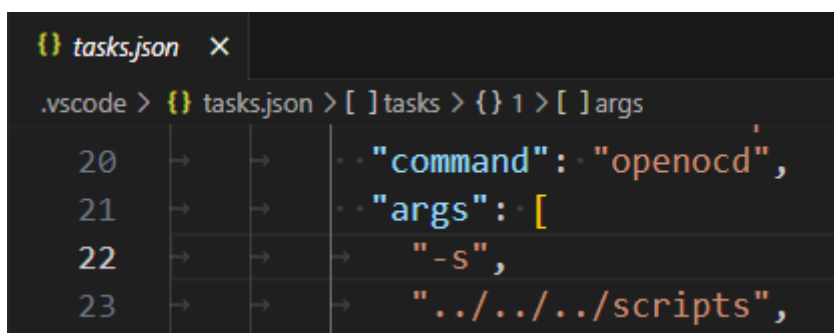
## 8.3 Copy task folder - .vscode

Figure 8.3-1. Copy all the .vscode folders under the Makefile folder to the Makefile folders of other projects.



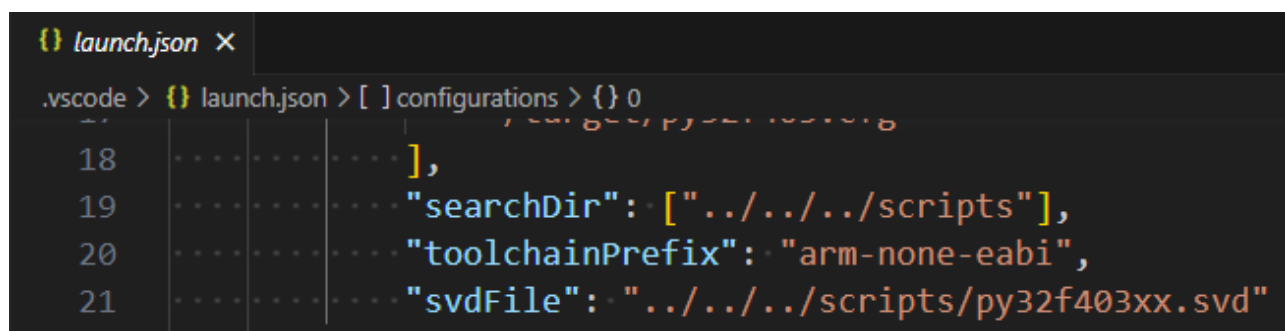
名称	修改日期	类型	大小
.vscode	2023/6/8 10:57	文件夹	
Makefile	2023/5/26 12:26	文件	5 KB
py32f403xd.ld	2023/5/26 11:39	LD 文件	6 KB
startup_py32f403xx.s	2023/5/26 12:28	Assembler Source	12 KB

Figure 8.3-2. Modify the path to the scripts folder inside the tasks.json file.



```
{} tasks.json X
.vscode > {} tasks.json > [ ] tasks > {} 1 > [ ] args
20  → → → → "command": "openocd",
21  → → → → "args": [
22  → → → →   "-s",
23  → → → →   "../../../../../scripts",
```

Figure 8.3-3. Modify the path to the scripts folder and svd file inside the launch.json file.



```
{} launch.json X
.vscode > {} launch.json > [ ] configurations > {} 0
18  → → → → ],
19  → → → → "searchDir": ["../../../../scripts"],
20  → → → → "toolchainPrefix": "arm-none-eabi",
21  → → → → "svdFile": "../../../../scripts/py32f403xx.svd"
```

9 Version History

Version	Date	Description
V1.0	2024-06-12	Initial version



Puya Semiconductor Co.

**IMPORTANT NOTICE**

Puya Semiconductor reserves the right to make changes without further notice to any products or specifications herein. Puya Semiconductor does not Puya Semiconductor does not assume any responsibility for use of any its products for any particular purpose, nor does Puya Semiconductor assume any liability arising out of the application or use of any its products or circuits. Puya Semiconductor does not assume any responsibility for use