# WB32F10x 搭建 GCC 开发环境

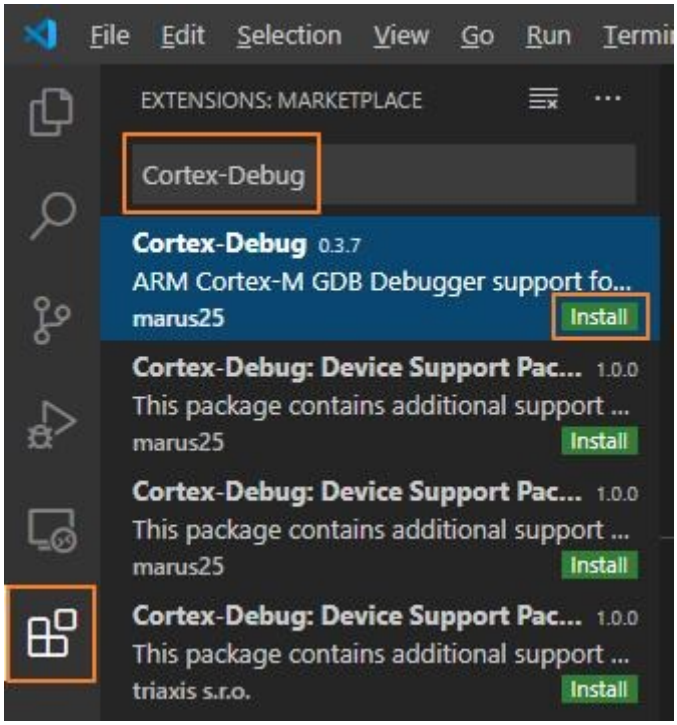
常州韦斯佰瑞电子科技有限公司

# 目录

# 1 安装必要软件

Step 01. 安装 Visual Studio Code: https://code.visualstudio.com/

VS Code 安装完成后，打开 VS Code 软件安装 **Cortex-Debug** 扩展插件。

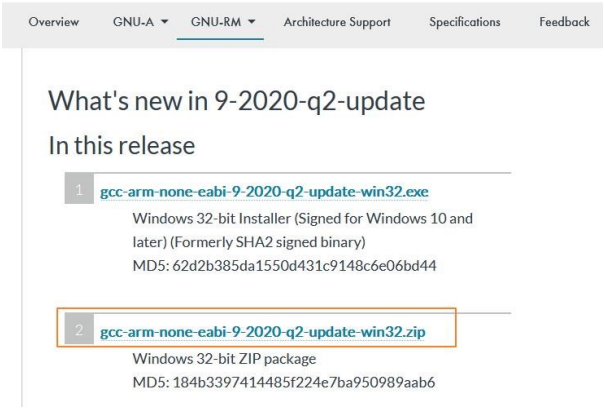



Step 02. 需要一个 J-Link 下载调试工具，并安装 J-Link 驱动程序:
https://www.segger.com/downloads/jlink

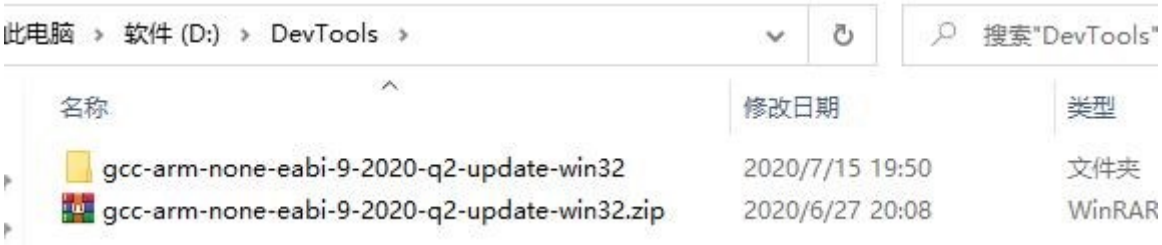
Step 03. 新建一个文件夹用于存放开发需要用到的工具。在本教程中新建文件夹的路径是:
D:\DevTools

Step 04. 下载 GCC-ARM 编译工具链压缩包，并将其放在 D:\DevTools 文件夹中:

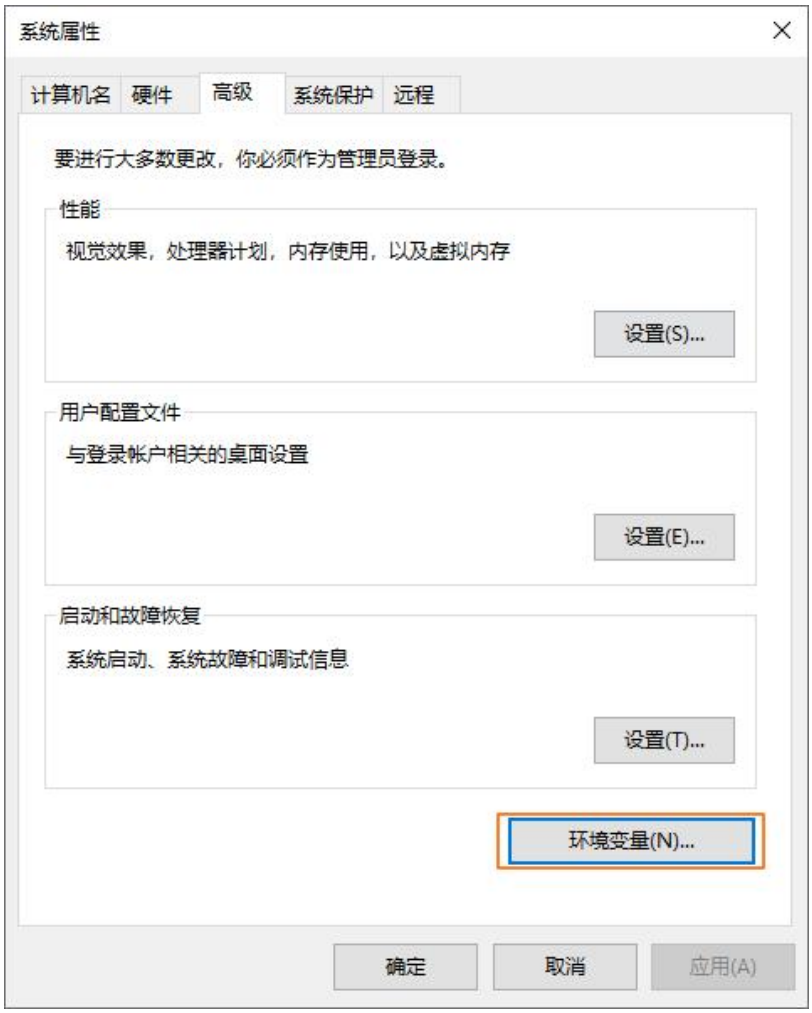
https://developer.arm.com/tools-and-software/open-source-software/developer-tools/gnu-toolchain/gnu-rm/downloads

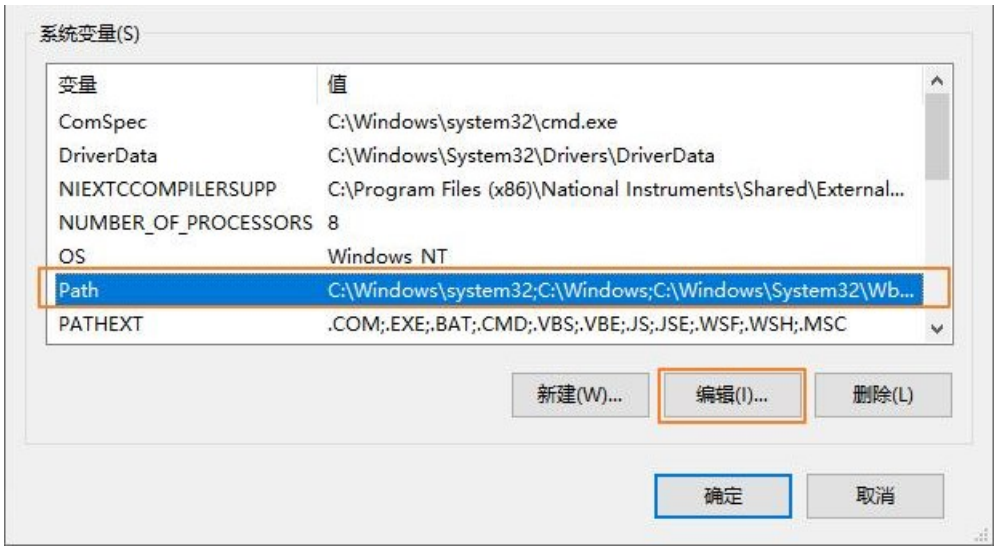


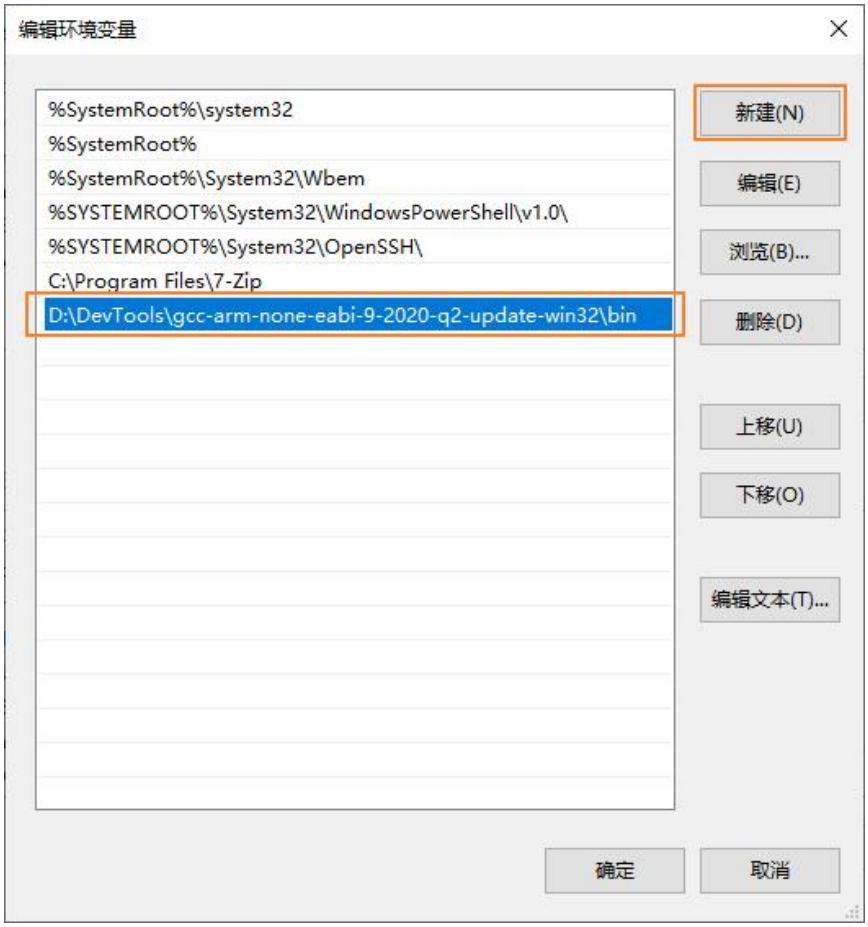
下载后解压 **gcc-arm-none-eabi-9-2020-q2-update-win32.zip** 到 D:\DevTools 中。




然后将路径 D:\DevTools\gcc-arm-none-eabi-9-2020-q2-update-win32\bin 添加到系统环境变量中



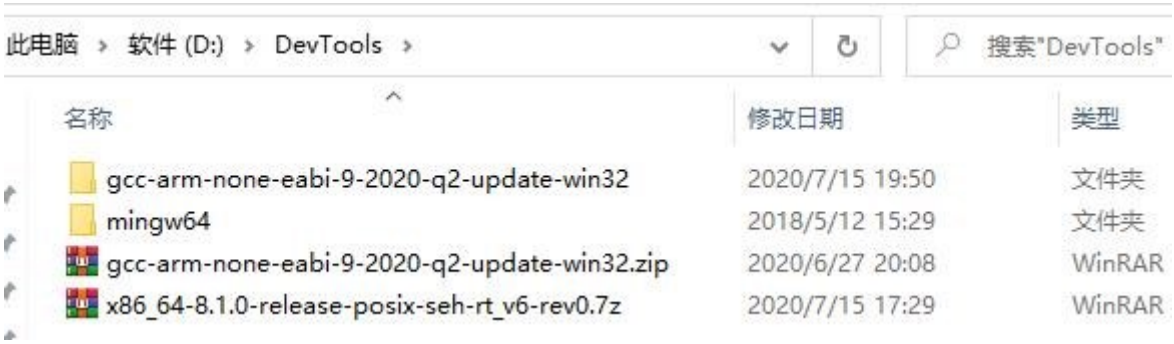


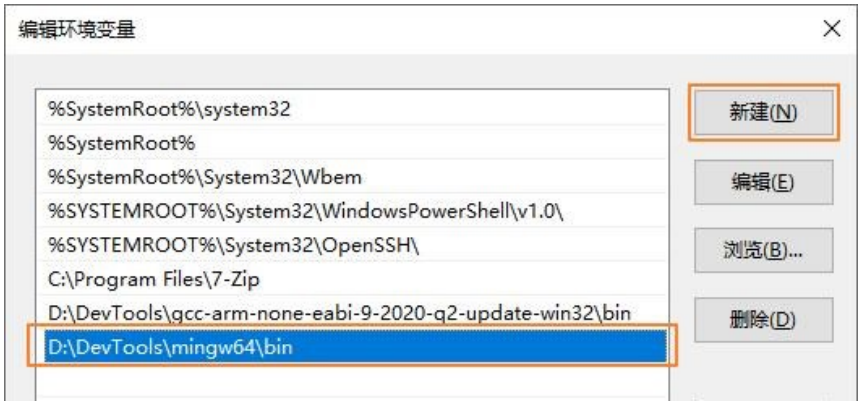

Step 05. 下载 **MinGW-W64** 离线压缩包：

*https://sourceforge.net/projects/mingw-w64/files/*


解压 MinGW-W64 离线压缩包 **x86_64-8.1.0-release-posix-seh-rt_v6-rev0.7z** 到 D:\DevTools 中



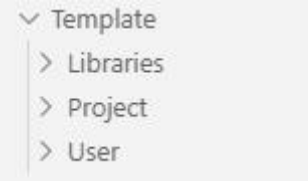

然后将路径 D:\DevTools\mingw64\bin 添加到系统环境变量中：



# 2 建立工程和编译

Step 01. 新建一个文件夹命名为 **Template** 的文件夹用以存放整个工程源码。

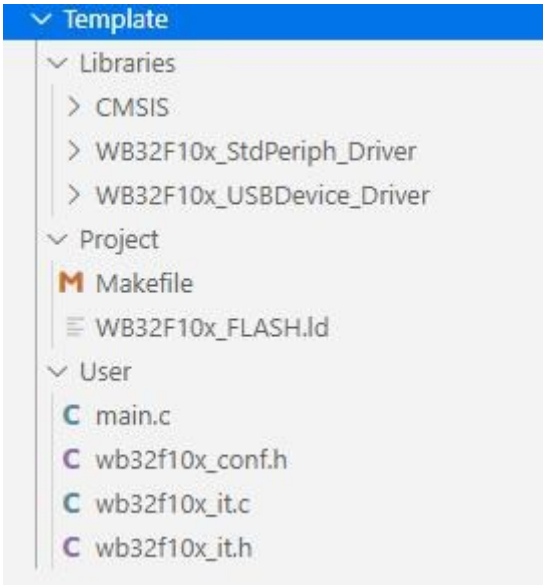
Step 02. 在 **Template** 文件夹中新建 **Libraries**，**Project** 和 **User** 三个子文件夹（当然用户可定义自己工程目录结构）




Step 03. 将 WB32F10x 标准固件库中 **Libraries** 目录中的内容复制到 Template\Libraries 目录中。

Step 04. 将 WB32F10x 标准固件库中 Project\WB32F10x_StdPeriph_Template 目录中 **main.c, wb32f10x_conf.h**, **wb32f10x_it.c**, **wb32f10x_it.h** 文件复制到 Template\User 目录中。

Step 05. 将 Project\WB32F10x_StdPeriph_Template\GCC 中的 **Makefile**, **WB32F10x_FLASH.ld** 复制到 Template\Project 目录中。



Step 06. 使用 VS Code 打开 Template 目录，对 Makefile 文件做如下修改

Makefile

Project > Makefile

```
34  ######################################
35  # source
36  ######################################
37  # C sources
38  C_SOURCES =  \
39  ../Libraries/CMSIS/Device/WB/WB32F10x/system_wb32f10x.c \
40  ../Libraries/WB32F10x_StdPeriph_Driver/src/misc.c \
41  ../Libraries/WB32F10x_StdPeriph_Driver/src/wb32f10x_adc.c \
42  ../Libraries/WB32F10x_StdPeriph_Driver/src/wb32f10x_anctl.c \
43  ../Libraries/WB32F10x_StdPeriph_Driver/src/wb32f10x_bkp.c \
44  ../Libraries/WB32F10x_StdPeriph_Driver/src/wb32f10x_crc.c \
45  ../Libraries/WB32F10x_StdPeriph_Driver/src/wb32f10x_dmac.c \
46  ../Libraries/WB32F10x_StdPeriph_Driver/src/wb32f10x_exti.c \
47  ../Libraries/WB32F10x_StdPeriph_Driver/src/wb32f10x_fmc.c \
48  ../Libraries/WB32F10x_StdPeriph_Driver/src/wb32f10x_gpio.c \
49  ../Libraries/WB32F10x_StdPeriph_Driver/src/wb32f10x_i2c.c \
50  ../Libraries/WB32F10x_StdPeriph_Driver/src/wb32f10x_i2s.c \
51  ../Libraries/WB32F10x_StdPeriph_Driver/src/wb32f10x_iwdg.c \
52  ../Libraries/WB32F10x_StdPeriph_Driver/src/wb32f10x_led.c \
53  ../Libraries/WB32F10x_StdPeriph_Driver/src/wb32f10x_pwr.c \
54  ../Libraries/WB32F10x_StdPeriph_Driver/src/wb32f10x_rcc.c \
55  ../Libraries/WB32F10x_StdPeriph_Driver/src/wb32f10x_rng.c \
56  ../Libraries/WB32F10x_StdPeriph_Driver/src/wb32f10x_rtc.c \
57  ../Libraries/WB32F10x_StdPeriph_Driver/src/wb32f10x_sfm.c \
58  ../Libraries/WB32F10x_StdPeriph_Driver/src/wb32f10x_spi.c \
59  ../Libraries/WB32F10x_StdPeriph_Driver/src/wb32f10x_tim.c \
60  ../Libraries/WB32F10x_StdPeriph_Driver/src/wb32f10x_uart.c \
61  ../Libraries/WB32F10x_StdPeriph_Driver/src/wb32f10x_wwdg.c \
62  ../User/main.c \
63  ../User/wb32f10x_it.c
64
65  # ASM sources
66  ASM_SOURCES =  \
67  ../Libraries/CMSIS/Device/WB/WB32F10x/startup/gcc/startup_wb32f10x.S
```

Makefile

Project > Makefile

```
109 # C defines
110 C_DEFS =  \
111 -DUSE_STDPERIPH_DRIVER \
112 -DMAINCLK_FREQ_96MHz
113
114
115 # AS includes
116 AS_INCLUDES =
117
118 # C includes
119 C_INCLUDES =  \
120 -I../Libraries/CMSIS/Include \
121 -I../Libraries/CMSIS/Device/WB/WB32F10x \
122 -I../Libraries/WB32F10x_StdPeriph_Driver/inc \
123 -I../User
124
```

Step 07.根据您使用的型号，在 WB32F10x_FLASH.ld 文件中配置 Flash 和 SRAM 大小，图中以 256KB Flash 和 36KBSRAM 举例（其他型号容量配置可参照下表）。

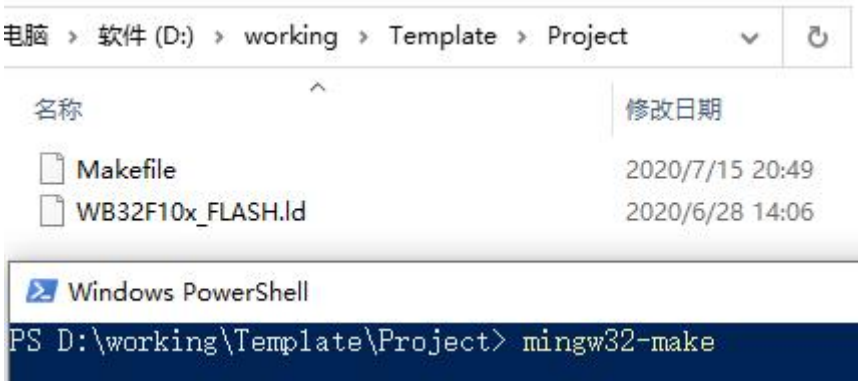
WB32F10x_FLASH.ld

Project > WB32F10x_FLASH.ld

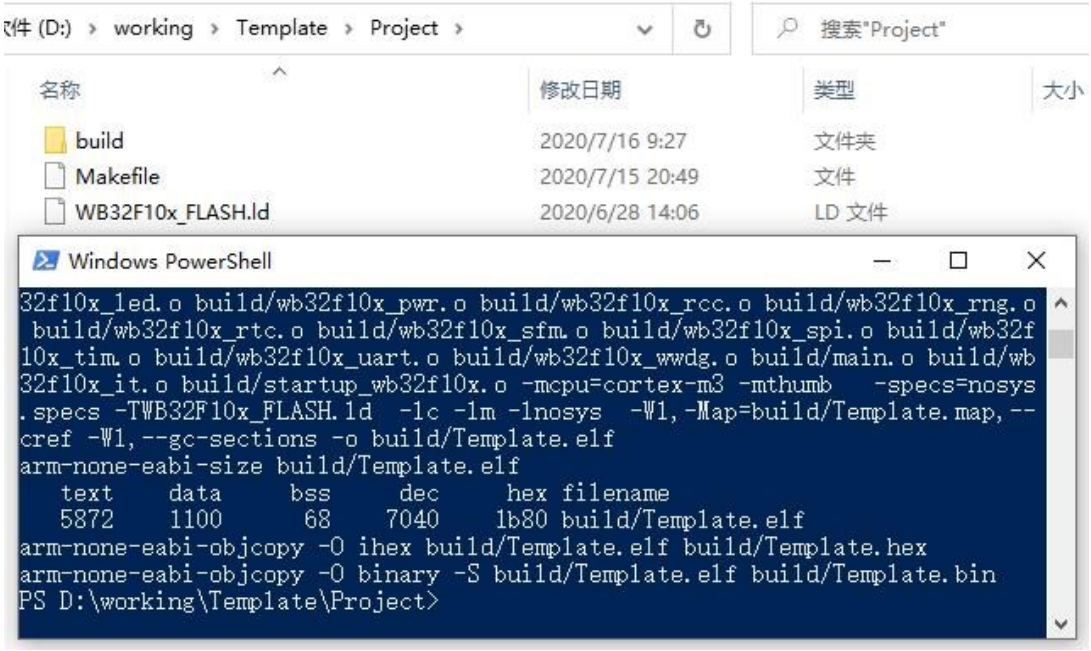
```
/*
 *-------- <<< Use Configuration Wizard in Context Menu >>> ------------------
 */

/*---------------------- Flash Configuration ----------------------------------
  <h> Flash Configuration
    <o0> Flash Base Address <0x0-0xFFFFFFFF:8>
    <o1> Flash Size (in Bytes) <0x0-0xFFFFFFFF:8>
  </h>
 ------------------------------------------------------------------------------*/
__ROM_BASE = 0x08000000;
__ROM_SIZE = 0x00040000;

/*--------------------- Embedded RAM Configuration ---------------------------
  <h> RAM Configuration
    <o0> RAM Base Address    <0x0-0xFFFFFFFF:8>
    <o1> RAM Size (in Bytes) <0x0-0xFFFFFFFF:8>
  </h>
 ------------------------------------------------------------------------------*/
__RAM_BASE = 0x20000000;
__RAM_SIZE = 0x00009000;

/*--------------------- Stack / Heap Configuration ---------------------------
  <h> Stack / Heap Configuration
    <o0> Stack Size (in Bytes) <0x0-0xFFFFFFFF:8>
    <o1> Heap Size (in Bytes) <0x0-0xFFFFFFFF:8>
  </h>
 ------------------------------------------------------------------------------*/
__STACK_SIZE = 0x00000400;
__HEAP_SIZE  = 0x00000C00;

/*
 *------------------- <<< end of configuration section >>> -------------------
 */
```

| 型号 | Flash 大小 | SRAM 大小 |
|---|---|---|
| WB32F10xx6 / WB32F11xx6 | 0x8000 (32KB) | 0x3000 (12KB) |
| WB32F10xx8 / WB32F11xx6 | 0x10000 (64KB) | 0x5000 (20KB) |
| WB32F10xx9 / WB32F11xx9 | 0x18000 (96KB) | 0x7000 (28KB) |
| WB32F10xxB / WB32F11xxB | 0x20000 (128KB) | 0x7000 (28KB) |
| WB32F10xxC / WB32F11xxC | 0x40000 (256KB) | 0x9000 (36KB) |

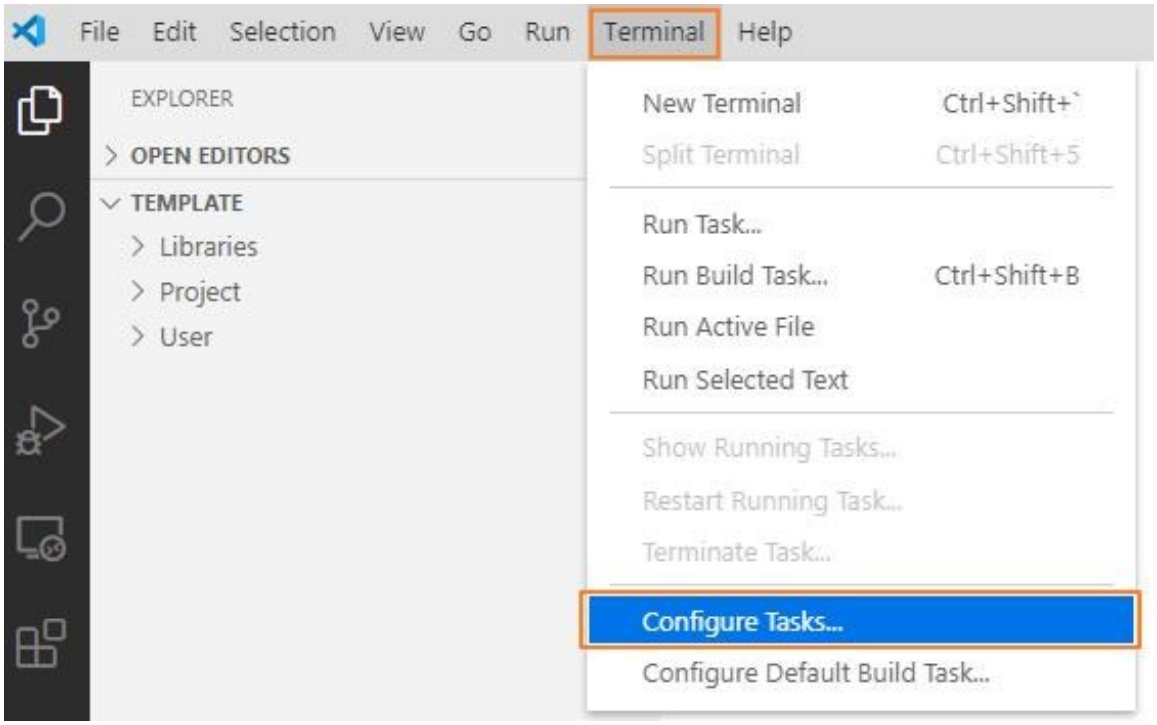
Step 08.在 Template\Project 目录中打开命令行，输入 **mingw32-make**，开始编译:

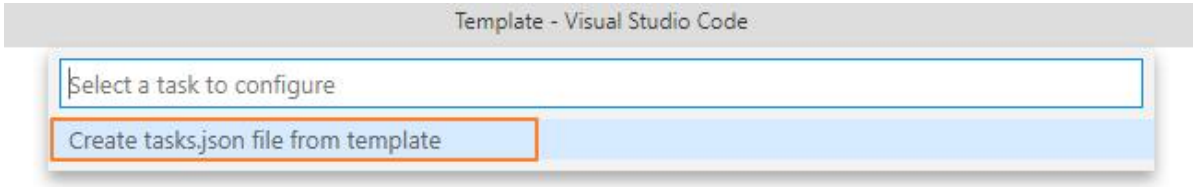


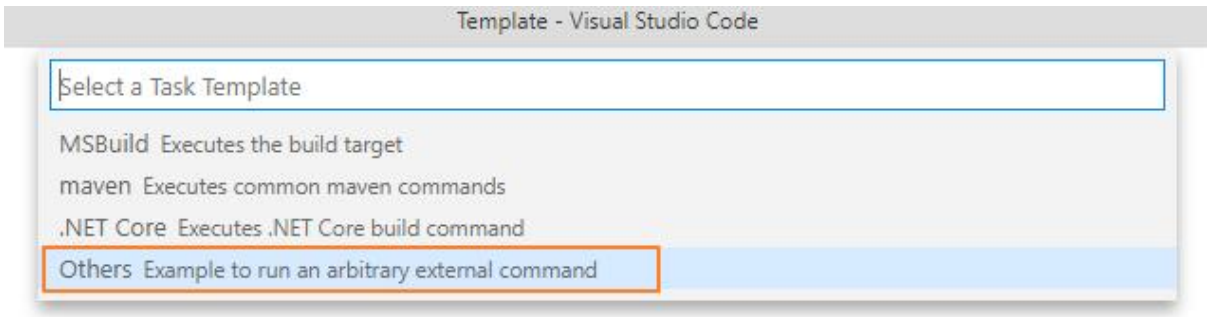

编译成功后，输出如下:



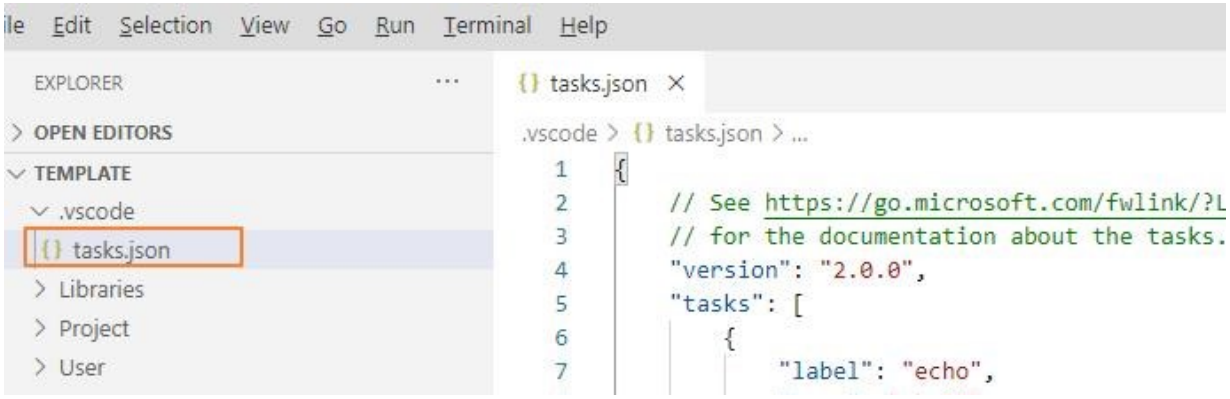
Step 09.在上一步我们已经能够成功编译 WB32F10x 的工程源码了，为了在 VS Code 中简化 编译这一步骤，我们需要在 VS Code 中创建一个任务。步骤如下所示:



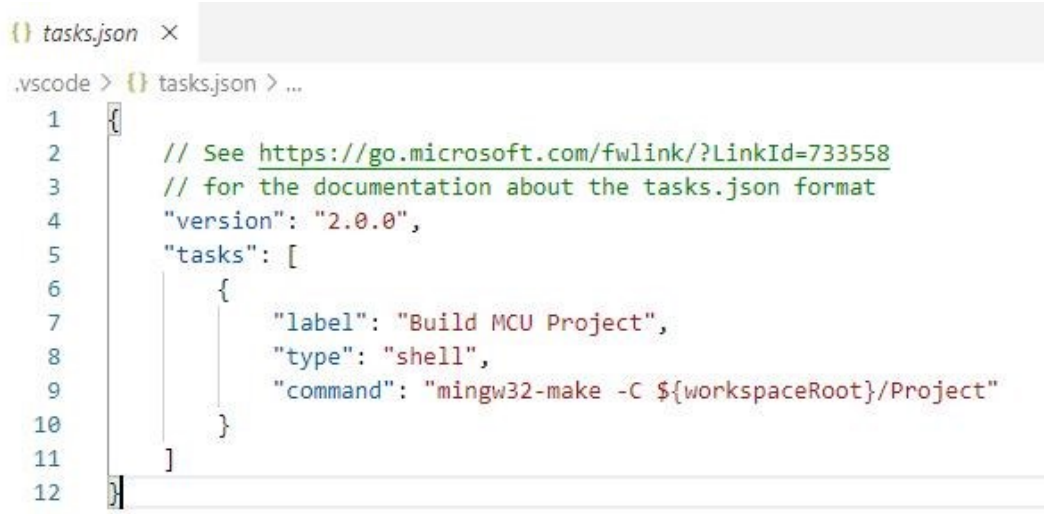



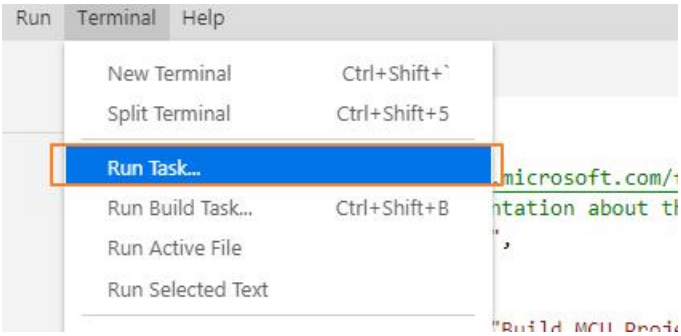
此时，在.vscode 目录中就出现了一个 tasks.json

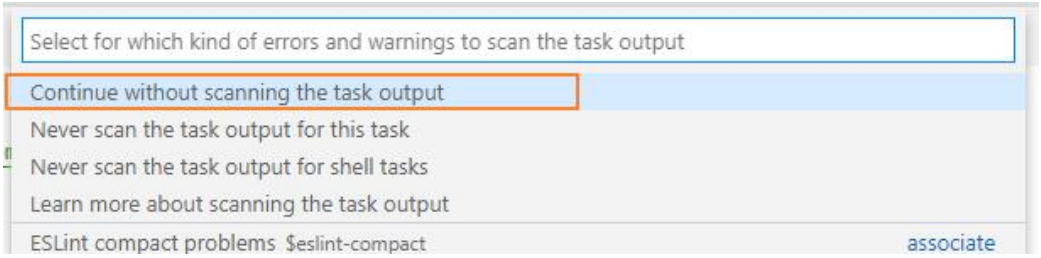



修改 tasks.json 为如下内容:

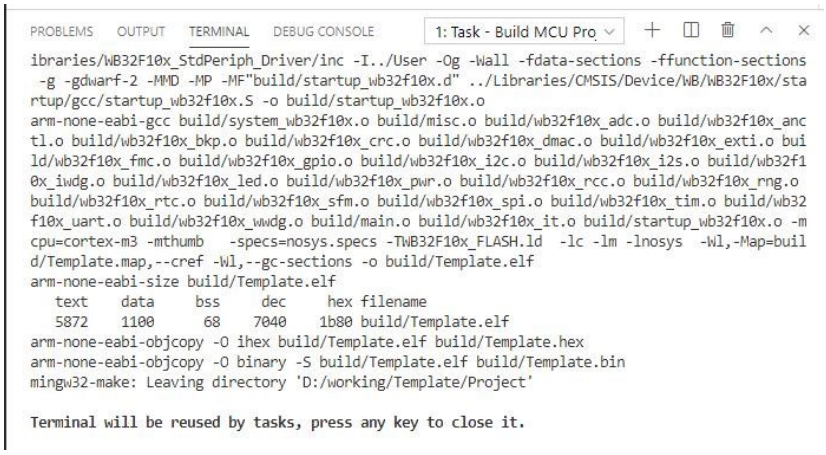


Step 10. 运行上一步在 VS Code 中创建的任务:




然后可以看到任务开始运行，并打印输出结果.



# 3 配置调试环境

Step 01.在 **J-Link** 安装目录 C:\Program Files (x86)\SEGGER\JLink_V614b\Devices 中新建一个名为 **WestBerryTech** 的文件夹；并在 WestBerryTech 文件夹中新建一个名为 **WB32F10x** 的文件夹。

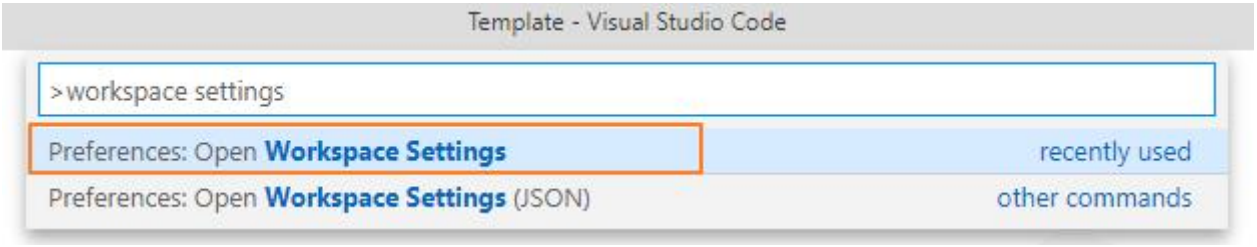
Step 02.将 WB32F10x 的烧录算法文件 **WB32F10x_256.FLM** 复制到

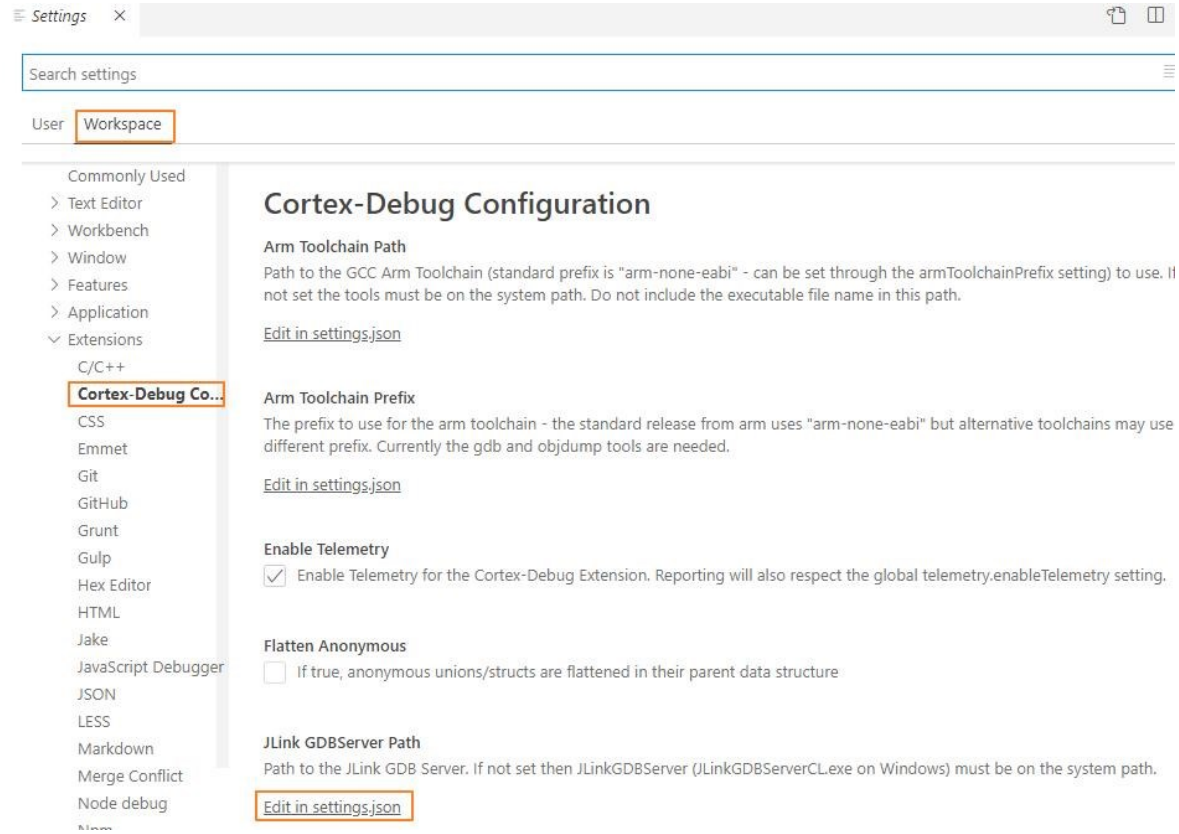
C:\Program Files(x86)\SEGGER\JLink_V614b\Devices\WestBerryTech\WB32F10x 文件夹中

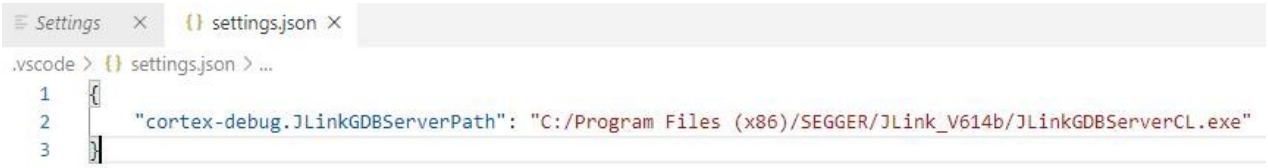

Step 03.在 C:\Program Files (x86)\SEGGER\JLink_V614b\JLinkDevices.xml 文件中添加 WB32F10x 的信息并保存。

```
<Device>
  <ChipInfo Vendor="WestBerryTech" Name="WB32F10x" Core="JLINK_CORE_CORTEX_M3"
  WorkRAMAddr="0x20000000" WorkRAMSize="0x1000" />
  <FlashBankInfo Name="Internal Flash" BaseAddr="0x08000000" MaxSize="0x40000"
  Loader="Devices\WestBerryTech\WB32F10x\WB32F10x_256.FLM"
  LoaderType="FLASH_ALGO_TYPE_CMSIS" AlwaysPresent="1" />
</Device>
```

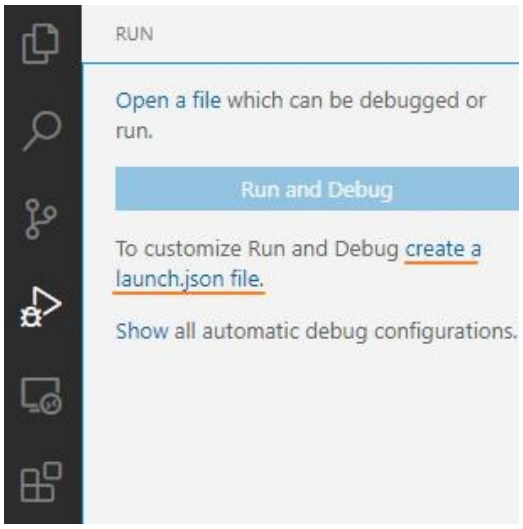

Step 04.在 VS Code 中配置 JLink GDBServer 的路径

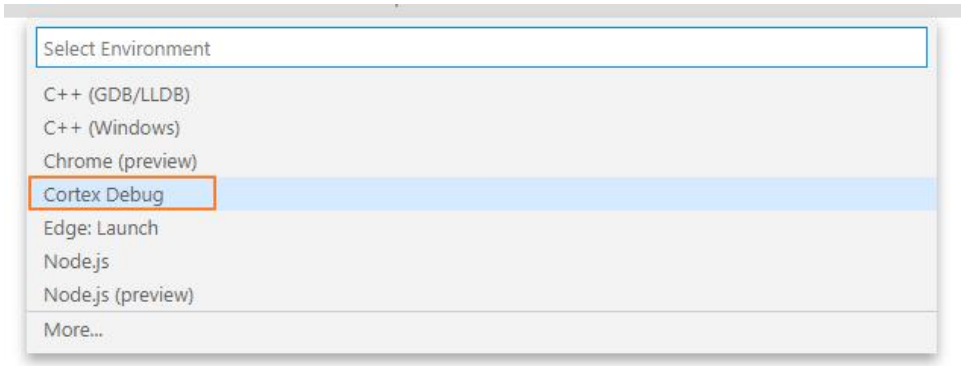



Settings
Search settings
User
Workspace
Commonly Used
Text Editor
Workbench
Window
Features
Application
Extensions
C/C++
Cortex-Debug Co...
CSS
Emmet
Git
GitHub
Grunt
Gulp
Hex Editor
HTML
Jake
JavaScript Debugger
JSON
LESS
Markdown
Merge Conflict
Node debug
Cortex-Debug Configuration
Arm Toolchain Path
Path to the GCC Arm Toolchain (standard prefix is "arm-none-eabi" - can be set through the armToolchainPrefix setting) to use. If not set the tools must be on the system path. Do not include the executable file name in this path.
Edit in settings.json
Arm Toolchain Prefix
The prefix to use for the arm toolchain - the standard release from arm uses "arm-none-eabi" but alternative toolchains may use different prefix. Currently the gdb and objdump tools are needed.
Edit in settings.json
Enable Telemetry
Enable Telemetry for the Cortex-Debug Extension. Reporting will also respect the global telemetry.enableTelemetry setting.
Flatten Anonymous
If true, anonymous unions/structs are flattened in their parent data structure
JLink GDBServer Path
Path to the JLink GDB Server. If not set then JLinkGDBServer (JLinkGDBServerCL.exe on Windows) must be on the system path.
Edit in settings.json


Settings
settings.json
.vscode > settings.json > ...
1 {
2 "cortex-debug.JLinkGDBServerPath": "C:/Program Files (x86)/SEGGER/JLink_V614b/JLinkGDBServerCL.exe"
3 }

Step 05. 创建调试启动配置文件



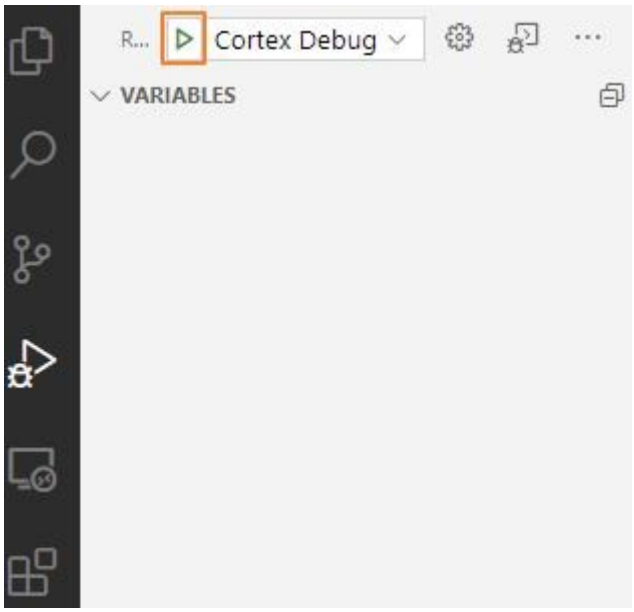

修改如下：

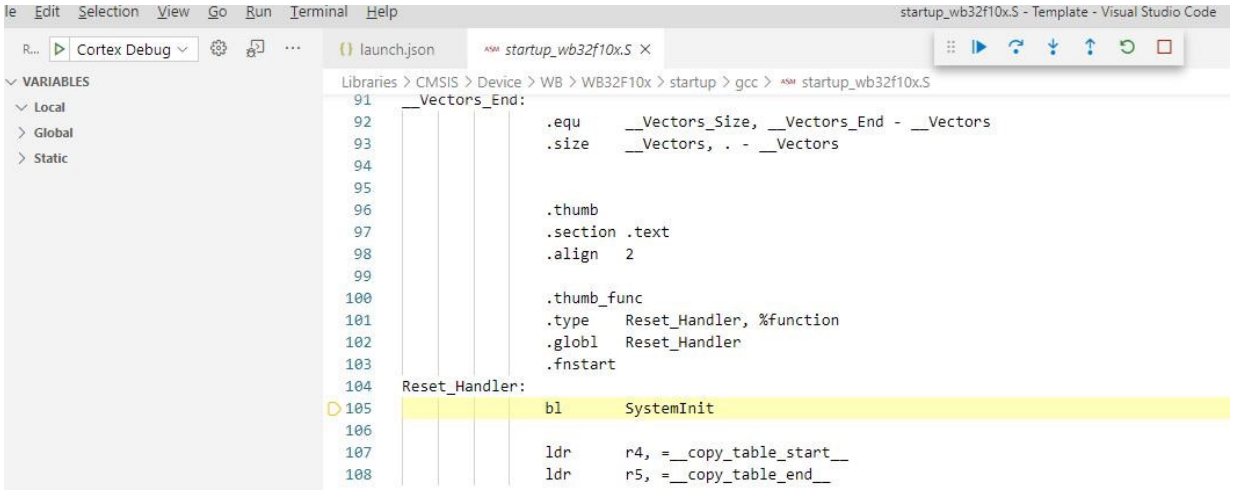
launch.json

.vscode > launch.json > ...

```
{
    // Use IntelliSense to learn about possible attributes.
    // Hover to view descriptions of existing attributes.
    // For more information, visit: https://go.microsoft.co
    "version": "0.2.0",
    "configurations": [
        {
            "name": "Cortex Debug",
            "cwd": "${workspaceRoot}",
            "executable": "./Project/build/Template.elf",
            "request": "launch",
            "type": "cortex-debug",
            "servertype": "jlink",
            "device": "WB32F10x",
            "interface": "swd"
        }
    ]
}
```

Step 06. 使用 J-Link 下载调试工具通过 SWD 连接到 WB 芯片，并再 VS Code 中开始调试



# 版本历史

| Revision | Date | Description |
|---|---|---|
| 1.2 | 2020/09/05 | Initial Release |

# 免责声明

本文档中的信息仅针对 WB 产品提供。本文档，包括本文档中描述的任何 WB 产品（“产品”），根据中华人民共和国和全球其他司法管辖区的知识产权法律和条约，归属 WB 所有。常州韦斯佰瑞电子科技有限公司及其子公司（“WB”）保留随时对本文档以及文档中所描述的产品与服务进行更改、更正、修改或改进的权利，恕不另行通知。WB 不承担任何因应用程序或使用本文档中描述的任何产品引起的任何责任。

购买者应对 WB 产品与服务的选择、选型和使用承担全部责任，并且 WB 不承担对 WB 产品与服务的选择、选型和使用的任何责任。

本文档未通过禁反言或其他方式对任何知识产权授予任何明示或暗示的许可。如果本文档的任何部分提及任何第三方产品或服务，则不应视为 WB 授予使用此类第三方产品或服务或其中包含的任何知识产权的许可，或视为涵盖在此类第三方产品或服务或其中包含的任何知识产权的任何方式。

除适用协议中明确规定的定制产品外，产品仅为普通商业、工业、个人或家庭应用而设计、开发或制造。产品并非设计、意图或授权用作设计或用于操作武器、武器系统、核装置、原子能控制仪器、燃烧控制仪器、飞机或宇宙飞船仪器、运输仪器、交通信号系统中仪器、生命支持设备或系统、其他医疗设备或系统（包括复苏设备和外科植入物）、污染控制或有害物管理、由于设备或仪器的故障可能导致人身伤害、死亡、财产损失或环境破坏的其他用途。

转授 WB 产品的条款与本文档中规定的声明和/或技术特征不同的，将立即使 WB 对此处描述的 WB 产品或服务的任何保证失效，并且不得以任何方式产生或扩展 WB 的任何责任。