WB32F10x
Set Up A GCC Development Environment (Through VS Code And J-Link)

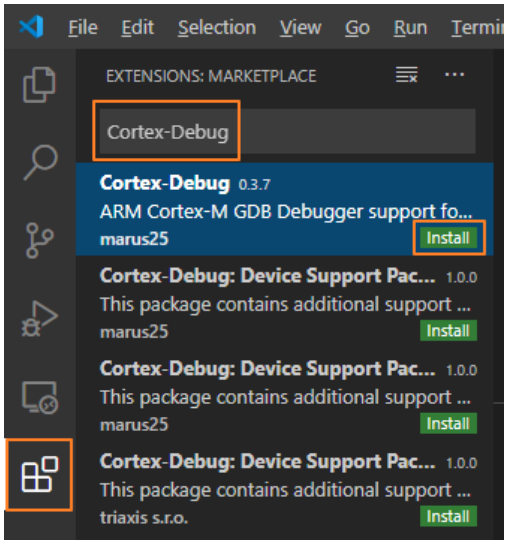
Westberry Technology (ChangZhou) Corp., ltd

Contents

1 Install the development environment

Step 01. Install Visual Studio Code (https://code.visualstudio.com/).

Step 02. Install the **Cortex-Debug** extension in Visual Studio Code.

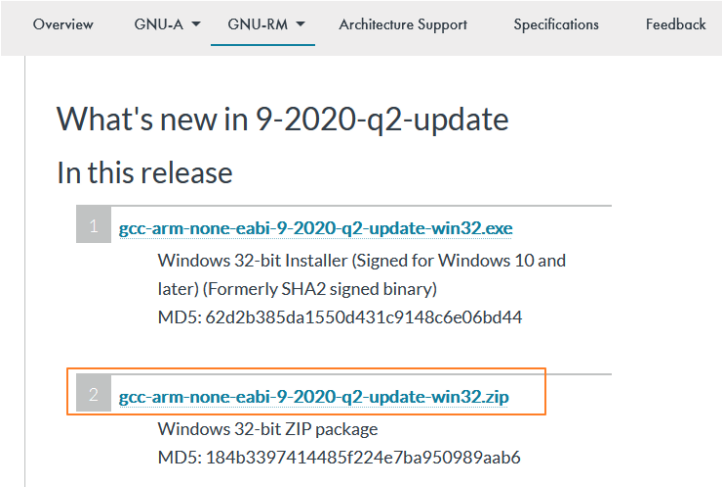


Step 03. Prepare a J-Link emulator and install the J-Link emulator driver:
(https://www.segger.com/downloads/jlink)

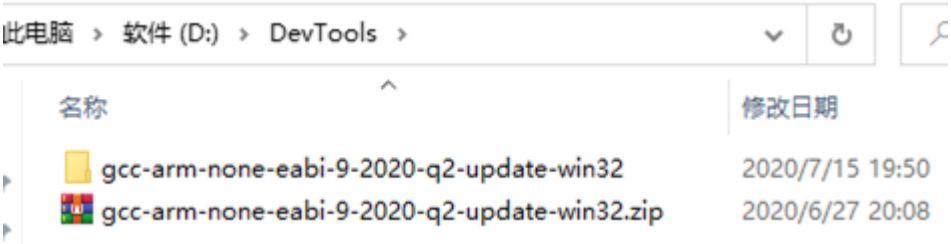
Step 04. Create a new folder to store the tools needed for development. The path to the new folder in this tutorial is D:\DevTools.

Step 05. Download the **GCC-ARM compilation toolchain** tarball and place it in the D:\DevTools folder:

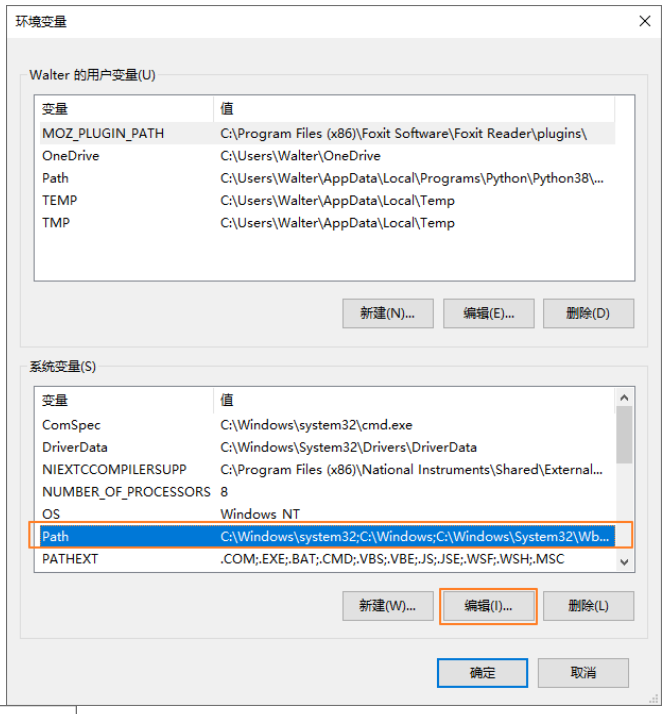
(https://developer.arm.com/tools-and-software/open-source-software/developer-tools/gnu-toolchain/gnu-rm/downloads)

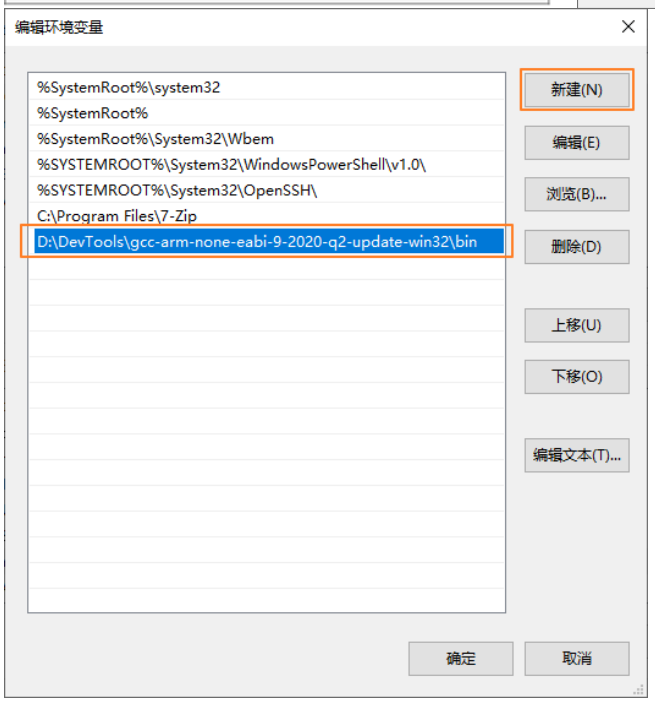


Step 06. Unzip the **gcc-arm-none-eabi-9-2020-q2-update-win32.zip**



Then add the path D:\DevTools\gcc-arm-none-eabi-9-2020-q2-update-win32\bin to the **System Environment Variables**

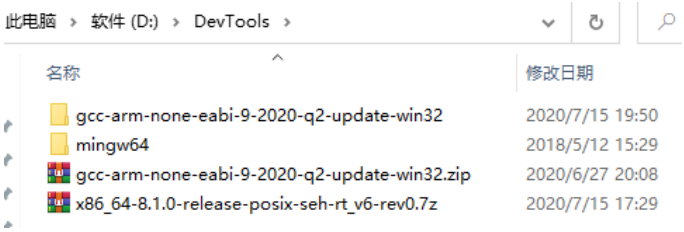




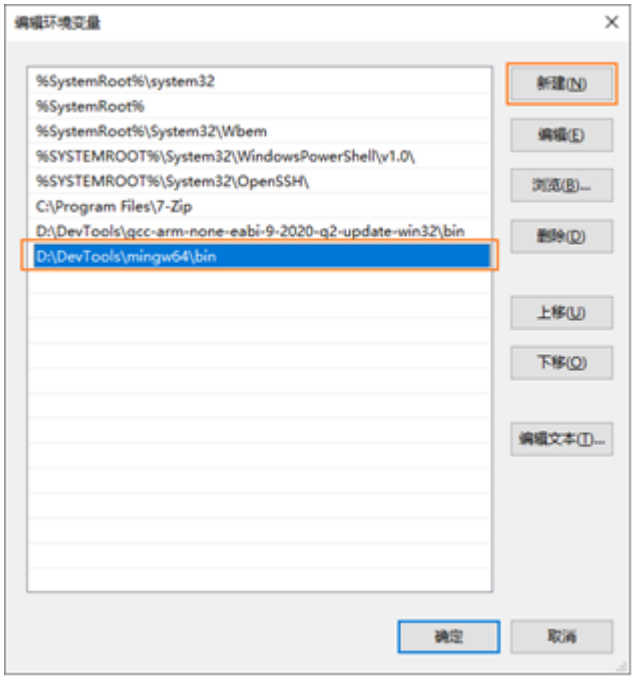
Step 07. Download the MinGW-W64 offline installer and place it in the D:\DevTools folder:

(https://sourceforge.net/projects/mingw-w64/files/)

Step 08. Unzip the **MinGW-W64 offline installer** (x86_64-8.1.0-release-posix-seh-rt_v6-rev0.7z)



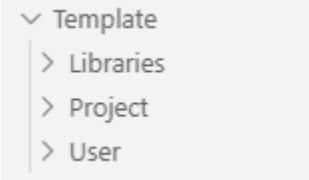
Then add the path D:\DevTools\mingw64\bin to the **System Environment Variables**



2 Create the project and compile

Step 01. Create a new folder named Template to store all the files of the project.

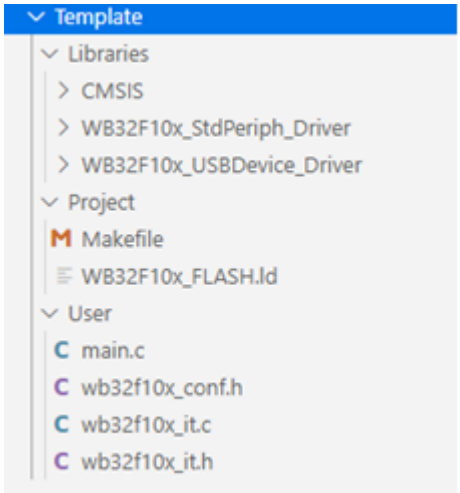
Step 02. Create **Libraries, Project** and **User** subfolders in the Template folder(You can also make the project directory structure according to your own habits.).



Step 03. Copy the contents of **Libraries** floder (in the **WB32F10x_StdPeriph_Lib** floder) to the Template\Libraries folder

Step 04. Copy the contents of Project\ WB32F10X_STDPeripher_Template folder from the **WB32F10x_StdPeriph_Lib** floder to the Template\User folder.

Step 05. Copy **Makefile** file and **WB32F10x_FLASH.ld** file in Project\WB32F10x_StdPeriph_Template\GCC folder to Template\Project folder.



Step 06. Use VS Code to open the **Template** folder and make the following modifications to the **Makefile** file.

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
```

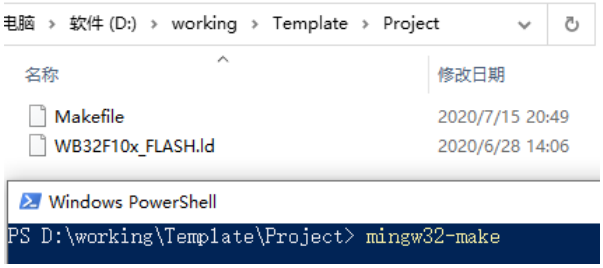
Step 07. According to the product code you use, configure the Flash and SRAM size in the WB32F10x_FLASH.ld file. The figure takes 256KB Flash and 36KB SRAM as an example (for the product capacity configuration of other codes, please refer to the table below).

```
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 >>> ------------------
 */
```

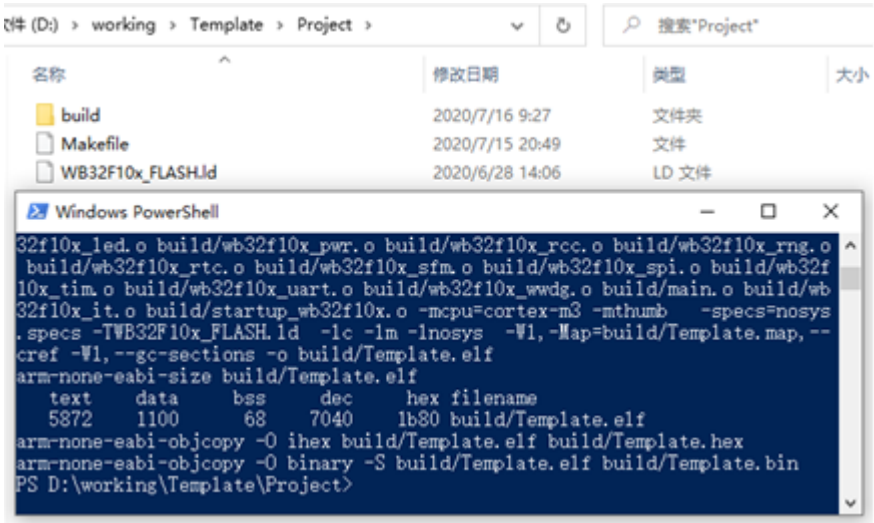
Product Code	Flash Size	SRAM Size
WB32F10xx6	0x8000 (32KB)	0x3000 (12KB)
WB32F10xx8	0x10000 (64KB)	0x5000 (20KB)
WB32F10xx9	0x18000 (96KB)	0x7000 (28KB)
WB32F10xxB	0x20000 (128KB)	0x7000 (28KB)
WB32F10xxC	0x40000 (256KB)	0x9000 (36KB)

Step 08. Open the **Windows PoweShell** (command line) in the Template\Project directory.

Step 09. Enter mingw32-make and start compiling



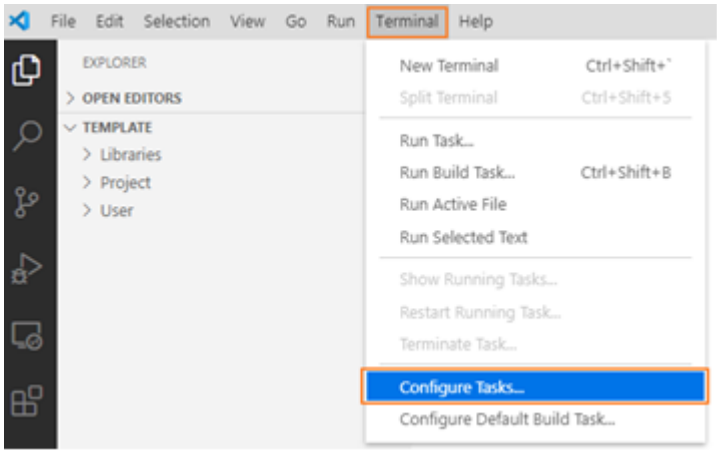
After the compilation is successful, the output is as follows:

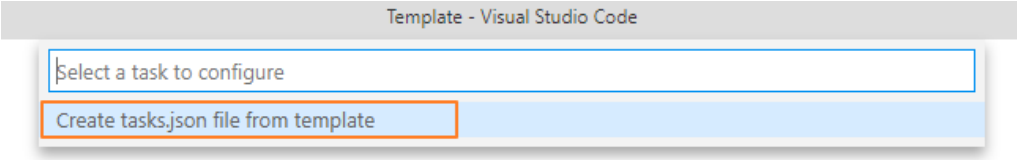


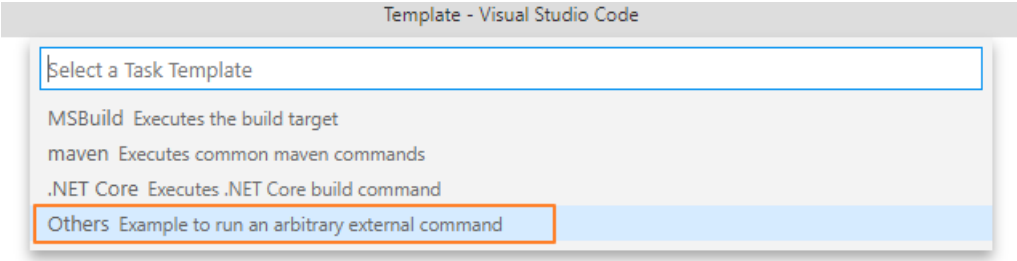
3 Simplify Compilation Step

In the previous step, we have been able to successfully compile the project source code of WB32F10x. In order to simplify the compilation step in VS Code, we need to create a task in VS Code. The steps are as follows:

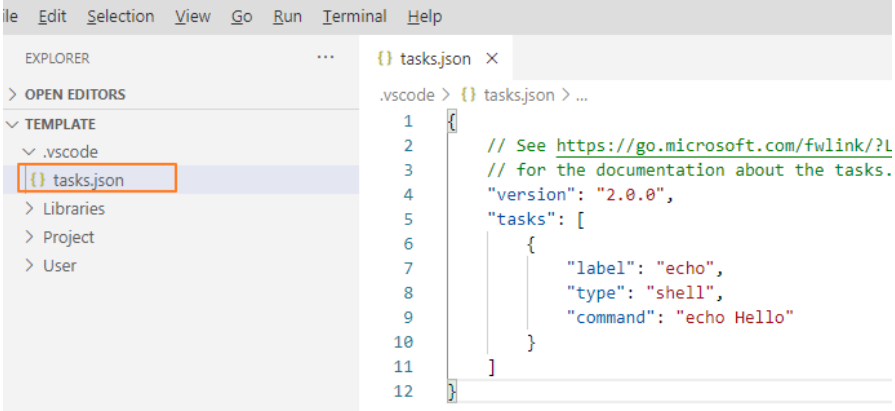
Step 01, As shown below:



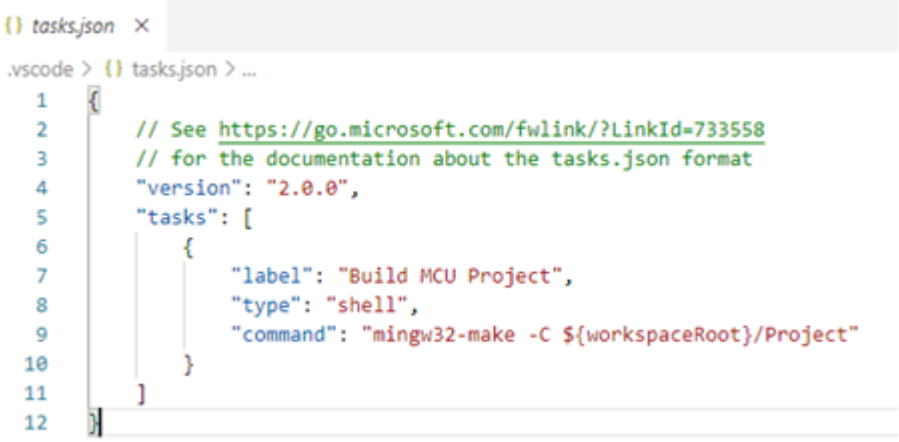




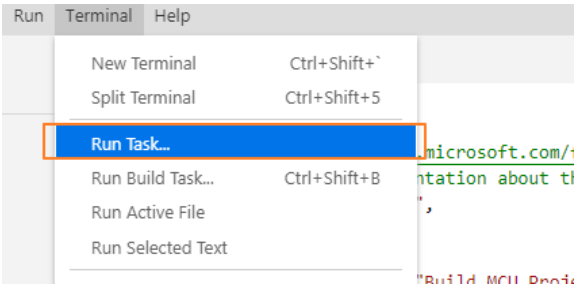
At this point, a **tasks.json** appears in the .VS Code directory.

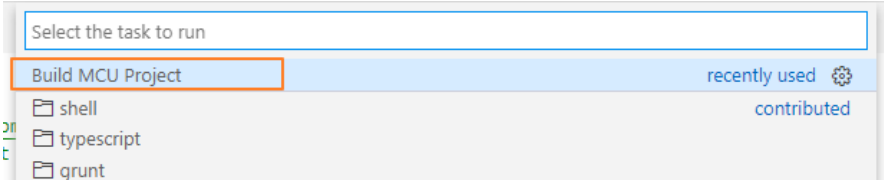


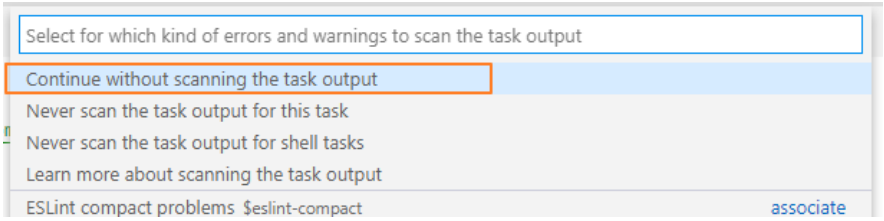
Step 02, Modify **tasks.json** to the following content



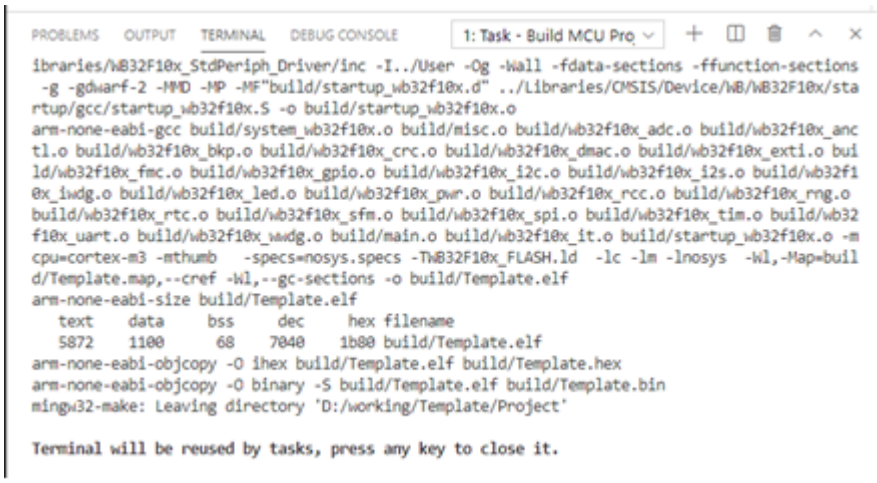
Step 03, Run the task created in VS Code in the previous step







Then you can see the task start running and print the output:



4 Configure the debug environment

Step 01, Create a new folder named **WestBerryTech** in the J-Link installation directory:

C:\Program Files (x86)\SEGGER\JLink_V614b\Devices

And then create a new folder named **WB32F10x** in the **WestBerryTech** folder.

Step 02, Copy the **WB32F10x_256.FLM** (programming algorithm file) to

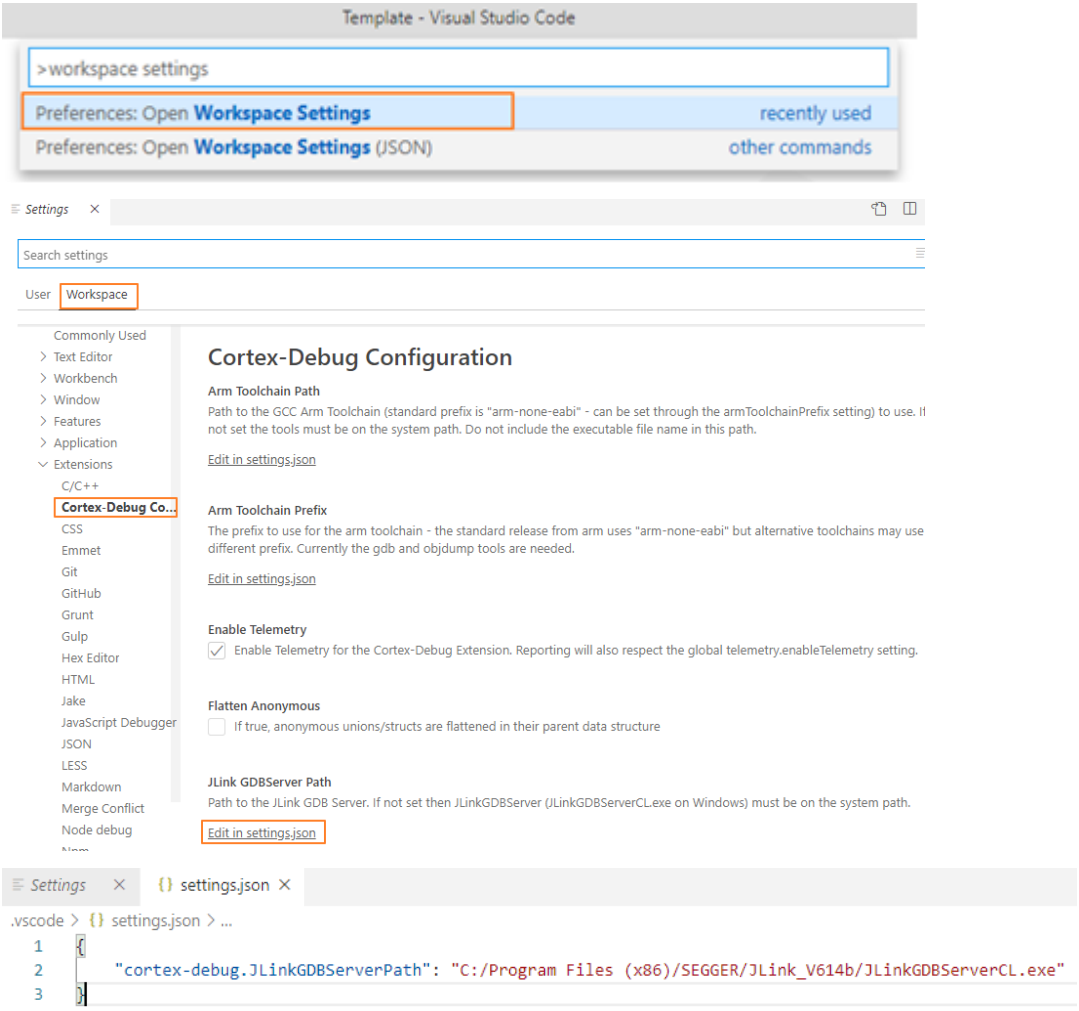
C:\Program Files (x86)\SEGGER\JLink_V614b\Devices\WestBerryTech\WB32F10x folder

Step 03, Add WB32F10x information to **JLinkDevices.xml** file and save it.

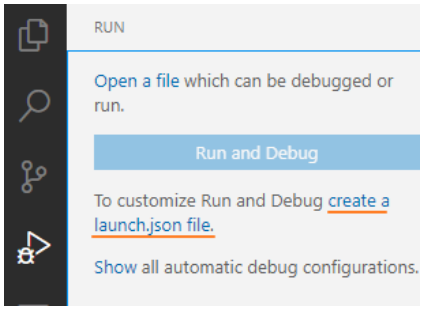
(**JLinkDevices.xml** file is in C:\ProgramFiles(x86)\SEGGER\JLink_V614b\ path)

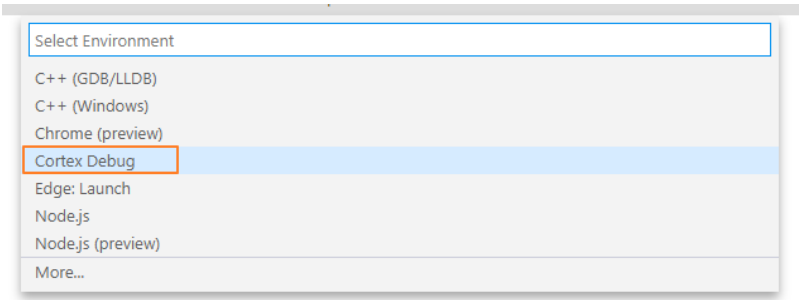
```
<Device>
    <ChipInfo Vendor="WB" Name="WB32F10x" Core="JLINK_CORE_CORTEX_M3" WorkRAMAddr="0x20000000" WorkRAMSize="0x9000" />
    <FlashBankInfo Name="Internal Flash" BaseAddr="0x08000000" MaxSize="0x40000" Loader="Devices\WestBerry\WB32F10x\WB32F10x_256.FLM"
   LoaderType="FLASH_ALGO_TYPE_CMSIS" AlwaysPresent="1" />
</Device>
```

Step 04, Configure the path to JLink GDBServer in VS Code



Step 05, Create a file to customize **Run** and **Debug**



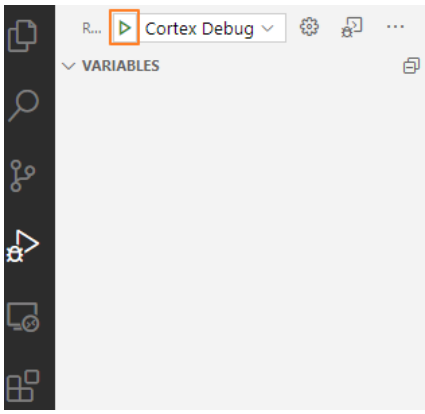


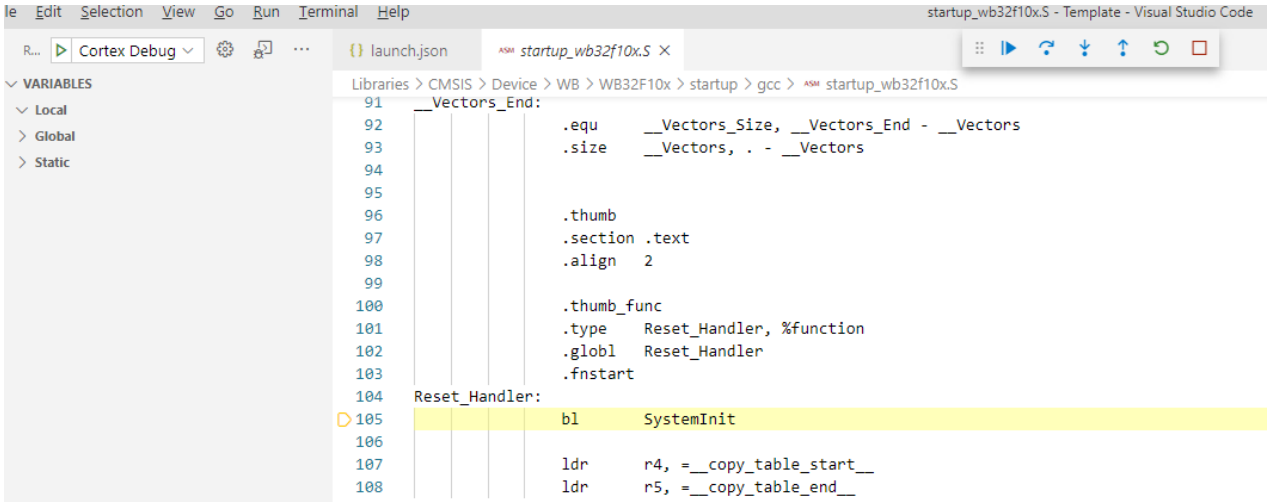
Modify as follows:

```
{} launch.json ×
.vscode > {} launch.json > ...
1  {
2      // Use IntelliSense to learn about possible attributes.
3      // Hover to view descriptions of existing attributes.
4      // For more information, visit: https://go.microsoft.co
5      "version": "0.2.0",
6      "configurations": [
7          {
8              "name": "Cortex Debug",
9              "cwd": "${workspaceRoot}",
10             "executable": "./Project/build/Template.elf",
11             "request": "launch",
12             "type": "cortex-debug",
13             "servertype": "jlink",
14             "device": "WB32F10x",
15             "interface": "swd"
16         }
17     ]
18 }
```

Step 06, Use a J-Link emulator to connect to the WB32F10x chip via the SWD interface.

Step 07, Start debugging





Revision History

Revision	Date	Description
1.2	2022/7/5	Initial Release

IMPORTANT NOTICE

Information in this document is provided solely in connection with WB products. This document, including any product of WB described in this document (the "Product"), is owned by WB under the intellectualproperty laws and treaties of the People's Republic of China and other jurisdictions worldwide. Westberry Technology (ChangZhou) Corp., ltd and its subsidiaries("WB") reserve the right to make changes, corrections, modifications or improvements, to this document, and the products and services described herein at any time, without notice.WB does not assume any liability arising out of the application or use of any Product described in this document. Purchasers are solely responsible for the choice, selection and use of the WB products and services described herein, and WB assumes no liability whatsoever relating to the choice, selection or use of the WB products and services described herein.

No license, express or implied, by estoppel or otherwise, to any intellectual property rights is granted under this document. If any part of this document refers to any third party products or services it shall not be deemed a license grant by WB for the use of such third party products or services, or any intellectual property contained therein or considered as a warranty covering the use in any manner whatsoever of such third party products or services or any intellectual property contained therein.

Except for customized products which has been expressly identified in the applicable agreement, the Products are designed, developed or manufactured for ordinary business, industrial, personal, or household applications only. The Products are not designed, intended, or authorized for use as components in systems designed or intended for the operation of weapons, weapons systems, nuclear installations, atomic energy control instruments, combustion control instruments, airplane or spaceship instruments, transportation instruments, traffic signal instruments, life-support devices or systems, other medical devices or systems (including resuscitation equipment and surgical implants), pollution control or hazardous substances management, or other uses where the failure of the device or Product could cause personal injury, death, property or environmental damage.

Resale of WB products with provisions different from the statements and/or technical features set forth in this document shall immediately void any warranty granted by WB for the WB product or service described herein and shall not create or extend in any manner whatsoever, anyliability of WB.