

WT32L064/032 Application Note

VSCODE 发展平台

(简体版)

Rev. 0.3
March 2021

目 录

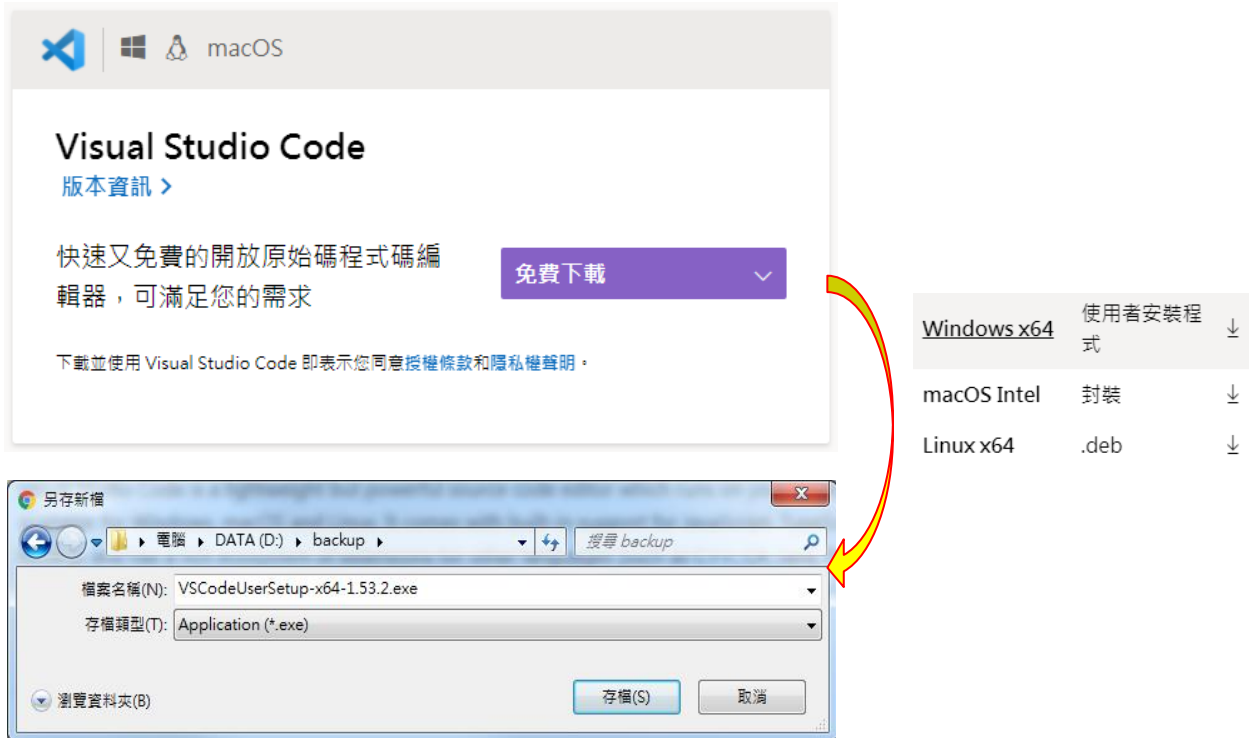
1. VSCODE 安装与环境设定	4
(STEP 1) 下载 VSCODE	4
(STEP 2) 下载 ARM-MDK.....	5
(STEP 3) 安装 SEGGER/J-LINK	7
(STEP 4) 安装 GCC COMPILER	8
(STEP 5) 安装 WINDOWS-BUILD-TOOL	9
(STEP 6) 安装 WT32FLASH	10
2. VSCODE 操作流程说明	11
2.1 代码产生器软件安装.....	11
2.2 代码产生器软件说明.....	11
3. VSCODE 平台环境说明	12
3.1 主选单.....	13
3.1.1 开启项目操作范例:.....	13
3.1.2 建立新 C 档案操作范例:	13
3.1.3 进行 C 程序编译操作范例:	14
3.1.4 开启 Terminal 操作刻录范例:.....	14
3.2 控制选单	15
3.2.1 开启项目文件列表范例:.....	15
3.2.2 开启侦错(Debug)操作范例:.....	16
3.2.3 开启扩充组件操作范例:.....	16
3.3 项目文件夹内容	17
4. VSCODE 程序开发与除错	18
4.1 编译功能说明	18
4.2 文法错误排除	18
4.3 刻录功能说明	19
4.4 逻辑除错功能	20

5. 附录	22
5.1 TASKS.JSON 功能说明	22
5.2 LAUNCH.JSON 功能说明	22
5.3 MAKEFILE 功能说明	23
5.4 LINKER 功能说明	23
6. 版本更改纪录:	24

1. VSCODE 安装与环境设定

(Step 1) 下载 VSCODE

下载网址如右 <https://visualstudio.microsoft.com/zh-hant/downloads/>

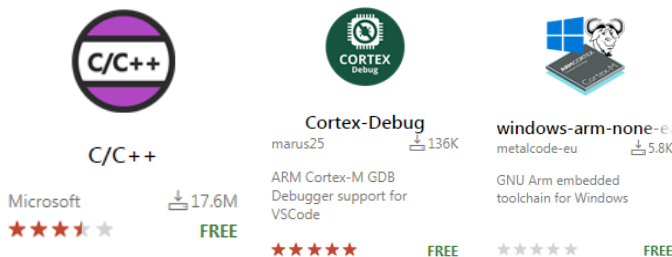


安装完后出现下列 ICON

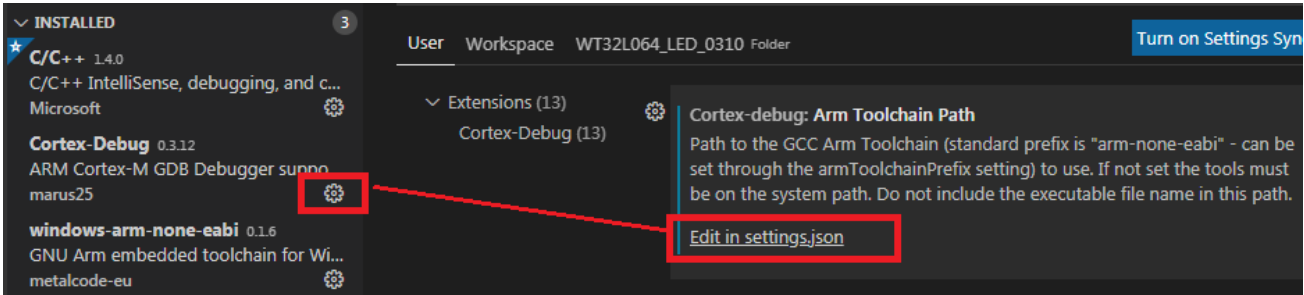


于网址 <https://marketplace.visualstudio.com/vscode> 安装下列 Extensions 模块。

1. C/C++
2. Cortex Debug
3. Windows-arm-none-eabi



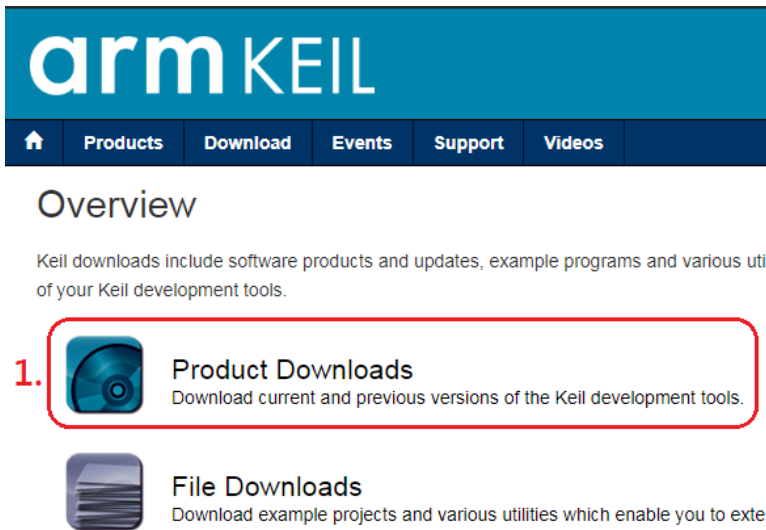
开启 VSCODE 软件左下设定 Extensions-> Settings, 新增 Cortex-Debug 需要的路径
"Cortex-debug.armToolchainPath"



```
{  
  "arm-none-eabi.bin": "${env:USERPROFILE}/.vscode/...../bin",  
  "cortex-debug.armToolchainPath": "C:/ARM/gcc/bin"  
}
```

(Step 2) 下载 ARM-MDK

下载网址如右 <https://www.keil.com/download/>



<https://www.keil.com/download/product/>

Download Products

Select a product from the list below to download the latest version.

2.  **MDK-Arm**
Version 5.29 (November 2019)
Development environment for Cortex and Arm devices.

 **C51**
Version 9.60a (May 2019)
Development tools for all 8051 devices.

 **C251**
Version 5.60 (May 2018)
Development tools for all 80251 devices.

 **C166**
Version 7.57 (May 2018)
Development tools for C166, XC166, & XC2000 MCUs.

Home / Product Downloads

MDK-ARM

MDK-ARM Version 5.29
Version 5.29

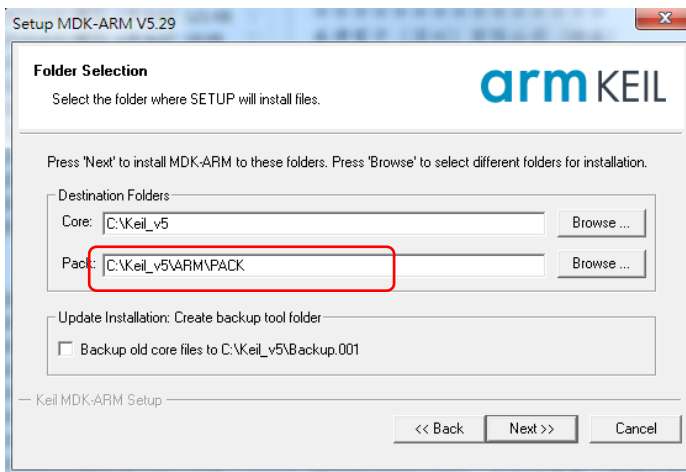
- Review the [hardware requirements](#) before installing this software.
- Note the [limitations of the evaluation tools](#).
- [Further installation instructions for MDK5](#)

(MD5:0D0654419D24A7C2BAE6C4858504B350)

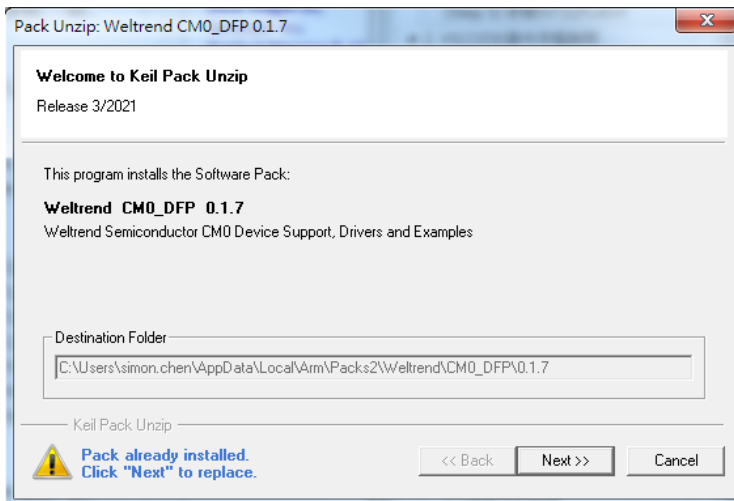
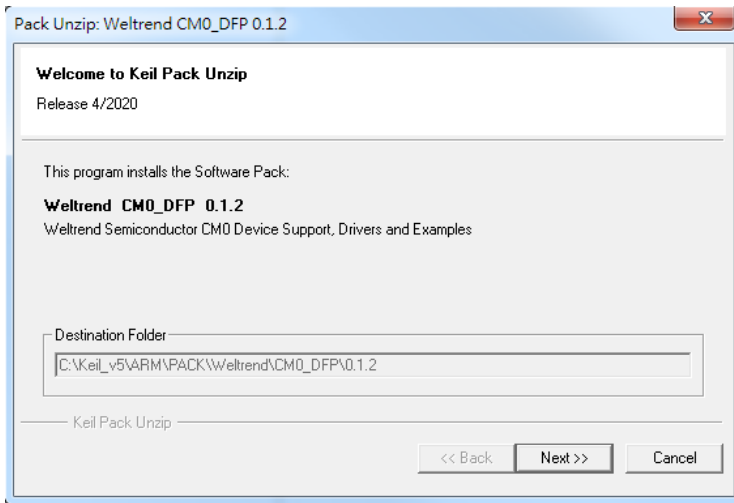
To install the MDK-ARM Software...

- Right-click on **MDK529.EXE** and save it to your computer.
- PDF files may be opened with Acrobat Reader.
- ZIP files may be opened with PKZIP or WINZIP.

3. **MDK529.EXE** (855,164K)
Monday, November 18, 2019



下载并安装 MDK 后，请于 PC 端再安装伟詮 PACK 档案 *Weltrend.CM0_DFP.0.1.x.pack*
官网下载路径 <http://www.weltrend.com.tw/zh-tw/support/detail/2/105/105>



(Step 3) 安装 SEGGER/J-Link

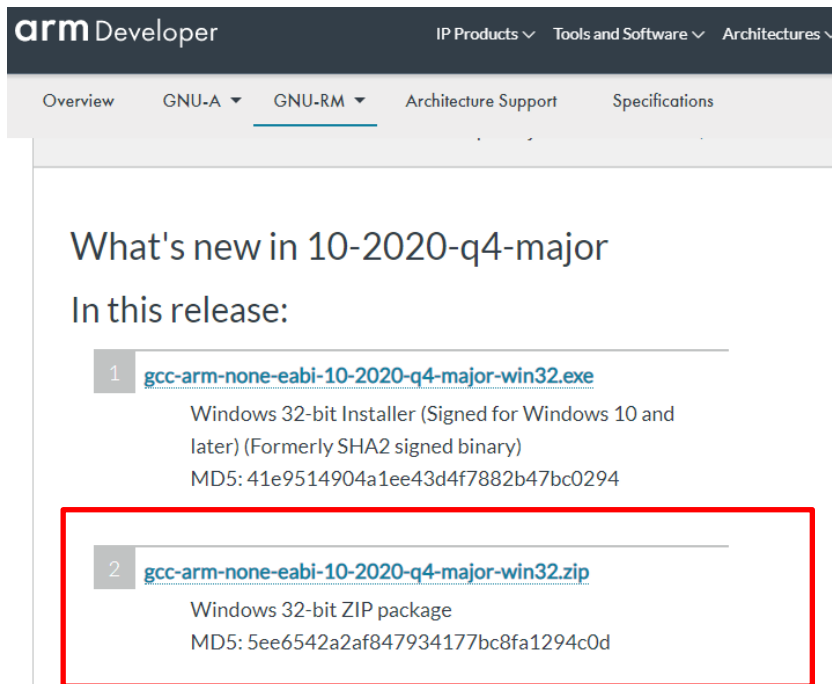
下载 SEGGER 安装如下列网址, 选用 V5.12 版本默认安装路径 C:/Program Files (x86) /SEGGER/JLink_V512
<https://www.segger.com/downloads/jlink/#J-LinkSoftwareAndDocumentationPack>



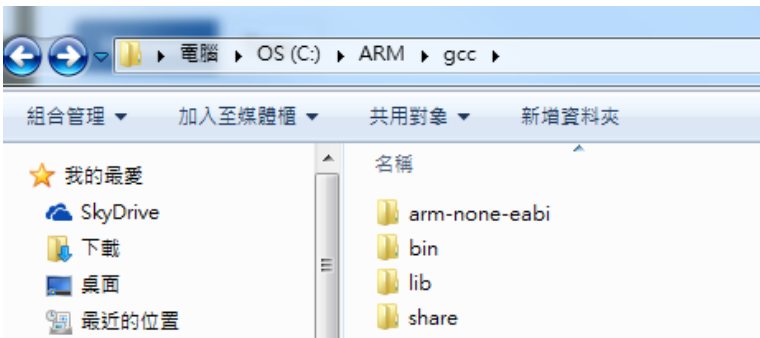
(Step 4) 安装 GCC compiler

下载网址如下，请选择下载 ZIP 档，解压缩后放置 **C:\ARM\gcc** 文件夹内

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



解压缩后放置的位置如下:



(Step 5) 安装 windows-build-tool

下载网址如下，请选择下载 ZIP 档，解压缩后放置 **C:\ARM\build tool** 文件夹内

<https://github.com/xpack-dev-tools/windows-build-tools-xpack/releases/tag/v2.12.2/>

xPack Windows Build Tools v2.12.2

ilg-ul released this on 14 Jul 2020 · 28 commits to xpack since this release







downloads@v2.12.2 5.4k

Version 2.12.2 is a maintenance release; it repacks the same tools from the previous release, but built with the new XBB v3.2 tools.

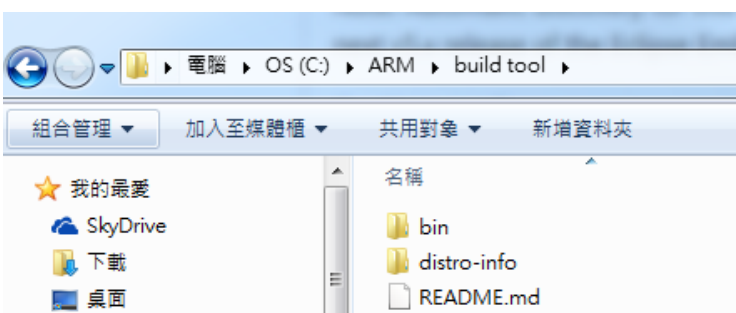
Note: Automatic discovery for this new package will be available in the next v5.x release of the Eclipse Embedded CDT plug-ins.

[Continue reading »](#)

Assets 6

 xpack-windows-build-tools-2.12.2-win32-x32.zip	1.62 MB
 xpack-windows-build-tools-2.12.2-win32-x32.zip.sha	113 Bytes
 xpack-windows-build-tools-2.12.2-win32-x64.zip	1.84 MB
 xpack-windows-build-tools-2.12.2-win32-x64.zip.sha	113 Bytes
 Source code (zip)	
 Source code (tar.gz)	

解压缩后放置的位置如下:



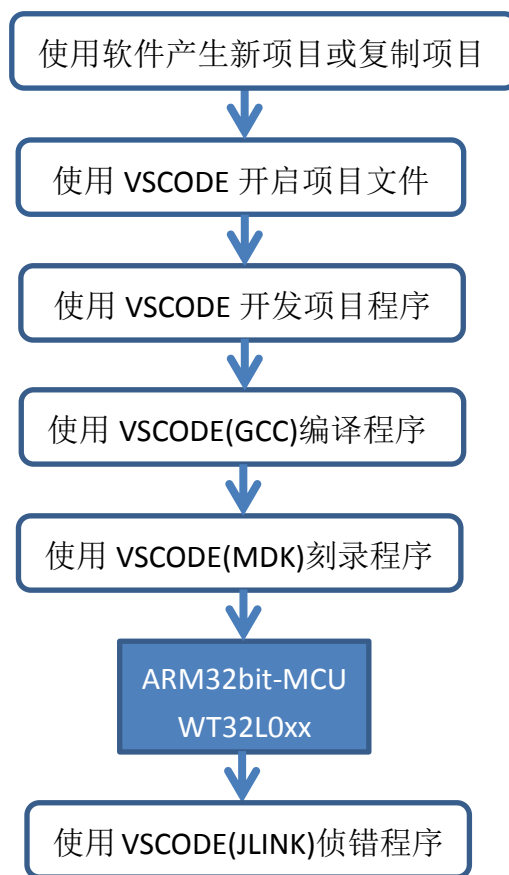
本文件为伟詮电子股份有限公司机密数据，未经许可不得擅自复印或备份。

(Step 6) 安装 WT32FLASH

执行安装项目 WT32FLASH_setup.exe 后可将 c:\WT32FLASH\WT32_LED 复制到任意的工作文件夹并使用 VSCODE 开启(Open folder)后可进行修改。

2. VSCODE 操作流程说明

建立应用方案可复制旧项目进行修改或使用 GUI 软件产生新项目,使用 GUI 软件自动产出对应方案的程序代码,范例程使可参考前章节 WT32FLASH 的路径 C:\WT32FLASH\ WT32L064_LED 项目,上述两个方式都支持 VSCODE 与 MDK 两种类型项目,范例原始码并可使用 VSCODE 或 MDK 进行编译,任何在项目内 SRC 的 C 档案都会进行编译,将编译后的 HEX 烧入至目标 IC,并可对该 IC 使用 VSCODE 进行验证与侦错工作,标准开发流程如下图所示。



2.1 代码产生器软件安装

于微软操作系统 WIN7 以上的计算机端上安装 WT32AutoGen_setup 后执行 WT32AutoGen_V1xx.exe 即可,开启软件后可先加载参数档 AutoGen_xxx.cfg (与执行档放置同个文件夹内)。

2.2 代码产生器软件说明

WT32AutoGen 可依照用户设定产出基础范例程序,若不使用 GUI 软件产出程序,用户可复制原本的范例项目并自行修改 SRC 文件夹内的程序,或任意增减 SRC 文件夹内的 C 档案(*.c)后可进行 GCC 编译,详细请参阅应用文件-泛用型代码产生器说明。

3. VSCODE 平台环境说明

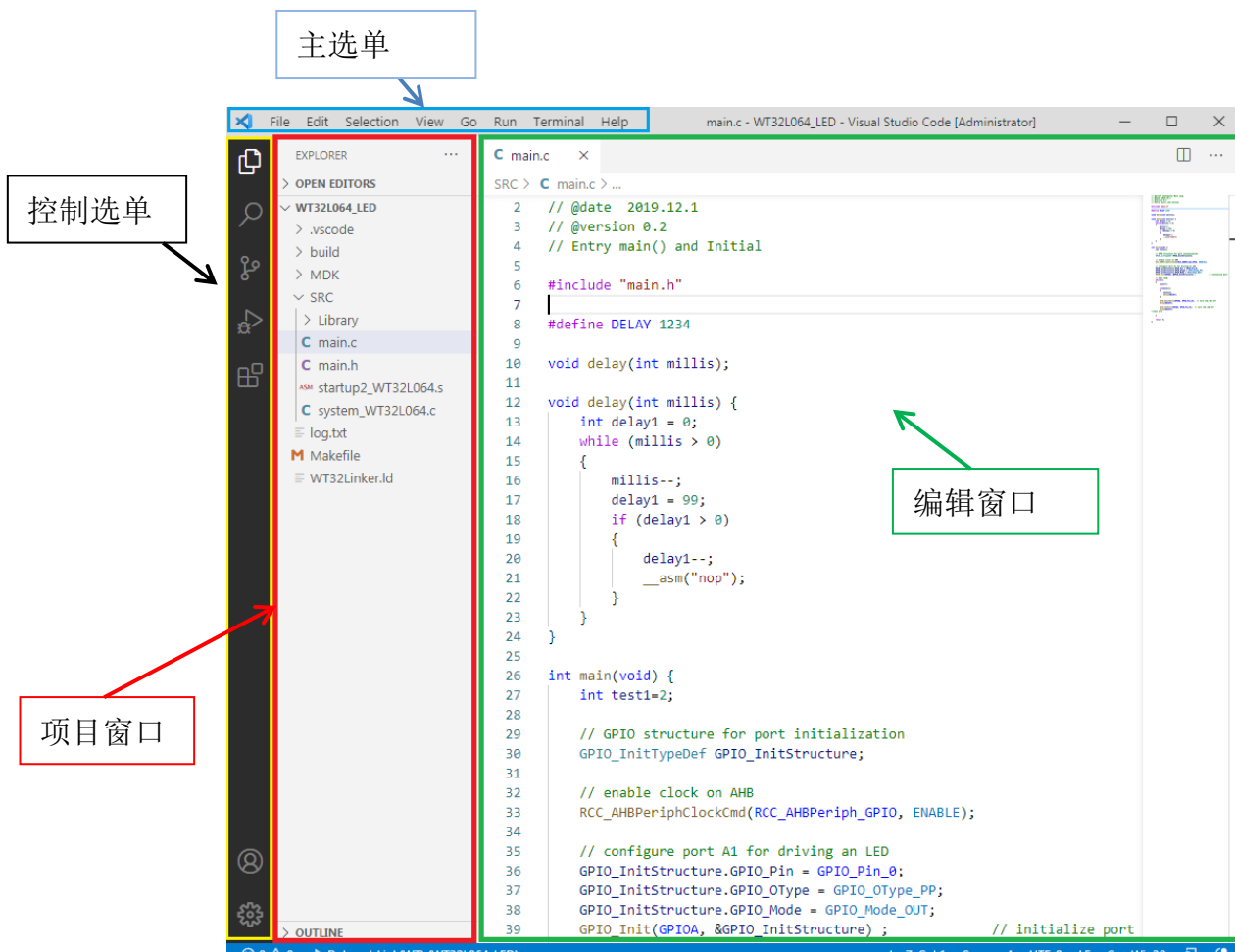
开启 VSCODE 软件后，有四个主要区域分别为主选单、控制选单、项目窗口、编辑窗口，个别功能概述如下，常用功能于下个章节有细部说明。

主选单: 开档、编辑、选择、检视、跳行、执行、任务、协助


控制选单: 项目浏览、查询、源档控制、侦错、扩充

项目窗口: 对应控制选单进行切换任务，与浏览档案

编辑窗口: 进行文字编辑的区域



所有的编辑工作都在编辑窗口内完成，所有关键词会自动反蓝凸显，可使用鼠标右键(Go To Define)

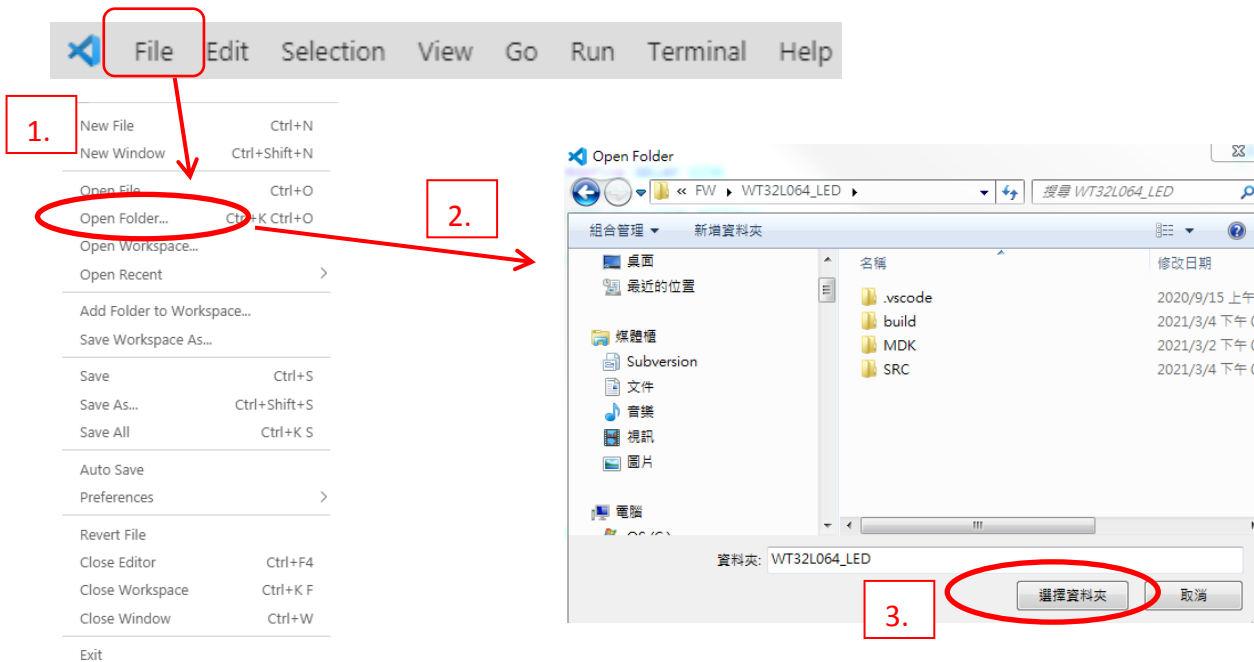
跳跃至源文档， 點選左上 ICON  项目窗口会显示所有工作文件夹内的档案清单。

3.1 主选单

如下图所示其常用选单功能为 File 与 Terminal，File 功能可开启档案、项目文件夹，Terminal 可执行编译、刻录、清除等任务，下列章节进行范例与说明。

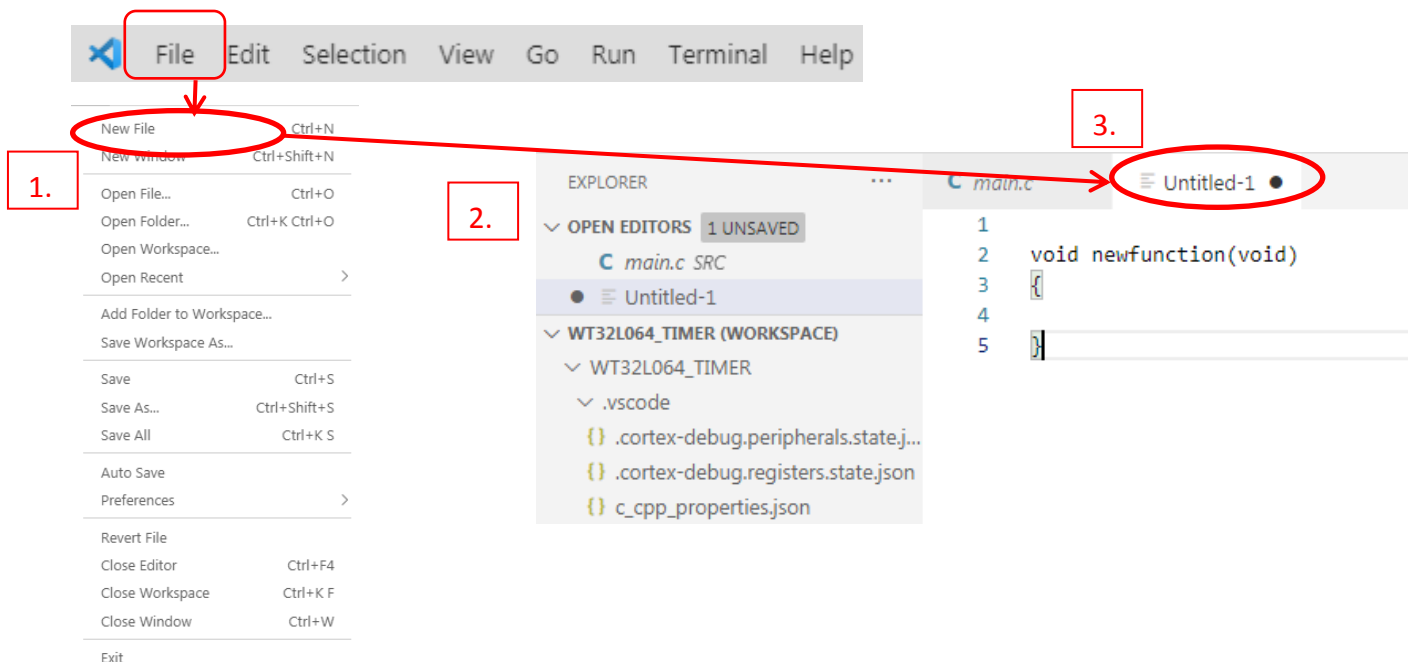
3.1.1 开启项目操作范例:

选择 File -> 选择 Open Folder-> 选择目标项目的文件夹。



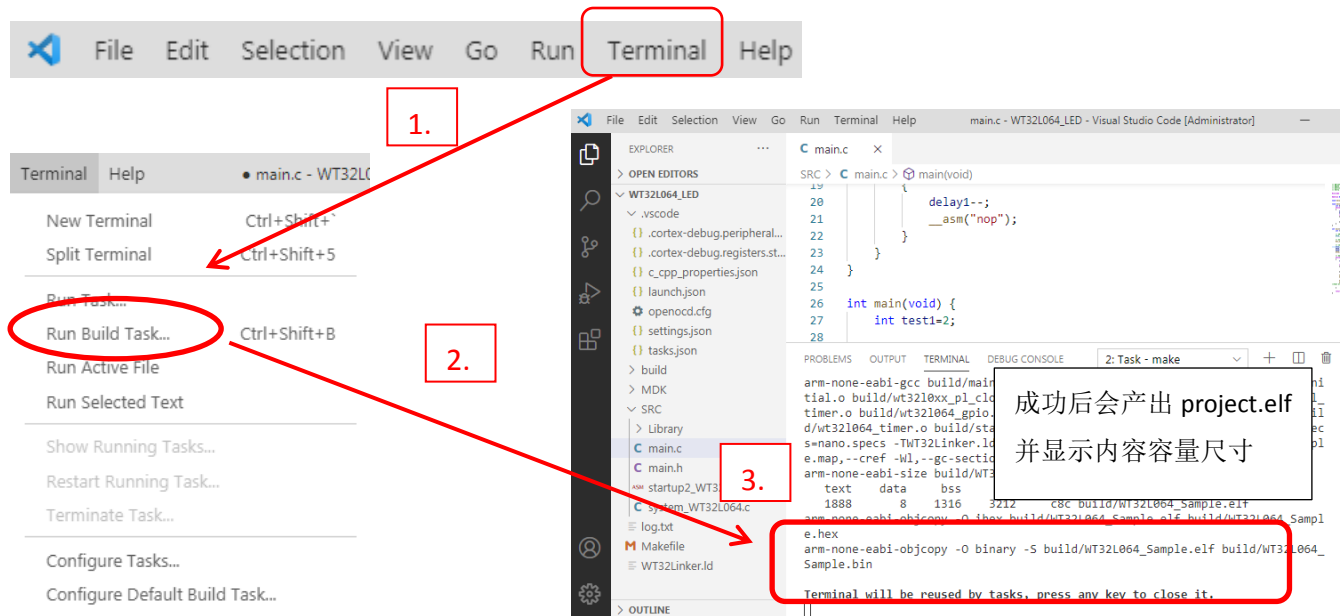
3.1.2 建立新 C 档案操作范例:

选择 File -> 选择 New File-> 出现新档 Untitled-1，可另存成 xxx.c 于 SRC 文件夹内方可一同编译。



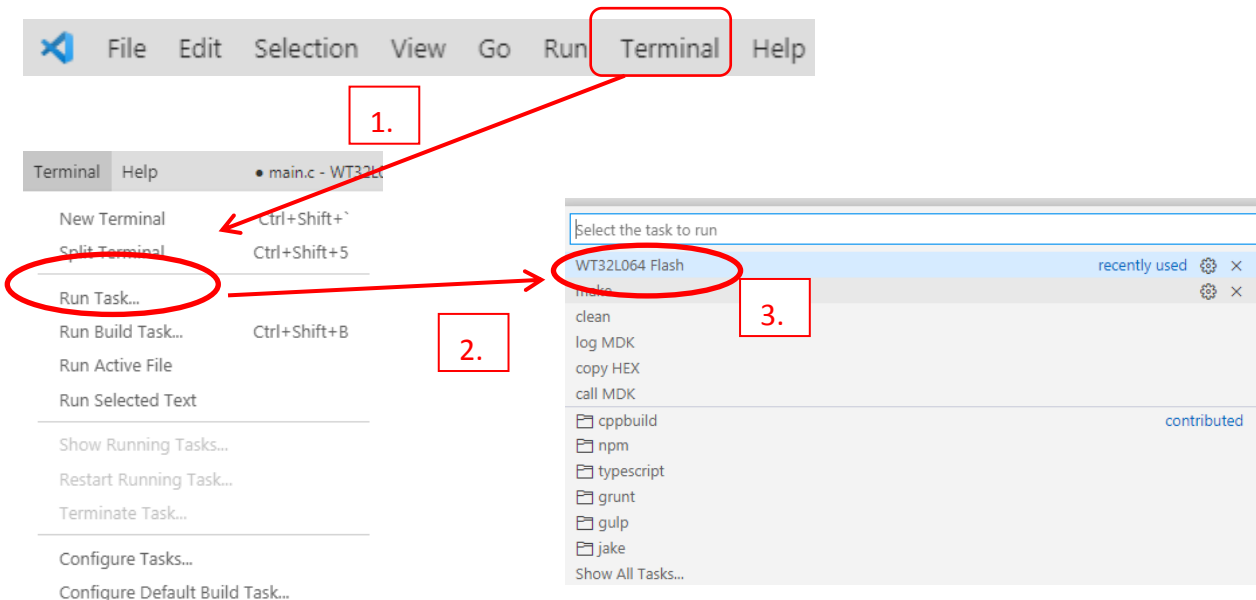
3.1.3 进行 C 程序编译操作范例:

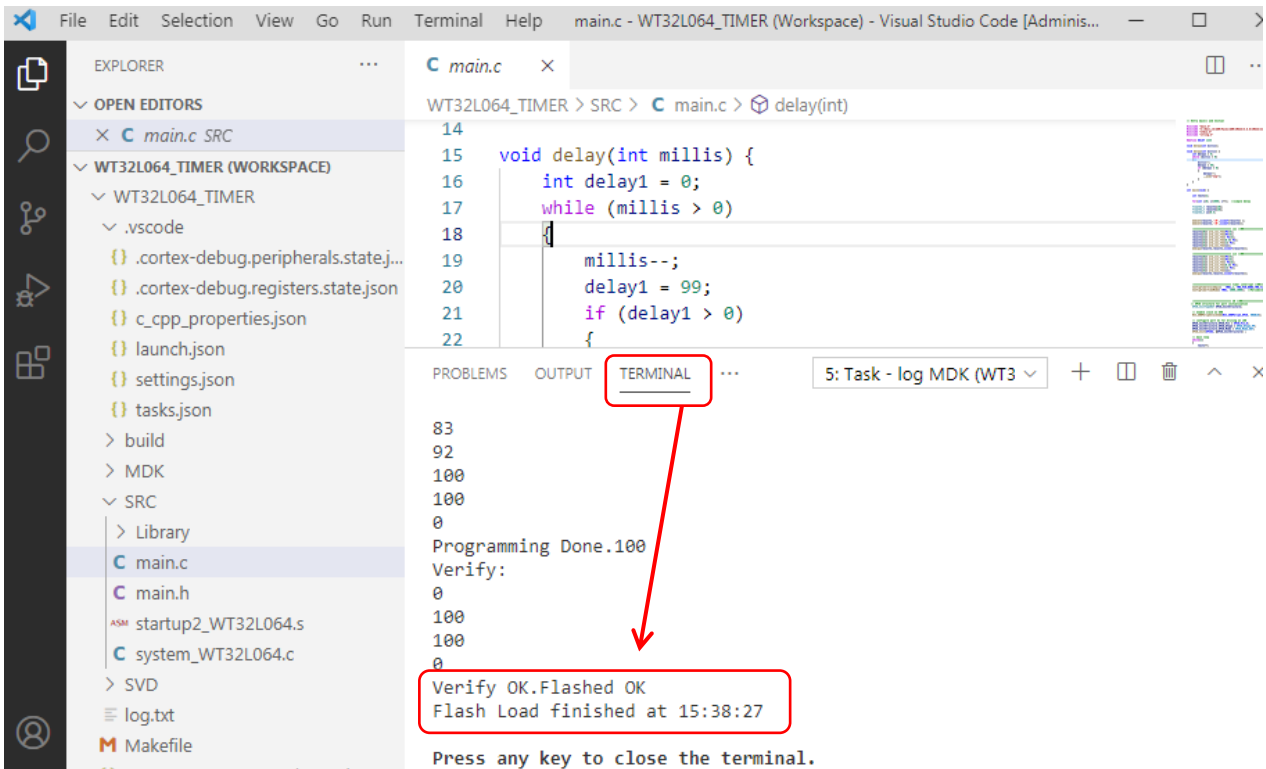
选择 Terminal-> 选择 Run Build Task-> 显示编译结果与侦错(ELF)并刻录(HEX)用档案, 如下图所示。



3.1.4 开启 Terminal 操作刻录范例:

选择 Terminal-> 选择 Run Task-> WT32L064 Flash 进行刻录(HEX)档案, 刻录成功后结果会显示在终端机窗口如下图所示。



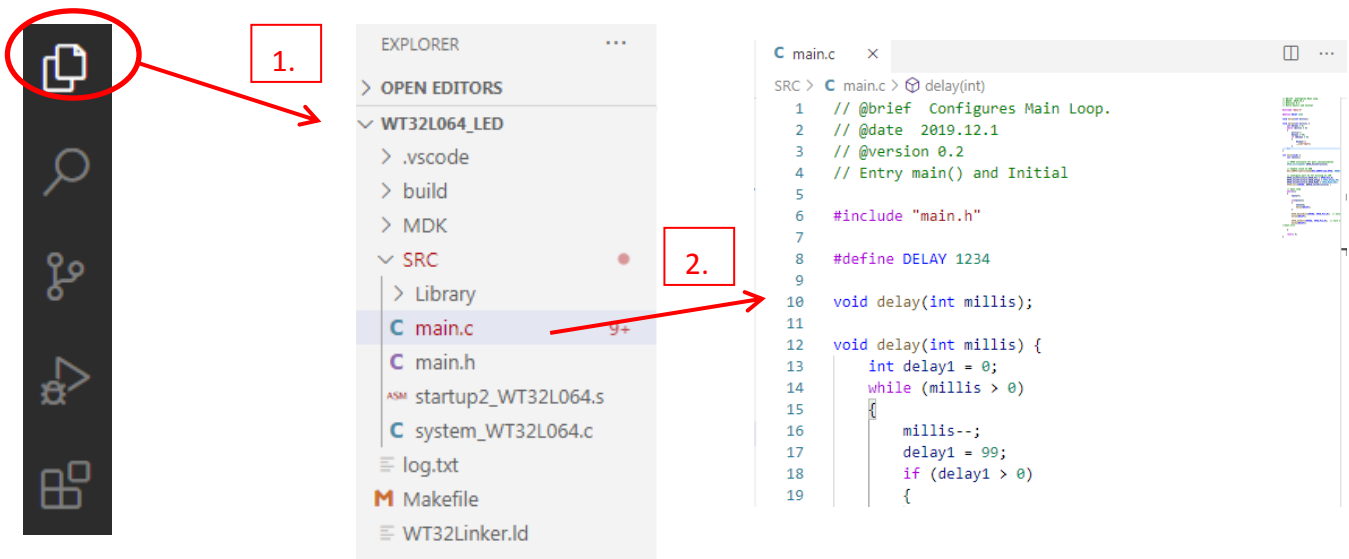


3.2 控制选单

控制选单一般在软件画面左侧，常用功能有浏览文件夹、侦测，扩充功能一般仅在第一次安装后会使用，下列章节进行范例与说明。

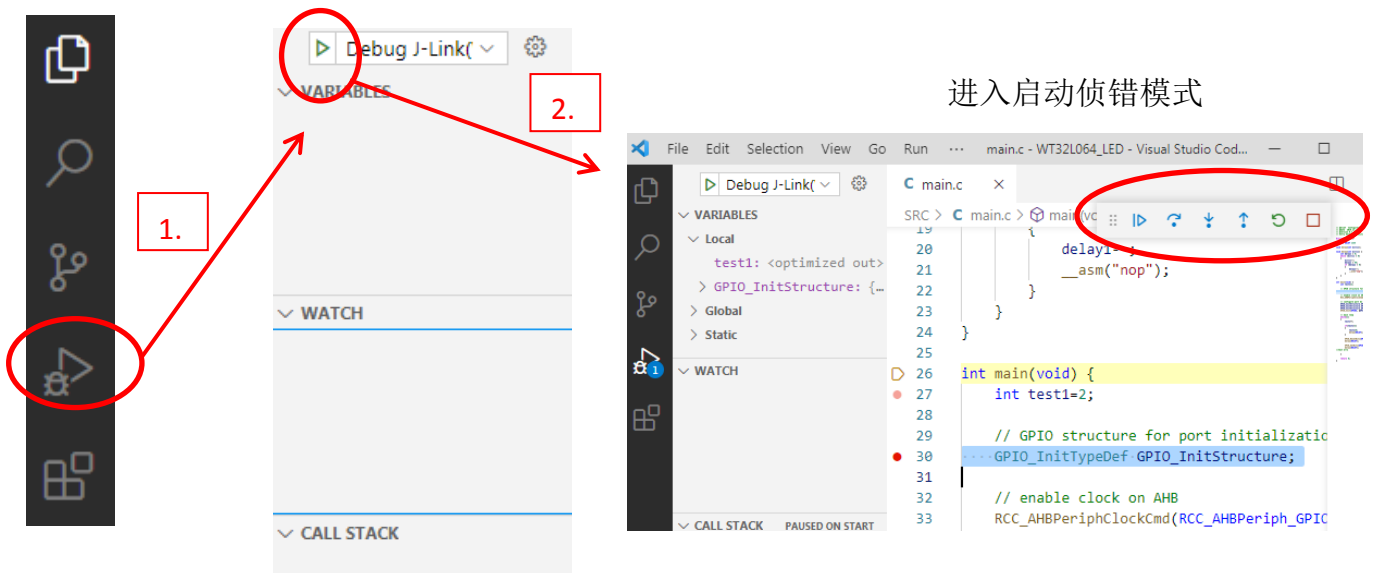
3.2.1 开启项目文件列表范例:

选择左侧边文件图标->会显示出目前档案内容列表



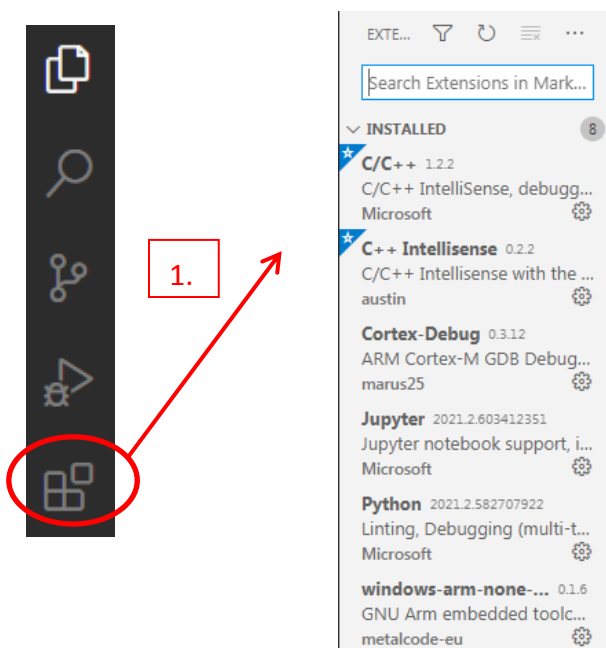
3.2.2 开启侦错(Debug)操作范例:

选择左侧边侦错三角图标->会显示出侦错控制列表-> 选择上方小绿色三角图示, 启动侦错



3.2.3 开启扩充组件操作范例:

选择左侧边方块图标->会显示出目前已经安装的扩充功能, 一般只会在安装初期使用, 主要四个必要组件为 C/C++、C++ Intellisense、Cortex-Debug、windows-arm-none-eabi



3.3 项目文件夹内容

开启项目文件夹进行浏览，其内容主要分四个文件夹，分别为.vscode、build、MDK、SRC，最外层为log.txt、Makefile、WT32Linker.Id三个档案，功能与图标如下依序说明。



Makefile: 针对 GCC 编译程序的设定

WT32Linker.Id: 针对 GCC 产出的 OBJ 进行链结与产出刻录档

startup2_wt32l064.s: 针对 GCC 使用的开机初始化档

startup_wt32l064.s: 针对 MDK 使用的开机初始化档

4. VSCODE 程序开发与除错

进行程序开发若有写法错误会出现编译失败并无法产生 ELF 与 HEX 刻录档，也无法进行侦错，错误主要分两类为文法错误和逻辑错误，硬件使用 J-Link 传输 SWD 讯号并执行除错命令，下列说明如何进行错误的程序写法的排除。

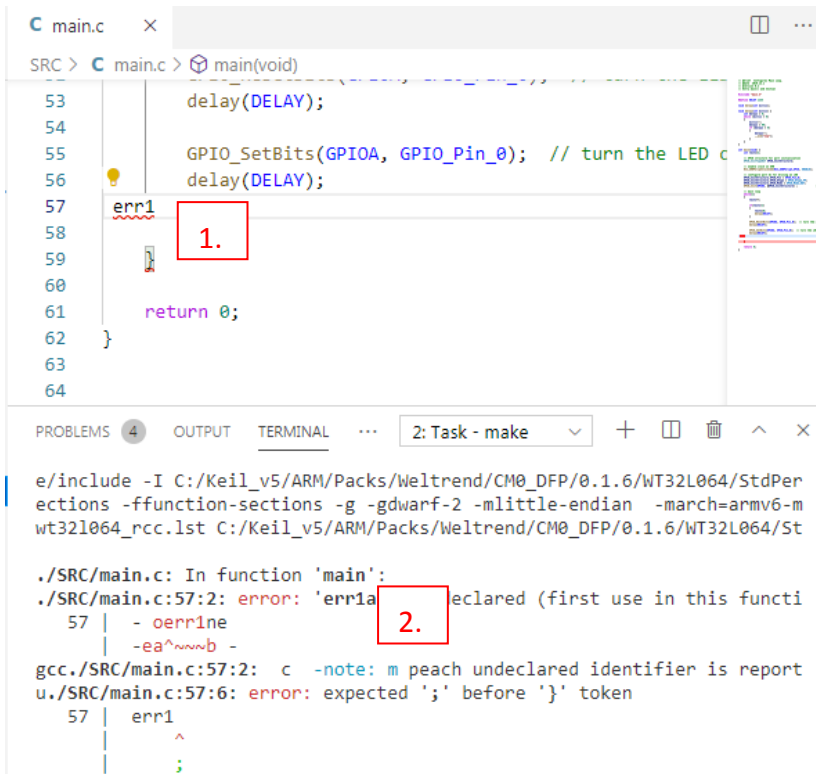
4.1 编译功能说明

程序编写完成后，可于上方主选单选择 Terminal->Run Build Task，或按下快捷键 CTRL+SHIFT+B，编译成功会将出现下方提示文字于终端机窗口，并告知产出 HEX 档案与其路径，默认路径为项目名称\build，其中 ELF 档案为侦错 DEBUG 使用

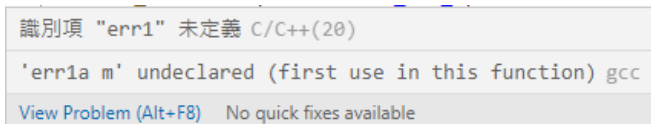
```
thumb -specs=nano.specs -TWT32Linker.ld -lc -lm -lnosys -Wl,-Map=build/WT32L064_Sample.map,  
creef -Wl,--gc-sections -o build/WT32L064_Sample.elf  
arm-none-eabi-size build/WT32L064_Sample.elf  
text    data    bss     dec      hex filename  
1888     8    1316    3212     c8c build/WT32L064_Sample.elf  
arm-none-eabi-objcopy -O ihex build/WT32L064_Sample.elf build/WT32L064_Sample.hex  
arm-none-eabi-objcopy -O binary -S build/WT32L064_Sample.elf build/WT32L064_Sample.bin
```

4.2 文法错误排除

一般错误写法出现时会立即出现红色波浪如下图标示 1，进行编译时下方输出画面则会出现红色字样如下图标示 2。



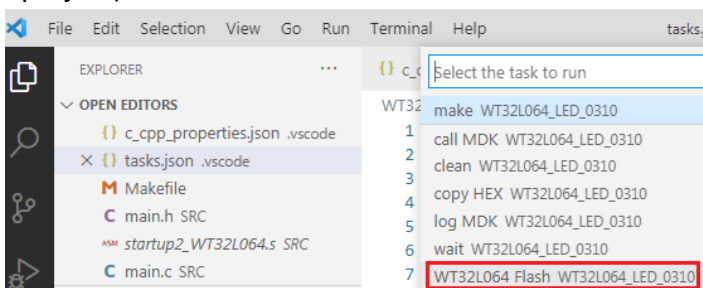
标示 1 的错误提示，将鼠标移动到波浪处，会立即出现说明



标示 2 的错误提示，则会出现关键档名与行数列数，按下 CTRL+鼠标左键，编辑器会协助跳至错误处，可进行程序编辑除错。

4.3 刻录功能说明

编译完成后并产出 HEX 档案，可参考 3.1.2 说明操作刻录程序，点选下方图标 WT32L064 FLASH xxx (your project)选单，之后会于下方终端机窗口出现刻录成功讯息，如下图所示。



刻录成功的讯息如下。

```

Hardware-Breakpoints: 4
Software-Breakpoints: 8192
Watchpoints: 2
JTAG speed: 1000 kHz

100
Full Chip Erase:
0
100
0
Full Chip Erase Done.100
Program:
0
54
100
100
0
Programming Done.100
Verify:
0
100
100
0
Verify OK.Flashed OK
Flash Load finished at 18:43:19
Press any key to close the terminal.
    
```

4.4 逻辑除错功能

一般逻辑错误会使用 ICE 仿真模拟进行除错，编译完成后并产出 HEX 档案，参考 3.2.2 章节操作，點選下方绿色三角图示 Debug-Jiink(WT)，编辑窗口会出现向右红框箭头并停在 main.c 程序起始第一行，下方终端机会出现呼叫 arm-none-eabi-gdb 字样并提出目前已停止在 int main(void) 如下图所示。

EXTENSION MARKETPLACE

- INSTALLED (8)
- C/C++ 1.2.2
- C/C++ IntelliSense, debugg...
- Microsoft
- C++ IntelliSense 0.2.2
- C/C++ IntelliSense with the ...
- austin
- Cortex-Debug 0.3.12
- ARM Cortex-M GDB Debug...
- marus25
- Jupyter 2021.2.603412351
- Jupyter notebook support, i...
- Microsoft
- Python 2021.2.582707922
- Linting, Debugging (multi-t...
- Microsoft
- windows-arm-none-... 0.1.6
- GNU Arm embedded toolc...
- metalcode-eu

```


C main.c x
SRC > C main.c > delay(int)
25
26 int main(void) {
27 int test1=2;
28
29 // GPIO structure for port initialization
30 GPIO_InitTypeDef GPIO_InitStructure;
31
32 // enable clock on AHB
    
```

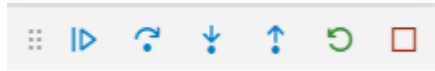
```

DEBUG CONSOLE
Filter (e.g. text, !exclude)

rogram Files (x86)/SEGGER/JLink_V512/JLinkGDBServerCL.exe
Launching server: "C:/Program Files (x86)/SEGGER/JLink_V512/
LinkGDBServerCL.exe" "-if" "swd" "-port" "50000" "-swoport"
"50001" "-telnetport" "50002" "-device" "Cortex-M0"
Launching GDB: "C:\ARM\gcc\bin\arm-none-eabi-gdb.exe" "-q" "
-interpreter=m12"
undefinedC:\ARM\gcc\bin\arm-none-eabi-gdb.exe: warning: Cou
n't determine a path for the index cache directory.
Reading symbols from D:\vscode\FW\WT32L064_LED/build/WT32L06
_Sample.elf...
0x20000000 in SystemCoreClock ()
Not implemented stop reason (assuming exception): undefined
Resetting target

Temporary breakpoint 1, main () at ./SRC/main.c:26
26 int main(void) {
    
```

此时可以在行列数字段点两下下红点  断点，如下图左侧红点位置，或可点选浮动功能列



进行单步执行或停止功能。

```
36 GPIO_InitStructure.GPIO_Pin = GPIO_Pin_0;
37 GPIO_InitStructure.GPIO_OType = GPIO_OType_PP;
38 GPIO_InitStructure.GPIO_Mode = GPIO_Mode_OUT;
39 GPIO_Init(GPIOA, &GPIO_InitStructure) ;
40
41 // main loop
42 while(1)
43 {
44     test1++;
```

5. 附录

下列补充说明于 VSCODE 平台下主要使用的平台设定与编译程序配置文件案。

5.1 tasks.json 功能说明

在主选单 Terminal->Run Task 下所可执行的任务，依序为 make 编译程序、clean 清除旧文件、WT32L064 Flash 刻录程序，若无特殊任务需求可使用目前默认任务即可，部分内如下所示。

```
"version": "2.0.0",
"options": {
  "env": {
    "path": "c:/ARM/gcc/bin;c:/ARM/build tool/bin;c:/ARM/OCD/bin;"
  },
  "shell": {
    "executable": "${env:windir}/System32/WindowsPowerShell/v1.0/powershell.exe",
    "args": [ "-NoProfile", "-ExecutionPolicy", "Bypass", "-Command" ]
  }
},
"tasks": [
  {
    "label": "make",
    "type": "shell",
    "command": "make -j", // -j: cpu core unlimited -?: explain
    "group": {
      "kind": "build",
      "isDefault": true
    },
    "presentation": {
      "focus": true
    },
    "problemMatcher": [
      "$gcc"
    ]
  },
  {
    "label": "clean",
    "type": "shell",
    "command": "make clean",
    "group": "build",
    "presentation": {
      "focus": true
    },
    "problemMatcher": [
      "$gcc"
    ]
  }
],
}
```

5.2 launch.json 功能说明

侦错使用的连结档，可设定呼叫特定路径的驱动连接至 VSCODE 进行同步侦错，若有变更网桥才需修改，部分内如下所示。

```

"version": "0.2.0",
"configurations": [
  {
    "name": "Debug J-Link(WT)",
    "type": "cortex-debug",
    "request": "launch",
    "cwd": "${workspaceRoot}",
    "executable": "${workspaceRoot}/build/WT32L064_Sample.elf",
    // "serverpath": "C:/Program Files (x86)/SEGGER/JLink/JLinkGDBServerCL.exe",
    "serverpath": "C:/Program Files (x86)/SEGGER/JLink_V512/JLinkGDBServerCL.exe",
    "servertype": "jlink",
    "device": "Cortex-M0",
    "interface": "swd",

    "serialNumber": "", //If you have more than one J-Link probe, add the serial n
    "runToMain": true,
  }
]

```

5.3 Makefile 功能说明

进行 GCC 程序编译的配置文件案, 包含 INCLUDE 路径与所有的目标源始文件进行编译, 可观察 Makefile 文件内容中目前 CMSIS 是否为最新版本如图红色标示, 目前在 SRC 文件夹内都会自动编译。

```

#####
# target
#####
#TARGET = WT32L064_Sample
TARGET := $(notdir $(CURDIR))

#####
# path to Cortex-M0 standard peripheral library
#####
CMSIS_LIBS ?=      C:/Keil_v5/ARM/Packs/Weltrend/CM0_DFP/0.1.6/WT32L064
CMSIS_CORE ?=      C:/Keil_v5/ARM/Packs/ARM/CMSIS/5.6.0/CMSIS

```

5.4 Linker 功能说明

参照 GCC 与 CMSIS 用法进行 Link 设定, 主要设定程序起始地址与 RAM、ROM 长度, 指定程序入口函式并安排中断与数据区域, 若有变更目标 IC 才需更换或修改, 部分内如下所示。

```

ENTRY(Reset_Handler)
MEMORY
{
  FLASH (rx) : ORIGIN = 0x10000000, LENGTH = 0x10000 /* 64k */
  RAM (rwx)  : ORIGIN = 0x20000000, LENGTH = 0x02000 /* 8k */
}
_ram_stack = 0x20002000; /* end of RAM, simon.c */
_Min_Heap_Size = 0x100; /* required amount of heap */
_Min_Stack_Size = 0x400; /* required amount of stack */
SECTIONS
{
  .isr_vector :
  {
    . = ALIGN(4);
    KEEP(*(.isr_vector)) /* Startup code */
    . = ALIGN(4);
  } >FLASH
}

```

6. 版本更改纪录:

版本	内容	时间
V0.1	初版	2021.03.02
V0.2	补充说明 gcc/build tools 安装	2021.03.18
V0.3	补充说明 Extension 与 GCC 默认路径	2021.03.31