文档编号：

上海东软载波微电子有限公司

用户手册

VSCode For ESSEMI

修订历史

版本	修改日期	更改概要
V1.0	2020.09.01	初版
V1.1	2022.08.18	添加数据监视点、可视化配置的说明

地　址：中国上海市古美路 1515 号凤凰园 12 号楼 3F
E-mail：support@essemi.com
电　话：+86-21-60910333
传　真：+86-21-60914991
网　址：http://www.essemi.com/

目录

1 快速入门

1.1 简介

VSCode For ESSEMI（简称 VSCode）是由上海东软载波微电子有限公司自主研发的新一版本的集成开发环境，为用户提供了完整的设计、开发、调试、部署嵌入式应用程序的工具包。本产品是基于 Visual Studio Code 开发的扩展包，其本身对于运行环境要求很低，不需要安装环境，支持 Windows 7 及以上版本、Ubuntu 16.04 及以上版本。本文档仅介绍基于 Visual Studio Code 扩展的相关功能，其他功能可参考 Visual Studio Code 官网（https://code.visualstudio.com/docs）进行查看。

1.2 Windows

1.2.1 安装VSCode

运行 VSCode 的安装程序，进入安装界面，依次执行以下操作：

1. 打开 VSCode 安装软件，跳出“用户账户控制页面”，点击“是”。
2. 进入 VSCode 安装程序界面，单击“下一步”。
3. 选择安装目录，单击“下一步”。
4. 单击“完成”。

1.2.2 安装驱动

VSCode 支持用户使用 ES10M、ESLink2 设备调试 8 位项目，或使用 ESLink2、JLink 设备调试 32 位项目，调试之前需要首先安装驱动程序。驱动程序在以下目录：

C:\用户\用户名\.vscode\extensions\luh.esext-版本号\Driver

- ES10M 驱动：USBXpressInstaller.exe。
- ESLink2 驱动：Windows7 环境需要安装；Windows 8/10 无需另外安装。
- JLink 驱动：Setup_JLink_V500h.exe。

1.2.3 安装工具链

HR7P/ES7P/ES7P-V2 项目需要使用 HRCC 工具链，Cortex-M 项目需要使用 ES32CC 工具链或 GCC 工具链，RISC-V 项目需要使用 ES32CC 工具链或 GCC 工具链，编译项目之前需要首先安装相对应的工具链。

- ES32CC 工具链：ES32CC_v1.0.0.9.exe，ES32 工具链依赖 vc2015，因此还需要安装 vc_redist2015.x86.exe；
- HRCC 工具链：HRCC_v1.2.0.118.exe。

1.2.4 卸载VSCode

用户可以通过以下方式卸载 VSCode：

- 在“开始”菜单中选择“卸载 VSCode”，在“用户账户控制页面”单击“是”→卸载界面单击“下一步”→单击“完成”；
- 在“控制面板”中选择“程序和功能”，右击“VSCode”，在快捷菜单中选择“卸载”，在“用户账户控制页面”单击“是”→卸载界面单击“下一步”→单击“完成”。

1.3 Linux

VSCode 在 Linux 环境下支持 ES32/ES8P 系列 32 位芯片开发，支持使用 JLink 进行仿真调试；不支持 ES7P 系列 8 位芯片编译调试。

1.3.1 安装VSCode

在 Linux 下安装 VSCode 依次执行以下操作，桌面会生成 VSCode 快捷方式：

1. 复制 ESSEMI.tar.gz 及 Install.sh 到目标安装目录；
2. 在终端窗口运行 Install.sh 脚本。

2 快速入门

2.1 启动界面

VSCode 具有简单直观的布局，可最大程度地为编辑器提供空间，同时为浏览或访问文件夹或项目的整个上下文留出足够的空间。用户界面主要分为以下五个区域：菜单工具栏、活动栏、边栏、状态栏、面板，如下图所示。关于 VSCode 界面相关的详细介绍可参考 Visual Studio Code 官网：https://code.visualstudio.com/docs/getstarted/userinterface。

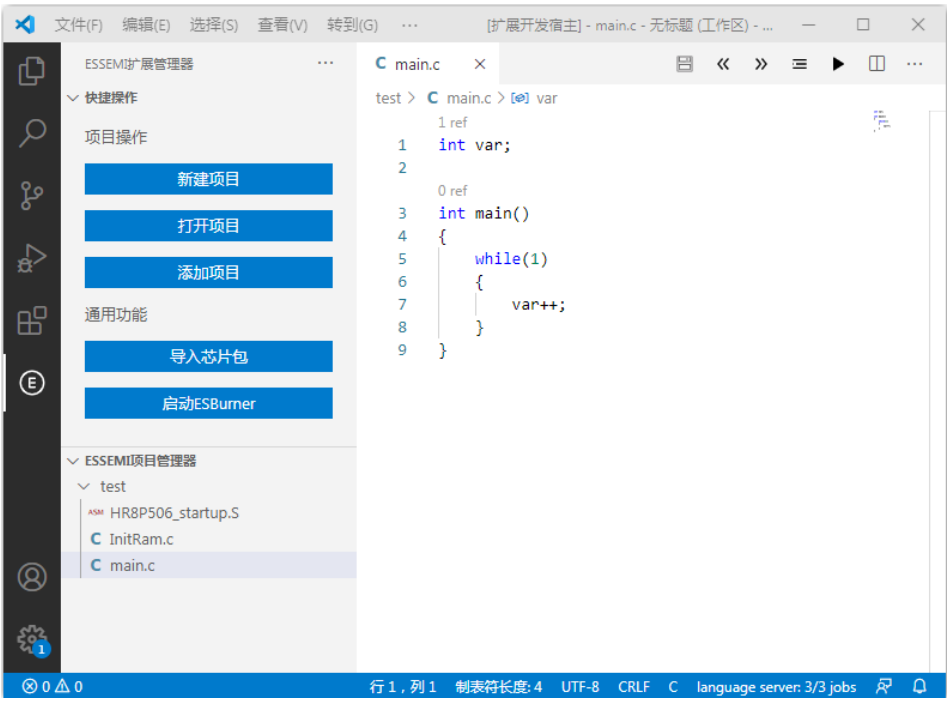


图 2-1 启动界面

菜单工具栏：包括“文件”、“编辑”、“选择”、“查看”、“转到”、“运行”、“终端”等主菜单项，通常位于界面的顶部。根据应用状态不同，菜单栏的项目状态也随之更改。VSCode 的绝大多数操作都可以通过相应的菜单命令完成。对于常用的菜单命令或者菜单项，VSCode 都提供了快捷方式，用户可以通过相应的快捷键访问。

活动栏：位于最左侧，可让用户在六个视图之间切换，包括：资源管理器、搜索、源代码管理、运行、扩展、ESSEMI 扩展管理器。其中，“资源管理器（ ）”用来浏览、打开和管理项目所在的文件夹。“运行（ ）”用来配置调试项目且在调试状态下显示调试相关窗口：变量、监视、调用堆栈、寄存器以及断点。“扩展（ ）”视图用来显示已启用的插件包以及用户可以根据自己的需求在商店里安装相应的插件包。“ESSEMI 扩展管理器（ ）”视图用来管理项目相关的操作以及显示项目的树形视图。

状态栏：有关打开的项目和用户编辑的文件信息。

2.2 创建项目

用户程序一般由源程序文件及某些相关文件组成，通常将这些元素称为“项”（item），为了帮助用户有效地管理开发过程中所需的项，VSCode 提供了“ESSEMI 项目管理器”，以 Group 的形式管理项目文件。“ESSEMI 项目管理器”可包含多个项目，而一个项目通常包含多个项。

用户可以通过多种方式进行项目的构建：新建项目、打开或添加项目，将构建好的项目在“ESSEMI 项目管理器”中以树形视图显示同时还支持多项目的编译和调试。

用户可以通过“ESSEMI 项目管理器”对项目进行查看、管理项目及项。“ESSEMI 项目管理器”是 VSCode 的重要组成部分，通常点击活动栏的“ESSEMI 扩展管理器（  ）”图标，则关于项目的相关操作视图出现在界面的左侧，如下图所示。其中，在“快捷操作”一栏提供了项目操作相关内容：新建项目、打开项目、添加项目；以及通用功能：导入芯片包和启动 ESBurner。

图 2-2“ESSEMI 扩展管理器”窗口

通过“项目资源管理器”窗口可完成以下任务：

- 关闭项目。
- 添加、删除、重命名项目中的分组（group）。
- 添加、移除、重命名、比较文件。
- 打开需要编辑的文件。
- 编译文件或项目。
- 查看项目的属性。
- 清理输出目录中产生的临时文件

以上任务都可以通过在“ESSEMI 项目管理器”窗口不同结点位置的右键快捷菜单中的相应菜单命令完成。

2.3 新建项目

1. 单击 VSCode 活动栏的“ESSEMI 扩展管理器”图标，然后单击“新建项目”，出现“选择文件夹”对话框，如下图所示：

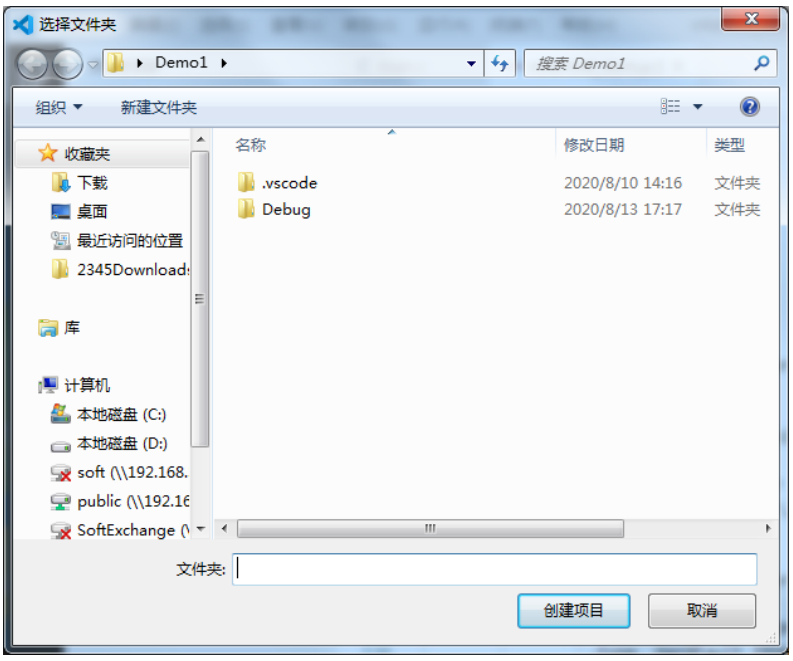


图 2-3“新建项目”对话框

2. 在“文件夹”栏中选择或输入保存项目的位置。
3. 单击“创建项目”。
4. 在 VSCode 界面出现项目名称输入框，用户根据自己需求输入项目名称，则创建相应的项目目录，如下图所示：

2-4“项目名称”输入框

5. 在 VSCode 界面出现芯片类型选择框，支持 HR7P/ES7P/ES7P-V2 架构 8 位芯片类型，及 Cortex-M 架构、RISC-V 架构 32 位芯片类型。用户根据自己的项目需求选择相应的芯片架构。如下图所示：

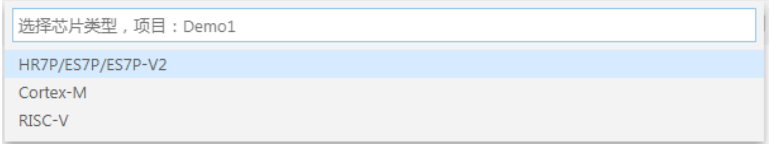


图 2-5“芯片类型”选择框

6. 在 VSCode 界面出现所选芯片类型对应的芯片列表下拉框，如下图所示，点击选择芯片名

称，即可成功创建项目，并将项目加载到“项目资源管理器”中。如下图所示：

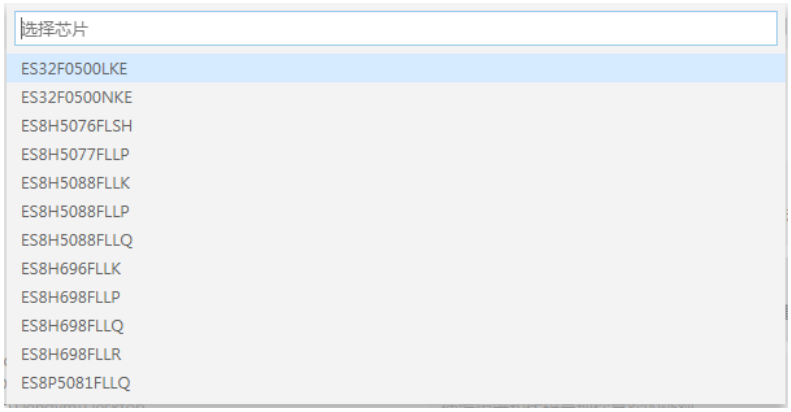


图 2-6“芯片列表”下拉框

注意：如果“项目资源管理器”中已存在其他项目，则先关闭当前所有项目，再将新建项目加载到“项目资源管理器”中，同时，也会关闭工作区当前所有项目文件夹，再将新建项目所属的文件夹加载到工作区中。

2.3.1 打开项目

VSCode 支持直接打开 ESSEMI（.escfg）项目，以及将 iDesigner（.hrccproj）项目和 Keil（.uvproj）项目转换为 VSCode 所支持的项目，即通过“打开项目”操作直接将项目加载到“项目资源管理器”中。操作步骤如下：

1. 单击 VSCode 活动栏的“ESSEMI 扩展管理器”图标，然后单击“打开项目”，出现“打开”对话框，如下图所示：

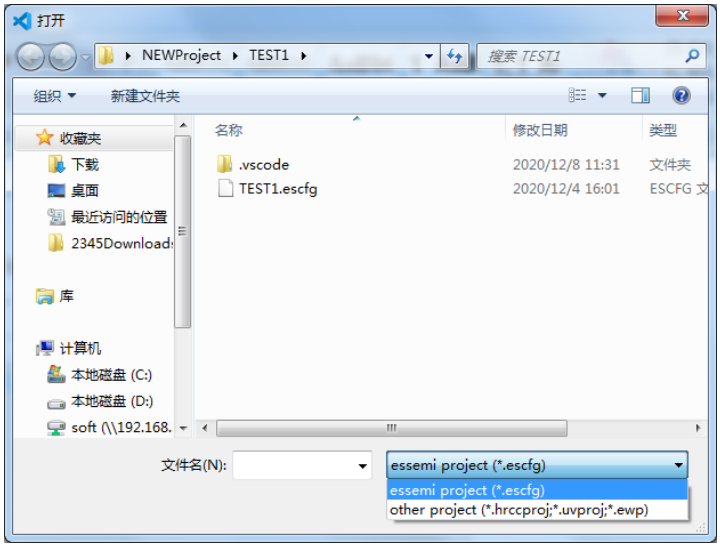


图 2-7 “打开项目”对话框

2. 在“文件名”一栏中选择或输入 ESSEMI（.escfg）项目文件，若要打开其他类型项目（如：Idesigner 项目、Keil 项目等）则下拉文件过滤框，切换到其他类型项目栏（other project），如上图所示。

3. 单击“选择项目文件”按钮，项目成功加载到“ESSEMI 项目管理器”中。

注意：如果“ESSEMI 项目管理器”中已存在其他项目，则先关闭当前所有项目，再将新建项目加载到“ESSEMI 项目管理器”中；同时，也会关闭工作区当前所有项目文件夹，再将新建项目所属的文件夹加载到工作区中。

2.3.2 添加项目

用户可以通过“添加项目”操作，向“ESSEMI 项目管理器”中添加多个项目。具体操作步骤如下：

1. 单击 VSCode 活动栏的“ESSEMI 扩展管理器”图标，然后单击“添加项目”，出现“选择文件夹”对话框，如下图 2-8 所示。
2. 在“文件名”一栏中选择或输入项目文件，如：ESSEMI 项目（.escfg）、idesigner 项目（.hrccproj）、Keil 项目（.uvproj）等。
3. 单击“选择项目文件”按钮，则成功将项目添加到“ESSEMI 项目管理器”中，且将项目所在文件夹添加到工作区中。

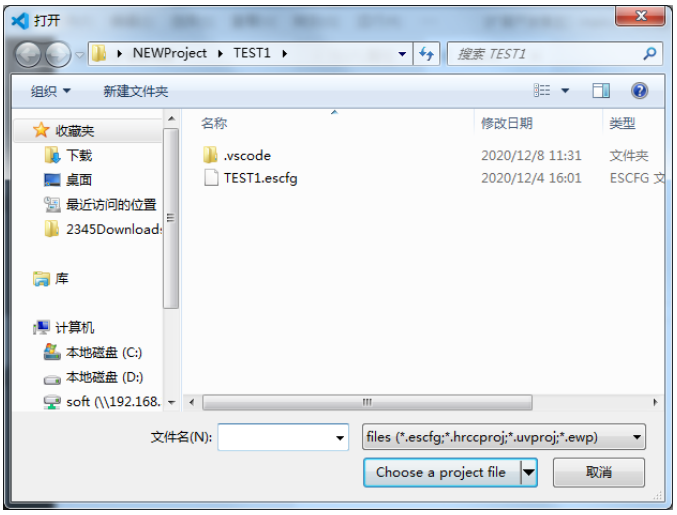


图 2-8 “添加项目”对话框

2.3.3 属性设置

VSCode 在生成项目、启动调试之前需要先设置项目的属性，如果用户没有设置，VSCode 将使用默认的配置进行项目编译和启动调试。

属性界面

打开项目，在“ESSEMI 项目管理器”中选中项目任意位置右键出现操作菜单，单击“属性”菜单，在主编辑窗体显示该项目的属性页，用户可以通过切换选项卡切换：常规选项页、芯片选项页、调试选项页、编译选项页。如下图所示：

图 2-9 属性界面

注意：属性设置更改后，用户需要手动点击更改页的保存按钮。

芯片配置

每一个新建的项目都需要设置芯片类型，使之与硬件设备相匹配。在项目的任意节点位置右击，即出现操作菜单，单击“属性”菜单栏，在主编辑窗体显示该项目的属性页，通过切换选项卡切换到“常规选项”页，可在“选择芯片”栏里看到当前芯片名称、芯片搜索框、芯片列表。用户可通过以下方式更换芯片：

- 在“查找”栏里输入芯片名称，在芯片列表里显示相应的芯片名，单击芯片名并且点击“保存常规配置”按钮，则芯片更换成功，如下图所示。
- 直接在芯片列表里下滑查找所需芯片，单击芯片名称并且点击“保存常规配置”按钮，芯片更换成功。

图 2-10“常规选项”页

2.4 编辑代码

VSCode 提供了丰富的代码编辑功能，具有高效、强大、准确的语言支持：

- 代码补全
- 诊断提示
- 悬停提示
- 函数签名
- 查找所有引用
- 符号重命名
- 显示文件所有符号
- CodeLens
- 格式化代码
- 文件比较

2.5 编译项目

VSCode 支持多种编译形式：

- 编译项目：部分编译，仅编译当前项目。
- 重新编译项目：对当前的项目重新编译。
- 全部编译：部分编译，“ESSEMI 项目管理器”下的所有项目编译。

- 全部重新编译：将“ESSEMI 项目管理器”下的所有项目重新编译。
- 编译文件：对单个指定的项目文件进行编译。

代码编辑完成后，需要编译项目，生成 Hex 文件。在“ESSEMI 项目管理器”下右击项目任意位置，用户可根据需求从快捷菜单中选择相应的编译形式。如图红色方框内所示：

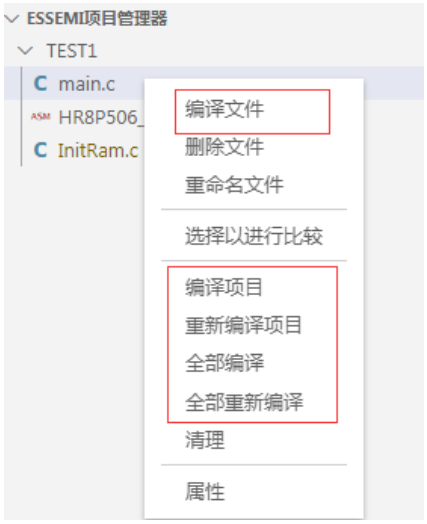


图 2-11 编译形式

编译完成之后，输出窗口中显示编译信息，包括编译结果、编译耗时、错误信息等，如下图所示：

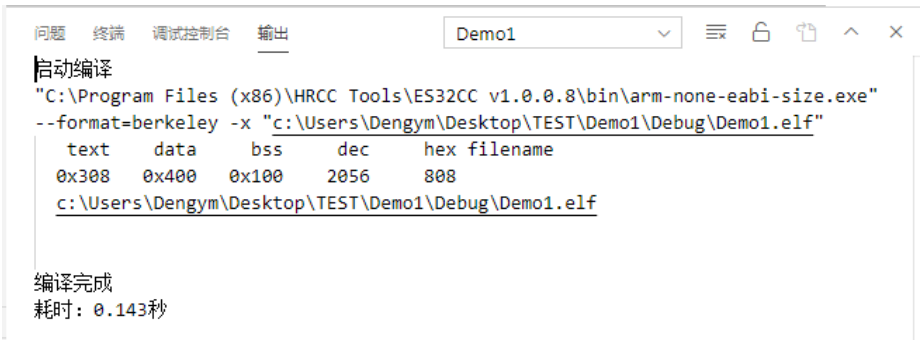


图 2-12“输出”窗口

若程序代码有错，用户可以在“问题”窗口中查看错误、警告信息，双击错误或警告信息可在编辑区域打开错误文件并跳转至错误或警告所在位置，如下图所示：

图 2-13“问题”窗口

2.6 调试项目

在成功编译生成项目并正确连接硬件设备后，用户即可调试项目。

配置字配置

在项目的任意节点位置右击，出现快捷菜单栏，点击“属性”菜单，在主编辑窗口出现属性界面，切换到“芯片选项”页，可看到“芯片配置”窗口，如下图所示：

图 2-14“芯片配置”窗口

注意：芯片配置字中 CFG_DEBUG 位必须设置为 Enable 才可以进入调试模式。用户在更改属性设置之后需手动点击更改页的保存配置按钮。

连接设置

启动调试前，用户需要在属性界面设置好相对应的调试工具。在“ESSEMI 项目管理器”下项目的任意位置右击，出现快捷菜单栏，单击“属性”菜单在主编辑窗口显示属性界面，切换选项卡到“调试选项”页，在调试工具下拉选项一栏，用户可下拉选择相对应的调试工具。如下图所示：

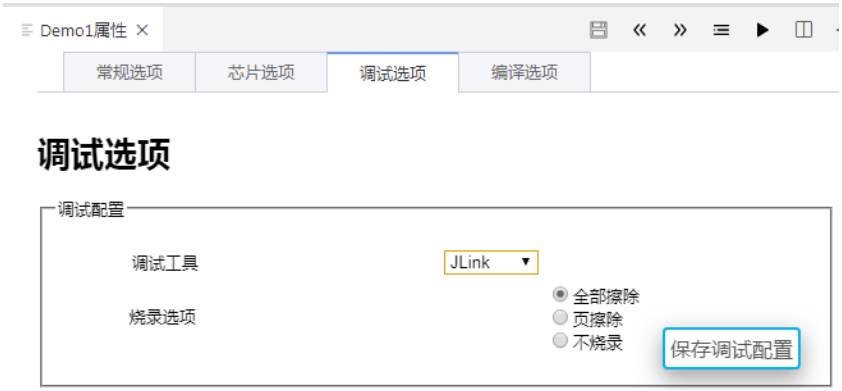


图 2-15“调试配置”窗口

启动调试

VSCode 支持多种方式启动调试：

- 方式①：点击主编辑界面上方的“启动调试（ ）”图标，即可启动调试，进入调试界面。
- 方式②：点击 VSCode 侧边栏“运行”图标，出现运行栏（ ），点击“开始调试（ ）”图标即可启动调试，进入调试界面。

启动失败

启动调试时，VSCode 会将调试信息下载至芯片，成功下载后将进入调试模式，若失败则会在界面出现错误信息悬浮框，如下图所示：

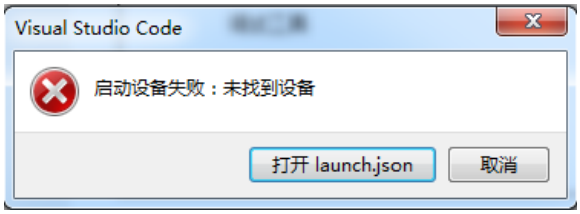


图 2-16 错误信息提示

调试程序

成功进入调试模式后，VSCode 在编辑界面提供了悬浮的调试工具栏，用户可以进行基本的调试操作，如下图所示：

图 2-17 调试工具栏

VSCode 在侧边栏的“运行”窗口提供了相关的调试窗口，如：变量、监视、调用的堆栈、断点、内核寄存器、特殊寄存器，其中，内核寄存器栏（CORE REGISTERS）和特殊寄存器栏（REGISTERS）仅在调试状态下显示在运行栏。如下图所示：

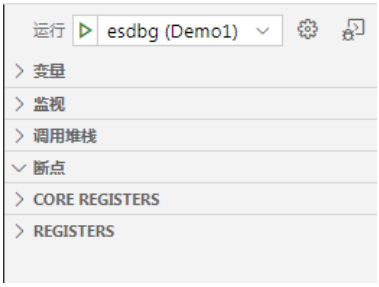


图 2-18 调试功能窗口

在调试运行到中断位置，右击主编辑窗口出现操作菜单，提供了调试操作的菜单选项，其中有：切换到反汇编单步、显示反汇编代码、显示内存窗口，如下图红色方框内所示：

图 2-19 调试功能子窗口

3 界面管理

3.1 起始界面

VSCode 在启动时会在主界面显示一个起始页面——“欢迎使用”页，关于此页的详细说明可查看官网资料（https://code.visualstudio.com/docs/getstarted/tips-and-tricks）。用户可以通过点击活动栏中“ESSEMI 扩展管理器”可以操作项目相关的功能，如下图所示。

图 3-1“欢迎使用”页

3.2 菜单栏

VSCode 在界面提供了一个菜单栏，如下图所示：

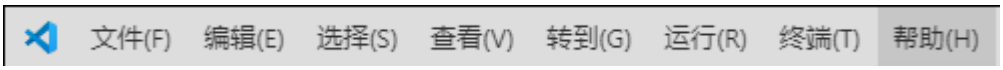


图 3-2 菜单栏

关于菜单栏的详细介绍及说明可查看 Visual Studio Code 官网资料 https://code.visualstudio.com/docs/editor/accessibility）。

在“文件”菜单中，“打开文件夹”、“打开工作区”、“打开最近的文件”仅作为以查看的方式打开文件或文件夹，如果进行项目相关操作则通过“ESSEMI 扩展管理器视图”中的相关操作来实现。

3.3 活动栏

活动栏位于 VSCode 最左侧，其从上到下依次为：资源管理器、搜索、源代码管理（git）、运

行（Debug）、扩展（使用插件），ESSEMI 扩展管理器，如下图所示：

图 3-3 活动栏

其中，用户可以通过“资源管理器（  ）”中的工作区查看、浏览项目所在文件夹的内容。通过“搜索（  ）”对整个工作区下的文件及内容进行筛选搜索。“运行（  ）”提供了启动调试的按钮，以及在运行状态下通过点击运行图标显示调试相关窗口：变量、监视、寄存器、断点等，如图 2-4 所示。

图 3-4 调试窗口

3.3.1 工作区

VSCode 在将项目加入到“ESSEMI 项目管理器”的同时也将项目所在文件夹加载到工作区中。用户可通过点击活动栏的“资源管理器”（  ），在工作区下查看当前所有项目所属的文件夹内容，

此工作区仅作为查看、浏览项目所在文件夹，不做其他操作。若用户删除了工作区中的项目文件夹，则将此项目同时从“ESSEMI 项目管理器”中删除。

关于工作区的相关操作及说明可参考 Visual Studio Code 的官网资料：https://code.visualstudio.com/docs/editor/multi-root-workspaces。

3.3.2 ESSEMI扩展管理器

“ESSEMI 扩展管理器”（）位于 VSCode 最左侧的活动栏中，如图 3-3 所示。用户可以通过“ESSEMI 扩展管理器”执行项目相关的操作，如下图所示。在“ESSEMI 扩展管理器”视图中，展开“快捷操作”一栏，其中，“项目操作”栏提供了新建项目、打开项目、添加项目的按钮；“通用功能”栏提供了导入芯片包、启动 ESBurner 的按钮。项目资源管理器一栏提供了项目的树形视图，在开发过程中，该视图可以帮助管理项目和文件，通过右击项目的不同节点来实现对项目的管理，其中包括：关闭项目、对项目的组和文件进行添加、移除和重命名以及编译项目或编译单个文件等操作。

图 3-5 ESSEMI 扩展管理器

3.4 工具栏

VSCode 工具栏提供了保存文件、跳转到上一导航、跳转到下一导航、切换注释、启动调试等常用按钮。如下图所示：

图 3-6 工具栏

注意：在调试状态下，此工具栏提供了切换到反汇编调试状态按钮。如下图红色标注所示：

图 3-7 调试状态下的工具栏

3.5 状态栏

VSCode 状态栏提供了查看问题和警告的按钮，通过这个按钮可以调出“问题”面板栏。关于状态栏的具体说明可查看 Visual Studio Code 官网资料：https://code.visualstudio.com/docs/getstarted/tips-and-tricks。

图 3-8 状态栏

3.6 设置界面

在活动栏的最下方点击设置图标（  ），在设置界面可进行一些常用设置，包括文本编辑器、工作台、窗口、功能、应用程序、扩展，设置内容为开发人员预先设置好的默认值，一般用户无需更改。如图所示。

用户通过各种设置可以轻松配置 VSCode，相关设置介绍可以在 Visual Studio Code 官网查看（https://code.visualstudio.com/docs/getstarted/settings）。

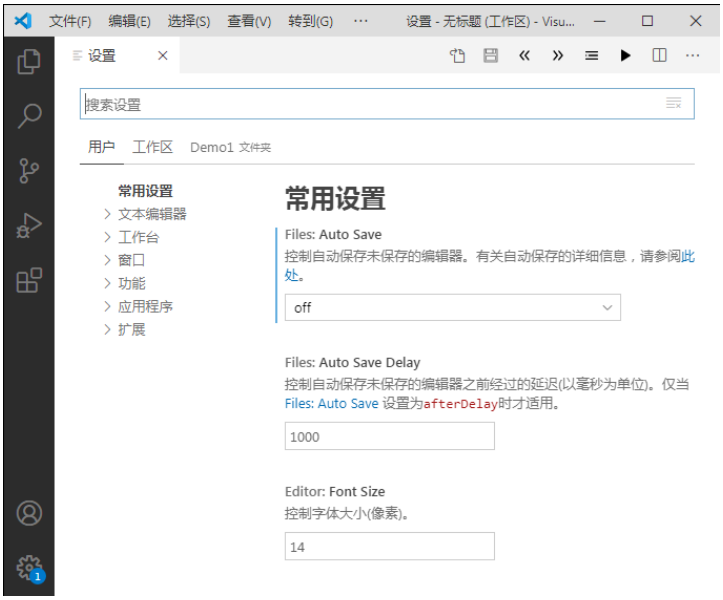


图 3-9 设置界面

4 资源管理

4.1 项目资源管理器

创建新项目时，VSCode 会自动将项目加入到“ESSEMI 项目管理器”中。然后用户可以根据需求对项目进行操作。在“ESSEMI 项目管理器”中，允许用户添加多个项目且对多个项目进行编译，但是启动调试时瞄定为当前编辑界面显示的文件所属的项目。

项目

项目通常包含了一组源文件，对于 32 位项目（Cortex-M）则必须包含相应的启动文件（.S），可执行项目的输出通常是.hex 文件，库项目的输出文件通常是.hrlib 文件。如图所示：

图 4-1 项目文件

4.2 项目类型

VSCode 支持三种芯片类型的项目，HR7P/ES7P/ES7P-V2 项目、Cortex-M 项目和 RISC-V 项目。

HR7P/ES7P/ES7P-V2 项目

HR7P/ES7P/ES7P-V2 项目即 8 位项目，支持扩展名为.c 的源文件，及扩展名为.h 的头文件，项目中至少要有一个.c 文件。

Cortex-M 项目

Cortex-M 项目即 32 位项目，支持扩展名为.c 和.S 的源文件，及扩展名为.h 的头文件，项目中至少要有一个.c 文件。Cortex-M 项目需要一个启动文件，启动文件中主要描述芯片的项目表处理方法和堆栈信息设置等内容。本地个人目录的.vscode 文件夹下会提供每款芯片的启动文件示例，用户可以根据实际情况加以修改，但建议不要修改中断向量表的排列。

RISC-V 项目

RISC-V 项目支持扩展名为.c 和.S 的源文件，及扩展名为.h 的头文件，项目中至少要有一个.c 文件。RISC-V 项目需要一个启动文件，启动文件中主要描述芯片的项目表处理方法和堆栈信息设置等内容。本地个人目录的.vscode 文件夹下会提供每款芯片的启动文件示例，用户可以根据实际情况加以修改，但建议不要修改中断向量表的排列。

4.3 项目管理

4.3.1 新建项目

VSCode 为用户提供了三个项目类型选项，用户可选择相应的项目类型建立基本的项目内容及源文件。

点击 VSCode 活动栏“ESSEMI 扩展管理器”图标，在出现的视图中，展开“快捷操作”一栏，然后点击“新建项目”按钮，在弹出的“选择文件夹”对话框中选择或输入项目文件夹，单击“创建项目”按钮，在界面出现的项目名称输入框中输入项目名称并回车，则在界面出现的芯片类型选择框中选择一种项目类型，如下图 4-2 所示。在跳出对应的芯片列表中选择一种芯片，如下图 4-3 所示，即可成功创建新的项目。

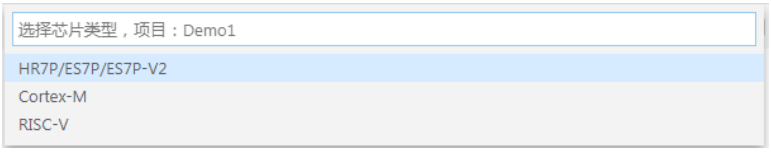


图 4-2“芯片类型”选择框

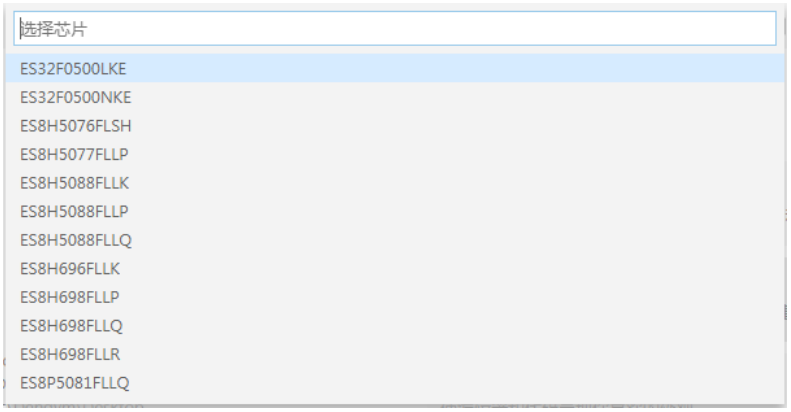


图 4-3“芯片列表”下拉框

注意：如果“ESSEMI 项目管理器”中已存在其他项目，则先关闭当前所有项目，再将新建项目加载到“项目资源管理器”中，同时，也会关闭工作区当前所有项目文件夹，再将新建项目所属的文件夹加载到工作区中。

4.3.2 添加现有项目

可以将现有的项目加入到“ESSEMI 项目管理器”中。

在“ESSEMI 扩展管理器”视图中，可通过“打开项目”和“添加项目”的操作将现有的项目加入到“ESSEMI 项目管理器”中。

打开项目

VSCode 支持直接打开 ESSEMI（.escfg）项目，以及将 iDesigner（.hrccproj）、Keil（.uvproj）等项目转换为 VSCode 所支持的项目打开，即通过“打开项目”操作直接将项目加载到“ESSEMI 项目管理器”中。单击 VSCode 活动栏的“ESSEMI 扩展管理器”图标，然后单击“打开项目”，

在出现的“打开”对话框中，选择或输入 ESSEMI（.escfg）文件，若要打开其他类型项目（如：Idesigner 项目、Keil 项目等）则下拉文件过滤框，切换到其他类型项目栏，如下图所示，单击“选择项目文件”按钮，则所选项目成功加载到“ESSEMI 项目管理器”中。如下图所示：

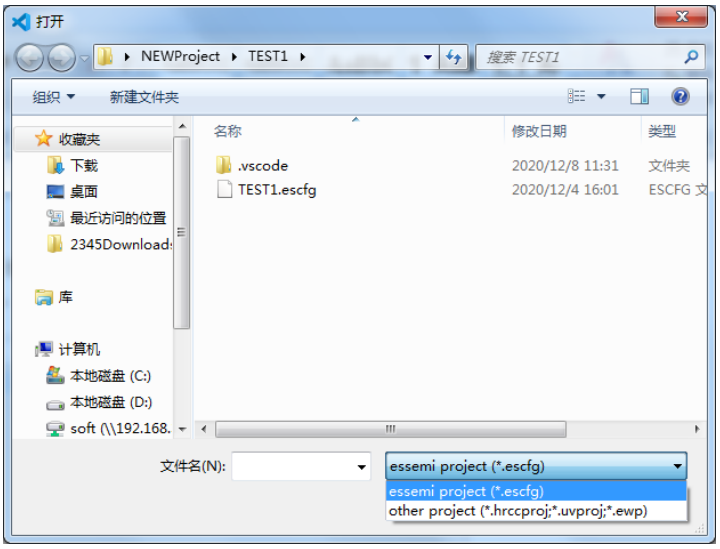


图 4-4“打开项目”对话框

注意：如果“ESSEMI 项目管理器”中已存在其他项目，则先关闭当前所有项目，再把将要打开的项目加载到“项目资源管理器”中，同时，也会关闭工作区当前所有项目文件夹，再将打开项目所属的文件夹加载到工作区中。

打开最近项目

VSCode 支持直接打开历史项目，即通过“最近项目列表”操作，显示最近项目列表将历史项目直接加载到“ESSEMI 项目管理器”中。单击 VSCode 活动栏的“ESSEMI 扩展管理器”图标，然后单击“最近项目列表”，在界面出现最近项目列表，如下图所示。可单击选择要打开的项目，则所选项目成功加载到“ESSEMI 项目管理器”中。

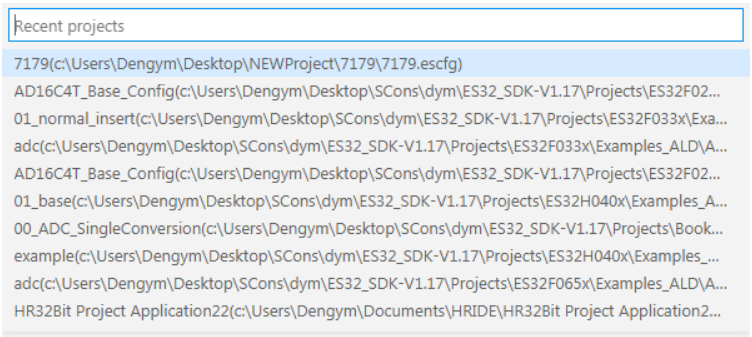


图 4-5 最近项目列表

注意：最近项目列表数量最多显示为：10。

添加项目

“项目资源管理器”支持添加多个项目。展开“ESSEMI 扩展管理器”视图中的“快捷操作”一栏，单击“添加项目”按钮，在出现的“打开”对话框中选择或输入项目文件，如下图所示，单

击“选择项目文件”按钮，则成功将项目添加到“ESSEMI 项目管理器”中，且将项目所在文件夹添加到工作区中。

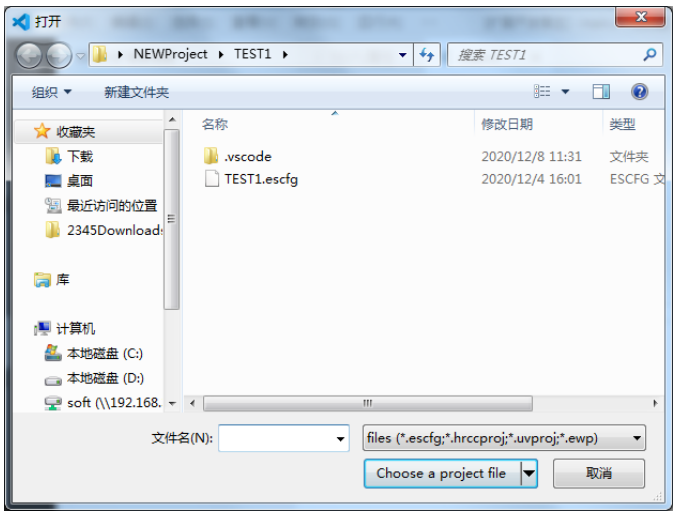


图 4-6“添加项目”对话框

4.4 管理项目资源管理器

4.4.1 分组

VSCode 在项目之下有一个容器，分组，用于管理项目存放文件。

分组是一个虚拟的概念，不存在于 Windows 资源管理器中。在分组中添加现有文件时，不会复制该文件，而是指向该文件的原路径。VSCode 支持在项目下循环创建分组，即在分组中可以创建子组，用户在“ESSEMI 项目管理器”中选择目标项目根节点或分组节点，在出现的操作菜单中选择“新建分组”来创建新的分组。

需要注意的是新建分组的名称不能和项目中的所有其他项重名，包括文件名。

4.4.2 向项目中添加现有项

可将已存在的文件直接加入到现有的项目中。

在“ESSEMI 项目管理器”中右击目标项目节点，从快捷菜单栏上选择“添加文件”，如下图所示，在弹出的“打开”对话框中定位并选择需要添加的项目项，单击“打开”按钮将该项添加到项目中。

图 4-7 项目操作菜单

注意：任何扩展名的文件都可以添加进项目，但是只有以.c、.asm 或.hrlib 为扩展名的文件才会参与编译。添加项目中已存在的文件，则不会再添加且在界面显示提示信息。如图所示：

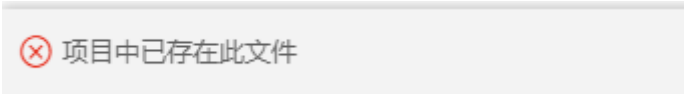


图 4-8 添加已存在项目提示信息

4.4.3 从项目中排除

在“ESSEMI 项目管理器”中右击需要排除的项，在出现的快捷菜单栏中选择相应的操作，其中，若选中的为项目根节点则选择“关闭项目”，若选择的为分组则选择“删除分组”，若选择的为文件则选择“删除文件”。

该操作只会将项从项目中删除，并不会真正删除相应的物理文件，若用户想要删除文件，必须在 Windows 资源管理器中手动删除。

4.4.4 重命名项

在“ESSEMI 项目管理器”中右击需要重命名的目标，在出现的快捷菜单中选择“重命名”，键入新的名称，即可完成重命名操作。

需要注意的是，在一个项目中的所有项不能重名，包括不同分组下的项也不能重名。

5 编辑功能

5.1 代码编辑

VSCode 提供了丰富的代码编辑功能，如：选择、更改文本和代码，查找和替换，高亮等。关于编辑功能的详细介绍可参考 Visual Studio Code 官方网站（https://code.visualstudio.com/docs/editor/codebasics）。

5.2 智能编码辅助功能

VSCode 具有高效、强大的语言支持，功能、效率、准确度相较于 iDesigner 有很大的提升，给用户提供智能编码辅助功能。

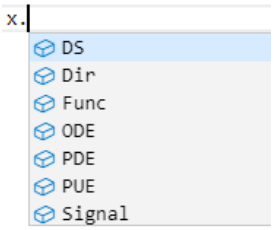
5.2.1 代码补全

“代码补全”功能会显示当前区域的有效成员，包括全局变量、局部变量、函数等。

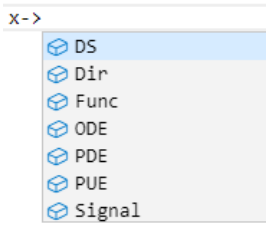
当用户在键入字符时，会弹出所有可能的补全建议，若有一个或多个匹配项，将自动选中第一个匹配项，选择列表中的某个成员，按 Tab 或 Enter 可以将该成员插入到代码中。按 Esc 快捷键可以关闭成员列表。

结构体、联合体的补全

“代码补全”功能还可以列出结构体及联合体的成员。在结构体或联合体的名称后键入“.”，或者在结构体类型的指针后键入“->”，补全列表会自动将其成员列出。如下图所示：



（a）结构体的成员变量



（b）结构体指针的成员变量

图 5-1 结构体补全

5.2.2 诊断提示

“诊断提示”功能是一种指示代码问题的方法。当用户代码编辑有问题时，则在有问题的代码下划红色曲线标注。鼠标悬停在有错误的地方则会提示相关的诊断信息。如下图所示：

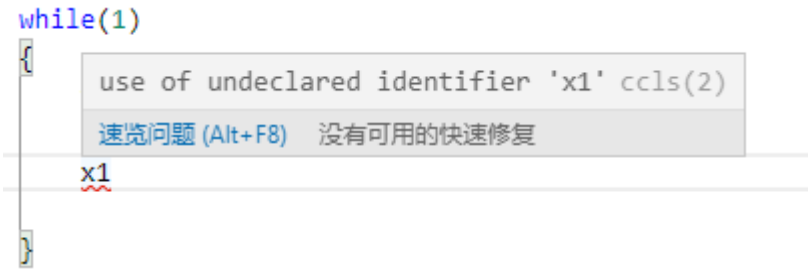


图 5-2 诊断提示

用户也能通过在“问题”窗口里查看相关的错误提示信息，“问题”窗口中包含所有项目当前错误的列表，用户可以单击错误信息则编辑界面跳转到此位置。如下图所示：

图 5-3 错误提示

5.2.3 悬停提示

“悬停提示”可为代码中的任意标识符显示完整的声明。

移动鼠标以使光标位于标识符上时，将会出现一个弹出框，其中显示了该标识符的快速信息，通常为标识符的类型和说明。从列出成员框中选择成员时也会出现快速信息。如下图所示：

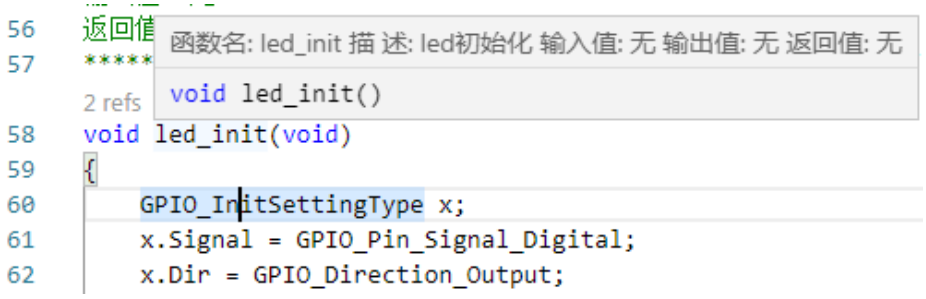


图 5-4 悬停提示

5.2.4 函数签名

用户输入方法或函数时，会出现一个弹出框显示有关正在调用的方法或函数的信息，包括返回值类型和参数列表等。如下图所示：

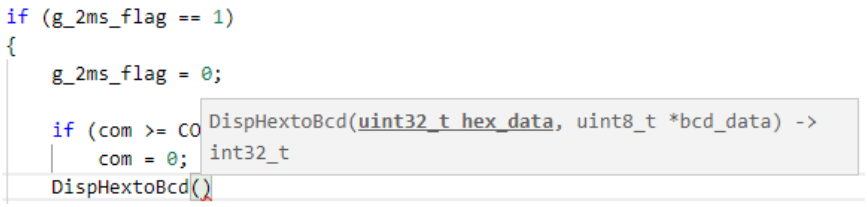


图 5-5 函数签名

5.2.5 查找所有引用

“查找所有引用”功能可以找到项目中所有引用到所选符号的位置。右击目标符号，在快捷菜单中选择“Find All References”或使用键盘快捷键“Shift+Alt+F12”，查找的结果将在侧边栏“REFERENCES RESULTS”窗口中显示，且显示引用的文件及结果数量。如下图所示：

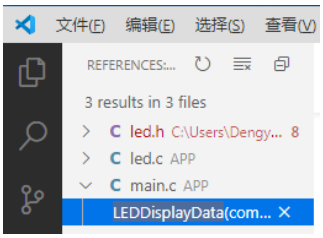


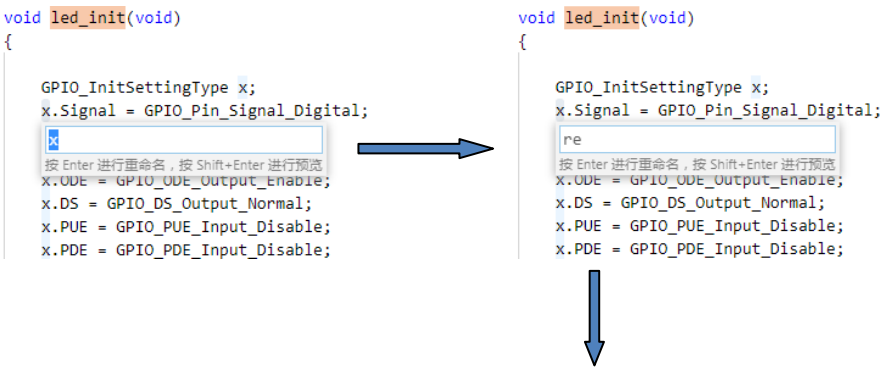
图 5-6 查找引用结果

结果以文件对应查找到的符号展示，用户可以通过展开文件查看该文件中引用的位置，双击子项，代码编辑器将跳转到该项的位置。

5.2.6 符号重命名

“符号重命名”功能允许用户重命名符号并更新对该符号的所有引用。

右击目标符号，在出现的快捷菜单中选择“重命名符号”或使用键盘快捷键 F2，在此目标符号下方出现的输入框里输入新的符号名称，则该符号以及该符号的所有引用都将更新为新名称。按 Esc 可以退出重命名。如下图所示：



```
void led_init(void)
{

    GPIO_InitSettingType re;
    re.Dir = GPIO_Direction_Output;
    re.Func = GPIO_Reuse_Func0;
    re.ODE = GPIO_ODE_Output_Enable;
    re.DS = GPIO_DS_Output_Normal;
    re.PUE = GPIO_PUE_Input_Disable;
    re.PDE = GPIO_PDE_Input_Disable;
```

图 5-7 符号重命名步骤

5.2.7 显示文件所有符号

用户可以通过在“大纲”中查看显示当前编辑页面的所有符号。

打开“资源管理器”一栏，展开“大纲”一栏显示当前活动文档的所有符号，单击任意符号则光标跳转到此符号位置。如下图所示：

图 5-8“大纲”列表

5.2.8 CodeLens

“CodeLens”功能向用户提供显示在源代码中的可操作的上下文信息。在变量、函数等符号的源代码位置上方显示可操作的上下文信息，显示形式为：xx refs，单击可操作的上下文信息，则在界面显示此符号在项目中的所有引用位置，用户可单击符号查看指定的符号引用位置。如下图所示：

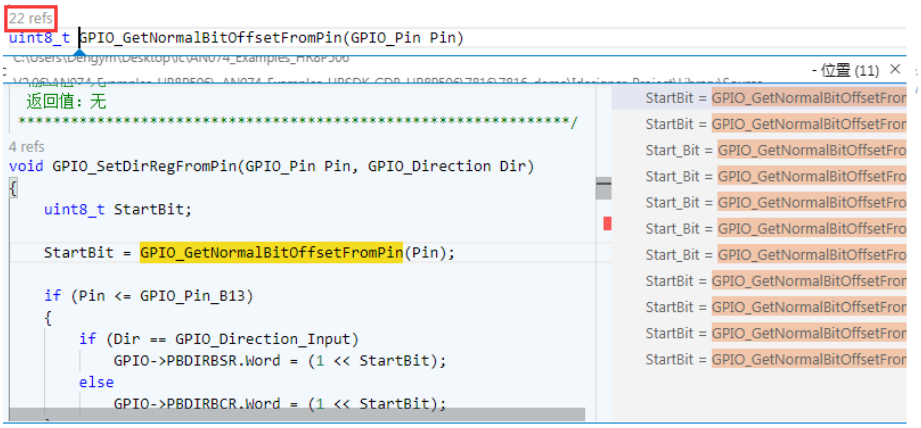


图 5-9CodeLens

用户可以通过“设置”->“扩展”->“esls”中的“Code Lens：Enabled”关闭该功能。如下图所示：

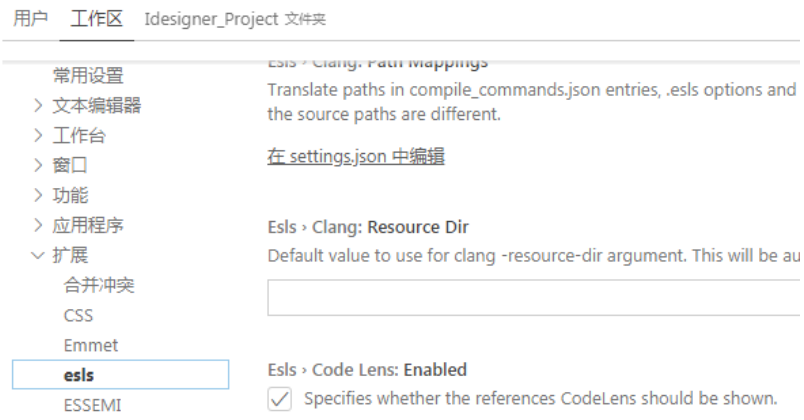


图 5-10 禁用 CodeLens 功能

5.2.9 格式化代码

VSCode 对源代码格式有很大的支持，“格式化代码”功能向用户提供了两个明确的格式操作：

- 格式化文档—对当前活动文档整个内容按照一定格式进行对齐等格式化操作。
- 格式化选定内容—将所选的代码片段按照一定格式进行对齐等格式化操作。

格式化文档

用户打开需要格式化的项目文件，在当前编辑界面任意位置右击，在出现的操作菜单中选择“格式化文档”，则当前活动文档的内容按照一定的格式进行缩进、对齐等格式化。

格式化选定内容

用户选定需要格式化的代码片段，然后右击，在出现的快捷菜单中选择“格式化选定内容”，则选定的代码段将按照一定的格式进行缩进、对齐等格式化。如下图所示：

图 5-11 格式化选定内容

5. 2. 10 文件比较

“文件比较”功能支持用户对选中的文件进行对比。在“ESSEMI 项目管理器”中右击目标项目文件，在出现的快捷菜单中选择“选择以进行比较”，然后右击选中与目标文件进行对比的文件，在出现的菜单栏中选择“与已选择文件比较”，则在主编辑界面显示两个文件的对比结果，将文件不同之处进行高亮标注，如下图所示。

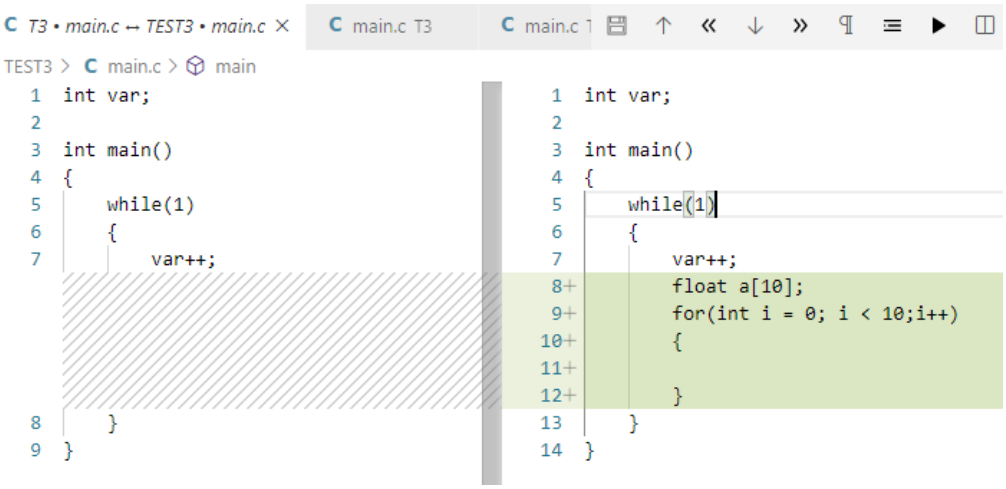


图 5-12 文件对比

6 编译生成

6.1 安装工具链

VSCode 根据用户所选项目类型，支持相应的不同版本的工具链：

- HR7P/ES7P/ES7P-V2 项目需要使用 HRCC 工具链编译
- Cortex-M/RISC-V 项目需要使用 ES32CC 工具链或 GCC 工具链编译。

6.2 工具链设置

新建项目时，VSCode 会根据项目类型使用默认的工具链编译该项目。若用户需要切换其他版本的工具链，在该项目的任意位置右击，在出现的菜单栏中选择“属性”，编辑界面出现项目的属性界面，将选项卡切换到“编译选项”页，则可在工具链一栏中设置对应项目的工具链，且工具链更改选择之后需要手动点击“保存编译配置”按钮。如下图所示：

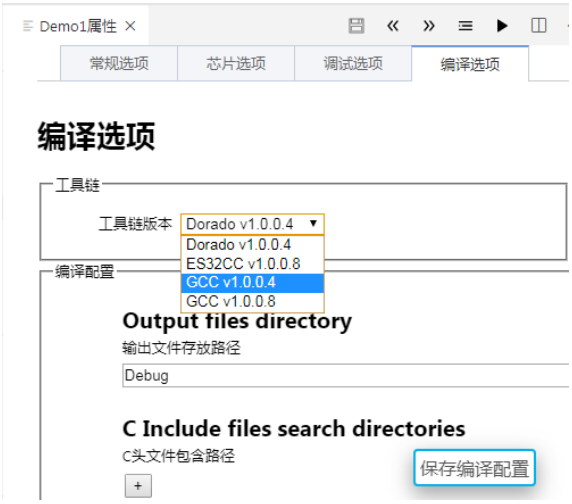


图 6-1 工具链选择界面

6.3 编译属性

VSCode 在生成项目之前需要先设置项目的编译属性，如果用户没有设置，VSCode 将使用默认的配置生成。

设置编译属性的操作步骤如下：

1. 在项目任意位置右击；
2. 在快捷菜单栏中选择“属性”；
3. 将项目属性页切换到“编译选项”；
4. 在“编译配置”一栏中设置编译属性；
5. 单击“保存编译配置”按钮。

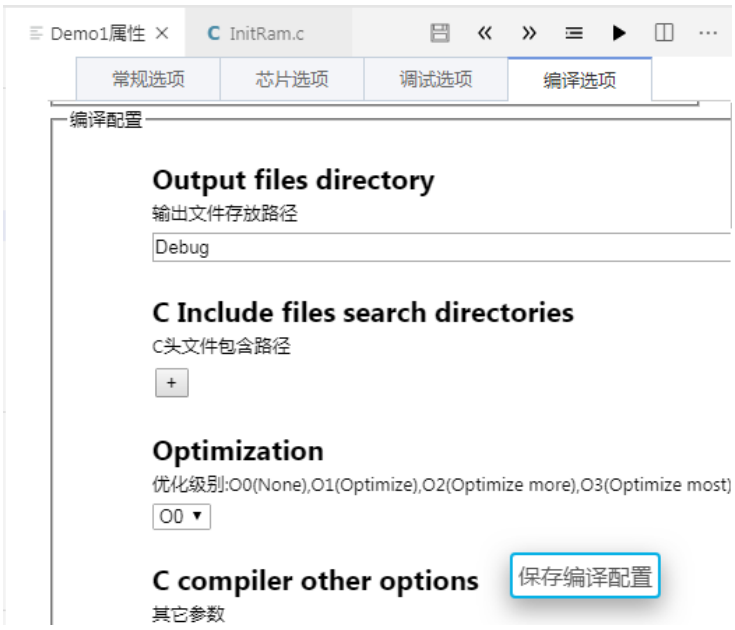


图 6-2 编译属性

HR7P/ES7P/ES7P-V2 项目支持以下编译属性：

- Preprocess Switch，预处理器版本开关，设置为 True 时使用旧版本预处理器，设置为 False 时使用新版本预处理器。
- Include files Search directories，用户头文件搜索路径，若用户的头文件与源文件不在同一目录，必须要设置搜索路径，若没有设置则使用默认路径。不支持中文路径。
- Predefinition macro，预定义宏。
- Optimization，优化级别，O0：常规优化，O1：指令优化。
- Support interrupt vectors，中断向量编译模式开关，设置为 True 时启用中断向量编译模式，设置为 False 时禁用中断向量编译模式。
- Pack const array，Const 数组压缩开关，设置为 True 时启用 Const 数组压缩，设置为 False 时禁用 Const 数组压缩。
- C compiler Switch，编译器版本开关，设置为 True 时使用旧版编译器，设置为 False 时使用新版编译器。
- Hrcc other options，Hrcc 命令的额外参数。
- IIC slave high speed mode，IIC 从机高速模式，设置 True 时使能 IIC 从机高速模式，设置 False 时关闭 IIC 从机高速模式。
- Link stdlib library，设置是否链入 stdlib 库，仅支持 79S 指令集芯片。
- Link maths library， 设置是否链入 maths 库，仅支持 79S 指令集芯片。
- Remove the dead variables in HRCO，设置是否在 HRCO 阶段进行全局变量分配优化。
- Hrco other options，Hrco 命令的额外参数。

- MSEC，控制存储分配策略严格一致，开启时可保证同一项目在不同环境下编译结果一致。
- Hasm other options，Hasm 命令的额外参数。
- Output hlink map file，设置 Hlink 命令是否输出 map 文件。
- Output call graph，函数调用关系，设置为 True 时输出函数调用关系，设置为 False 时不输出函数调用关系。
- Optimization for interrupt service，处理中断保护与恢复的方式，设置为 True 时以优化方式，设置为 False 时以默认方式。
- Hlink other options，Hlink 命令的额外参数。

Cortex-M 项目支持以下编译属性：

- Output files directory，输出文件存放路径，输出文件存放路径，过程文件及最终生成的 Hex 文件都存放在该目录，默认为项目路径下的 Debug 文件夹。
- C Include files search directories，C 文件用户头文件搜索路径，若用户的头文件与源文件不在同一目录，必须要设置搜索路径，若没有设置则使用默认路径。不支持中文路径。
- Optimization，优化级别。
- C compiler other options，C 编译命令的额外参数。
- Defined Symbols，C 宏定义。
- Undefined Symbols，取消 C 宏定义。
- Use user defined LD file，使用自定义 LD 文件。
- User library directories，用户库目录。
- LD file,Linker 脚本文件路径。用户可以选择自己编写的 linker 脚本进行编译而不使用系统默认生成的脚本。
- Do not use default libraries，不使用默认库。
- No standard libs，不使用标准库。
- Optimize libraries，使用优化库。
- Linker other optins，Linker 命令的额外参数。
- ASM Include files search directories，汇编文件用户头文件搜索路径，若用户的头文件与源文件不在同一个目录，必须要设置搜索路径。
- Defined ASM Symbols，汇编宏定义。
- Undefined ASM Symbols，取消汇编宏定义。

RISC-V 项目支持以下编译属性：

- Output files directory，输出文件存放路径，输出文件存放路径，过程文件及最终生成的 Hex 文件都存放在该目录，默认为项目路径下的 Debug 文件夹。
- C Include files search directories，C 文件用户头文件搜索路径，若用户的头文件与源文件不在同一目录，必须要设置搜索路径，若没有设置则使用默认路径。不支持中文路径。
- Optimization，优化级别。
- C compiler other options，C 编译命令的额外参数。
- Defined Symbols，C 宏定义。
- Undefined Symbols,取消 C 宏定义。
- Use user defined LD file，使用自定义 LD 文件。
- User library directories，用户库目录。
- LD file，Linker 脚本文件路径。用户可以选择自己编写的 linker 脚本进行编译而不使用系统默认生成的脚本。
- Do not use default libraries，不使用默认库。
- No standard libs，不使用标准库。
- Optimize libraries，使用优化库。
- Linker other options，Linker 命令的额外参数。
- ASM Include files search directories，汇编文件用户头文件搜索路径，若用户的头文件与源文件不在同一个目录，必须要设置搜索路径。
- Defined ASM Symbols，汇编宏定义。
- Undefined ASM Symbols，取消汇编宏定义。

相对路径

头文件搜索路径支持相对路径，相对路径指的是相对于项目文件夹的路径，没有根盘符的路径都作为相对路径处理。

相对路径有以下表示方式：

- .\：表示当前目录。
- ..\：表示上一级目录、

例如：项目文件为：D:\project\Demo\demo.escfg，则相对路径.\lib 表示 D:\project\Demo\lib，..\lib 表示 D:\project\lib。

6.4 编译命令

6.4.1 编译项目

“编译项目”操作将会编译目标项目，将中间文件及最终的 Hex 文件或库文件保存在编译选项中的输出路径中。

在“ESSEMI 项目管理器”窗口中右击需要编译的项目，从快捷菜单中选择“编译项目”；或者打开需要编译的项目文件，在当前活动文档界面右击，从快捷菜单中选择“编译项目”，即可执行“编译项目”操作。

“编译项目”操作并不会每次都编译所有的源文件，VSCode 会将上一次编译生成的结果与当前的源文件版本进行比较，仅对存在差异的文件进行编译，以减少编译所花时间。

6.4.2 重新编译项目

“重新编译项目”操作将会重新编译目标项目。

在“ESSEMI 项目管理器”窗口中右击需要重新编译的项目，从快捷菜单中选择“重新编译项目”；或者打开需要重新编译的项目文件，在当前活动文档界面右击，从快捷菜单中选择“重新编译项目”，将执行“重新编译项目”操作。

与“编译项目”操作不同的是，“重新编译项目”操作将完全编译所有的源文件。

建议，当用户修改少量源文件的代码时，执行“编译项目”操作可以节省大量编译时间；当用户添加、删除或重命名源文件时，执行“重新编译项目”操作可以确保编译的准确性。

注意，添加、删除、重命名.h 文件后，建议用户不要使用“编译项目”功能，而是使用“重新编译项目”，否则编译结果可能不正确。

6.4.3 全部编译

VSCode 支持用户对“ESSEMI 项目管理器”下的所有项目全部进行编译，将中间文件及最终的 Hex 文件或库文件保存在各个项目的编译选项中的输出路径中。

在“ESSEMI 项目管理器”窗口中任意位置右击，从快捷菜单中选择“全部编译”，即可执行“全部编译”操作。

“全部编译”操作同“编译项目”一样，不会对每个项目的所有源文件都进行编译，VSCode 会将各个项目的上一次生成的结果与当前的源文件版本进行比较，仅对存在差异的文件进行编译，以减少编译所花时间。

6.4.4 全部重新编译

“全部重新编译”操作将会重新编译“ESSEMI 项目管理器”下的所有项目。

在“ESSEMI 项目管理器”窗口中任意位置右击，从快捷菜单中选择“全部重新编译”，即可执行“全部重新编译”操作。

与“重新编译”操作一样，会将“ESSEMI 项目管理器”下的所有项目的源文件完全编译。

建议，当“ESSEMI 项目管理器”下多个项目时，当用户仅修改了其中的某个项目文件时，单独对此项目执行“编译项目”操作可以节省大量编译时间；当用户对“ESSEMI 项目管理器”下的

多个项目进行修改，执行“全部重新编译”操作可确保编译的准确性。

6.4.5 编译文件

VSCode 支持用户对项目的单个文件进行编译，即部分编译。右击项目的任意源文件，在快捷菜单中选择“编译文件”即可执行“编译文件”操作。

6.4.6 清理

“清理”操作将会删除目标项目生成过程中产生的临时文件，目标项目编译产生的文件默认保存在项目文件夹的 Debug 目录下。

在“项目资源管理器”窗口中右击需要清理的项目节点，在快捷菜单中选择“清理输出目录”，即可执行“清理”操作，其中，32 位项目保留 hex、elf 文件，8 位项目保留 hex、idb 文件。

6.5 编译结果

6.5.1 输出窗口

执行“编译项目”、“重新编译项目”、“全部编译”、“全部重新编译”、“编译文件”操作后，“输出”窗口将自动打开，并显示执行过程和结果。如下图所示：

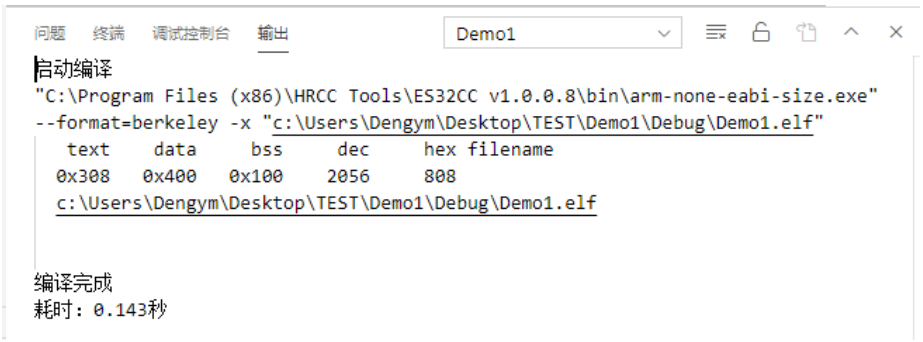


图 6-3“输出”窗口-编译成功

编译失败时，在输出窗口显示“编译出错”，如图所示：

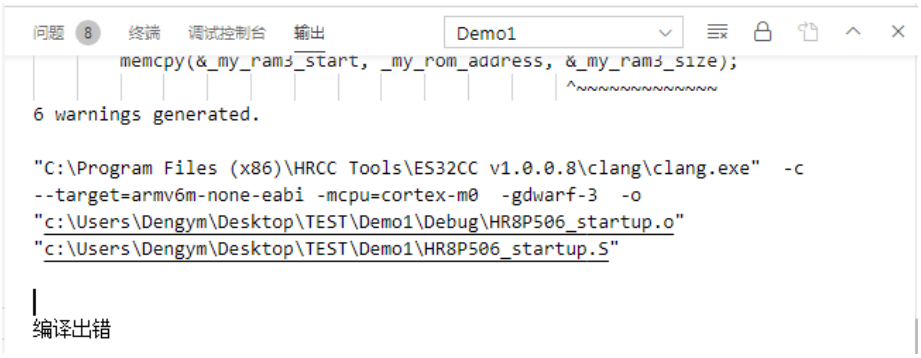


图 6-4“输出”窗口-编译失败

在该窗口中，用户可以查看编译生成的结果、花费的时间。

按住“Ctrl”按键且单击输出信息里的文件路径，即编辑窗口跳转到此文件。

在多项目编译时，点击窗口上方的下拉框可切换查看任意编译项目的编译结果。

6.5.2 问题窗口

编译相关操作产生的错误信息将会在“问题”窗口中显示，点击状态栏左侧边的“问题(⊗7 ⚠9)”图标，即可打开该窗口。如下图所示：

图 6-5“错误”窗口

在窗口上方工具栏中可以看到错误信息的数量。“错误”窗口中的信息分为两个级别：“错误”、“警告”，在错误项前会标注其错误级别，“错误”级别的标注符号为：⊗，“警告”级别的标注符号为：⚠，单击列表中的某一项，可以在源窗口中定位到信息指向的位置。

7 项目调试

7.1 调试工具

VSCode 支持使用 ES10M、ESLink2 设备调试 8 位项目，使用 ESLink2、JLink 设备调试 32 位项目，调试工具可以在项目属性中设置。

7.2 配置调试环境

调试项目需要配置相关的环境。配置步骤如下：

1. 连接硬件设备并上电，确认计算机与硬件设备连接成功。
2. 设置需要连接的通信口。
3. 设置项目的芯片类型。
4. 设置项目的芯片配置字。

7.2.1 设置调试工具

对于 32 位项目，右击项目节点，在快捷菜单中选择“属性”菜单，将主编辑界面出现的项目属性切换到“调试选项”页，从“调试工具”下拉列表中选择与硬件设备连接的调试工具类型，单击“保存调试配置”按钮，如下图所示：

图 7-1 设置调试工具

当只有一个指定类型调试工具与电脑连接时，VSCode 将直接使用该调试工具进行调试；当多个调试工具连接时，用户需要选择其一进行调试。

7.2.2 设置芯片类型

在调试过程中，项目的芯片类型需要与硬件设备相匹配。

右击项目的任意位置，在出现的快捷菜单栏中选择“属性”菜单，将主编辑窗口中出现的项目属性切换到“常规选项”页，在“选择芯片”一栏中可以设置启动项目的芯片，且更改芯片之后需手动点击“保存常规配置”按钮，保存成功则“当前芯片”栏显示更改后的芯片名称以及芯片列表选择框旁显示芯片的相关信息。如下图所示：

图 7-2 启动项目芯片选择

7.2.3 设置芯片配置字

启动调试之前还需要设置芯片配置字，其中，调试使能位（CFG_DEBUG 等，视芯片配置项名称可能不同）必须设置为 Enable 才可以进入调试模式。

在项目任意位置右击，选择快捷菜单栏的“属性”菜单，将主编辑窗口出现的项目属性切换到“芯片选项”页，在芯片配置一栏中可以设置芯片配置字，设置完成之后手动点击“保存芯片配置”按钮，即可完成芯片配置字的设置。如下图所示：

图 7-3 配置字界面

7.3 执行控制

7.3.1 开始调试

开始执行是最基本的调试功能之一。启动调试成功后，称调试器处于运行模式。

调试项目

在当前活动页上方栏选择“Debug”（）、在“运行”菜单中选择“启动调试”或在侧边栏选择“运行”按钮，调试器将开始调试。在“ESSEMI 项目管理器”下可能包含不止一个项目，这时候需要设置启动项目，VSCode 启动调试的项目为当前活动页面的文件所属的项目，即，打开需要启动调试的项目文件，即可指定此项目为启动项目。若 VSCode 的主编辑窗口没有打开任何文件直接启动调试，则界面会出现所有项目列表，用户需要手动点击选择需要调试的项目，如下图所示。

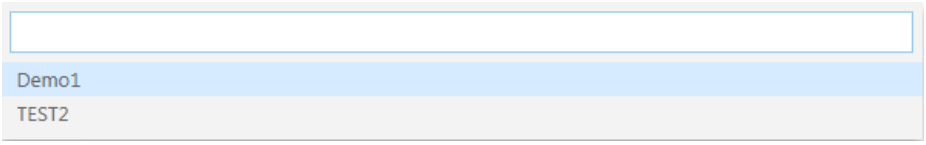


图 7-4 调试项目选择列表

7.3.2 停止调试

停止调试意味着终止调试会话，退出调试模式，请勿与暂停调试混淆。

停止调试

在“调试”工具栏中点击“停止调试”（）按钮，或在“运行”菜单中选择“停止调试”，即可停止当前调试的会话。

停止调试并重新启动

在“调试”工具栏中点击“重启”（）按钮，或在“运行”菜单中选择“重启调试”，即可停止当前正在调试的会话，并重新启动一个新的调试会话。

7.3.3 暂停调试

当执行到达一个断点时，调试器将暂停程序的执行。大部分的调试器功能（比如变量监视功能等）仅在暂停模式下可用。

除了到达一个断点外，调试器还可以手动暂停程序执行。在“调试”工具栏中点击“暂停”（）按钮，调试器将停止正在运行的程序，程序并不退出，调试器现处于暂停模式，即中断模式。随时可以恢复执行，用户也可以通过手动恢复执行，点击“调试”工具栏中的“继续”（）按钮，或在“运行”菜单中选择“继续”，则恢复执行。

7.3.4 代码单步执行

代码单步执行是最常见的调试过程之一，即每次执行一行代码。调试器提供了三个逐句执行代码的命令：“单步跳过”（）、“单步调试”（）、“单步跳出”（）。如果应用程序正在运行，则无法访问这些“单步执行”命令。“单步执行”命令只在暂停模式，即中断模式下有效。

“单步跳过”和“单步调试”的差异仅在于它们处理函数调用的方式不同。这两个命令都指示调试器执行下一行的代码。如果某一行包含函数调用，“单步跳过”执行整个函数，然后在函数外的第一行处停止。而“单步调试”仅执行调用本身，然后在函数内的第一个代码行处停止。如果要查看函数调用的内容，请使用“单步调试”。若要避免单步执行函数，请使用“单步跳过”。

位于函数调用的内部并想返回到调用函数时，请使用“单步跳出”。“单步跳出”将一直执行代码，直到函数返回，然后在调用函数中的返回点处中断。并不是所有芯片都支持“单步跳出”功能，具体请查询芯片的相关文档。

7.3.5 反汇编单步

用户可以对源代码对应的反汇编代码进行调试。主编辑界面上方的“反汇编”快捷按钮可切换“反汇编”和“源代码”两种调试状态。切换到“反汇编”调试状态则图标着色加深（ ），光标移到图标处显示悬浮信息为“切换到语句单步”，则标志着已进入反汇编调试状态，即可执行汇编单步相关操作。切换到“源代码”调试状态则图标着色变浅（ ），光标移到图标处显示悬浮信息为“切换到反汇编单步”，则标志着已进入源代码调试状态，即可执行源代码单步相关操作。当程序处于中断模式时，可通过以下方式执行“反汇编单步”的相关操作：

- 点击主编辑界面上方的切换调试状态的快捷按钮（ ），图标悬浮信息切换为“切换到语句单步”即标志进入反汇编状态，执行反汇编调试的相关操作，如：汇编单步，设置断点等。
  注意，退出反汇编状态进入“源代码”调试状态，再次点击主编辑界面上方的快捷按钮，图标切换为“切换到反汇编单步”即可。
- 在源代码界面，右击并从快捷菜单中选择“切换到反汇编单步”，且光标移到主编辑界面上方的切换调试状态快捷按钮则显示悬浮信息为“切换到语句单步”，即进入反汇编调试状态，执行汇编调试相关的操作如下图所示：

图 7-5“切换到反汇编单步”快捷键

注意，退出执行“反汇编”调试状态进入“源代码”调试状态，右击“汇编”窗口任意位置，在出现的快捷菜单中选择“切换到语句单步”，且光标移到主编辑界面上方的切换单步状态快捷按钮则显示悬浮信息为“切换到反汇编单步”，即进入到源代码调试状态，执行源代码调试相关的操作。

7.3.6 运行到指定位置

有时在调试过程中，用户想执行到代码中的某一行然后暂停，实现方式有以下几种：

- 在要中断的位置设置断点，在“调试”工具栏中点击“继续”（  ）按钮，或者在“运行”菜单选择“继续”，程序将运行到已设置断点的指定位置。
- 在源窗口或“反汇编”窗口中右击某行，并从快捷菜单中选择“运行到光标处”，调试器将运行程序，直至到达设置光标的位置。如果过程中遇到断点，则程序将提前中断。

7.3.7 复位

在悬浮的调试工具栏中选择“复位”（  ）按钮，可以将当前调试状态重置为刚启动调试时的状态。

7.4 断点控制

7.4.1 断点概述

断点通知调试器在执行到某行代码时中断程序并暂停执行，发生中断时，则称调试器处于中断模式，此时调试器停止在断点设置的指令上，断点所在指令未执行。

源窗口和“反汇编”窗口在左侧空白中显示有圆圈的标志符号，表示为此处设有断点。实心的标志符号（ ● ）指示断点已启用，空心的标志符号（ ○ ）指示断点已禁用。如果将鼠标光标停在断点标志符号上，将显示此断点状态的提示信息。断点禁用的原因可能有多种：

- 用户将该断点设置为“禁用”状态。
- 该断点所在行没有可执行代码。
- 已启用的断点数量超过调试芯片的限制，即，超过三个有效断点。

在源窗口中设置的断点将一直保留直到用户删除，而在“反汇编”窗口中设置的断点在停止调试后即会删除。

7.4.2 设置断点

设置断点可以通过以下方式进行：

- 在源窗口或“反汇编”窗口中，右击要设置断点的那行可执行代码，在快捷菜单中选择“添加断点”。
- 在源窗口或“反汇编”窗口中，单击要设置断点的那行可执行代码，在“运行”菜单中选择“切换断点”。
- 在源窗口或“反汇编”窗口中，单击要设置断点的那行可执行代码左侧空白处。

7.4.3 删除断点

删除断点可以通过以下方式进行：

- 在“运行”视图的“断点”一栏中，右击一个断点，并从快捷菜单中选择“删除断点”。如下图所示：

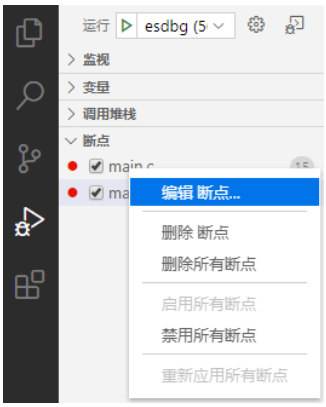


图 7-6 在“断点”视图中操作断点

- 在源窗口或“反汇编”窗口中，单击设置断点的那行可执行代码，从“运行”菜单中选择“切换断点”。
- 在源窗口或“反汇编”窗口中，单击断点标志符号。
- 在源窗口或“反汇编”窗口中，右击断点标志符号，并从快捷菜单中选择“删除断点”。

7.4.4 删除所有断点

删除所有断点可以通过以下方式进行：

- 在运行视图的“断点”一栏中，展开断点列表，点击“断点”栏的“删除所有断点”按钮。
- 在运行窗口视图的“断点”一栏中，展开断点列表，右击，并从快捷菜单中选择“删除所有断点”。
- 在源窗口或“反汇编”窗口中，从“运行”菜单中选择“删除所有断点”。

7.4.5 启用断点

启用断点可以通过以下方式进行：

- 在源窗口或“反汇编”窗口中，右击已禁用断点标志符号，并从快捷菜单中选择“启用断点”。
- 在源窗口或“反汇编”窗口中，单击已禁用断点标志符号。

7.4.6 禁用断点

在源窗口或“反汇编”窗口中，右击已启用断点标志符号，并从快捷菜单中选择“禁用断点”。

7.4.7 启用/禁用所有断点

启用/禁用所有断点可以通过以下方式进行：

- 从“运行”菜单中选择“启用所有断点”或“禁用所有断点”。
- 在“运行”视图的“断点”一栏中，展开断点列表，右击，并从快捷菜单中选择“启用所有断点”或“禁用所有断点”。

7.4.8 条件断点

8 位项目支持条件断点设置功能，条件断点窗口提供了丰富的中断条件设置。在运行过程中，调试器的状态一旦满足条件断点，就会中断运行状态。

右击项目任意节点，在出现的菜单栏中选择“条件断点”即可打开“条件断点”窗口，如下图所示：

图 7-7“条件断点”窗口

PC 断点使能

根据芯片特性不同，PC 断点支持断点地址个数不同，当在“配置字”栏中选择 PC 断点使能，用户参考章节《断点控制》介绍的方法设置断点时，调试器运行到断点所设置的地址会自动中断运行；当选择 PC 断点不使能时，调试器不会在断点所设置的地址自动中断。

PC 断点缺省为使能状态。

注意，此时调试器尚未执行断点所在指令。

A 寄存器断点使能

在“配置字”栏中选择 A 寄存器断点使能时，在 A 寄存器值与“A 寄存器”设置栏中用户输入的数据内容一致时，调试器会自动中断运行。

注意，此时调试器处于指令执行完成的地址上。

PSW 状态位断点使能

在“配置字”栏中选择 PSW 状态位断点使能时，在 PSW 寄存器内容与“PSW”设置栏中用户设置的 PSW 寄存器各个标志位使能状态和设置内容一致时，调试器会自动中断运行。

注意，此时调试器处于指令执行完成的地址上。

普通寄存器断点使能

设置普通寄存器条件断点的步骤如下：

1. 在“配置字”栏中，使能“普通寄存器断点使能”功能，并选择“类型”，普通寄存器断点设置类型分为如下几类：
   - 00：读寄存器断点，指定寄存器读操作。
   - 01：读寄存器匹配断点，指定寄存器读操作时读出值与预设值相等。
   - 10：写寄存器断点，指定寄存器写操作。
   - 11：写寄存器匹配断点，指定寄存器写操作时写入值与预设值相等。
2. 在“普通寄存器”->“地址”一栏中，设置普通寄存器地址。
3. 在“普通寄存器”->“数值”一栏中，设置普通寄存器预设数值。

在普通寄存器断点使能状态下，根据“普通寄存器断点设置类型”，在当前设置寄存器地址的读写状态或内容与预设内容匹配时，调试器会自动中断运行。

注意，此时调试器处于指令执行完成的地址上。

条件断点匹配次数设置

PC 断点支持计数匹配功能，可以通过“匹配次数”->“PC 指针”一栏设置，PC 第 n 次指向断点指令时调试器停止运行。N 为次数计数值，范围 1<<n<<1024.

普通寄存器条件断点的匹配次数是指当 PC 执行到断点处且满足条件的次数，可以通过“匹配次数”->“寄存器”一栏设置。

7. 4. 9 数据监视点

32 位项目支持数据监视点功能，在运行过程中，当指定内存被访问（读、写或两者都有）时，触发中断。

当调试器处于中断模式时，右击项目任意节点，在出现的菜单栏中选择“数据监视点”即可打开“数据监视点”窗口，如下图所示：

图 7-8“数据监视点”窗口

- 最多支持添加两个数据断点。
- 界面可设置地址长度，且地址长度以 size=2^n 对齐，n=0,1……15。
- 界面可设置访问方式，读、写或二者都有。

7.5 调试窗口

7.5.1 断点窗口

在“运行”视图的“断点”部分中可以查看当前“ESSEMI 项目管理器”中的项目中已设置的断点，因此在“ESSEMI 项目管理器”有多个项目时用户需要删除对非启动调试的项目的断点设置，否则会影响当前执行项目的调试状态。在“断点”栏中用户可以执行删除断点、切换断点状态、转到断点位置等操作，如下图所示：

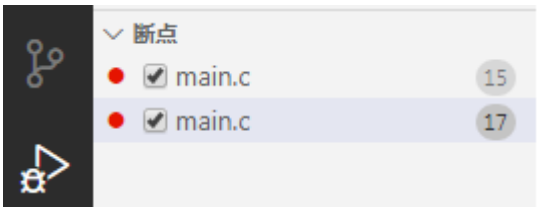


图 7-9“断点”视图

7.5.2 变量窗口

VSCode 调试器提供了“变量”视图和“监视”视图，它们在调试器处于中断模式时显示变量的信息。且 32 位项目支持切换十六进制显示数据，点击“监视”或“变量”一栏的“切换十六进制显示”按钮，即可将数据值切换为十六进制显示，如下图所示：

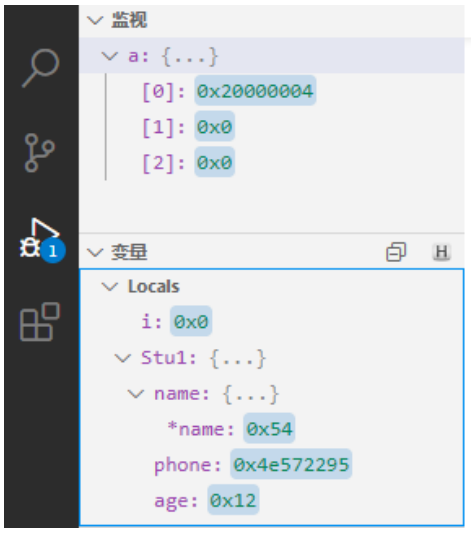


图 7-9 “变量”视图

在“运行”视图的“变量”部分可以检查变量，其中包括“Locals”、“Globals”和“Globals Statics”三种变量类型。用户可以同时展开这三种类型进行查看，且双击变量即可编辑变量的值。如下图所示：

图 7-10 “变量”视图

“Locals”视图、“Globals”和“Globals Static”中的变量分别是调试器自动添加的局部变量、全局变量和全局静态变量，用户不可修改，但可通过双击此变量在出现的输入栏编辑此变量的值。

在变量视图中编辑变量值的步骤如下：

1. 调试器必须处于中断模式。
2. 如果变量是数组或对象，则变量名称旁边将出现控件（ ）。如有必要，请单击展开变量，以找到要编辑其值的元素。
3. 在要更改的行中双击。
4. 键入新值，按“回车”确认。

在“监视”视图中用户可以通过以下方式添加自己的变量或表达式：

- 在“监视”视图中单击“添加表达式”控件（ + ），输入要监视的变量或表达式。
- 在源窗口中右击需要监视的变量，在快捷菜单中选择“添加到监视”。

VSCode 还提供了另外一种变量监视方式“数据提示”。当鼠标在变量上悬浮时，鼠标处会出现浮动提示，其中显示了该变量的成员和值。如下图所示：

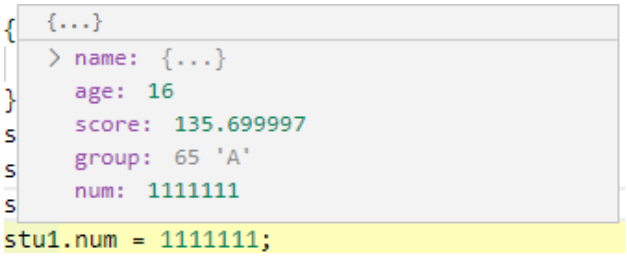


图 7-11 数据提示

7.5.3 调用堆栈窗口

“调用堆栈”视图记录了当前函数调用关系，展开“运行”视图的“调用堆栈”一栏，即可打开“调用堆栈”视图。如下图所示：

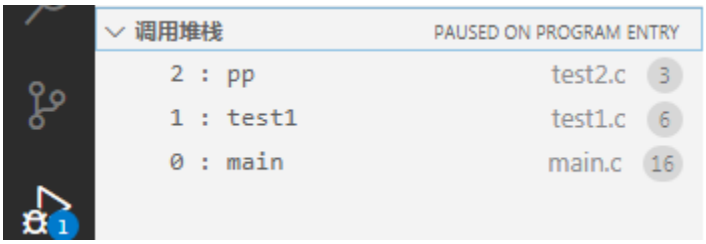


图 7-12“调用堆栈”视图

双击堆栈列表中的函数名称，可以在源窗口或“反汇编”窗口中跳转至该函数调用的位置。不同的芯片的堆栈级数限制不同，如果调用层数超过芯片的限制，程序将无法正常运行。

7.5.4 内核寄存器窗口

内核寄存器窗口显示了每次暂停时内核寄存器的值。

在“运行”视图中点击“CORE REGISTERS”一栏，在展开的列表中点击“core”即可打开查看内核寄存器的值。如下图所示：

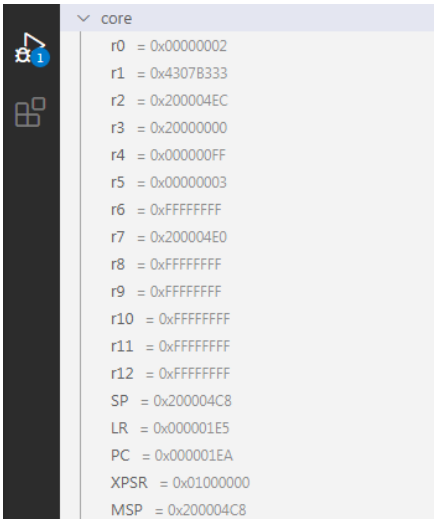


图 7-13 内核寄存器视图

32 位项目支持对内核寄存器的修改。右击寄存器的值，在出现的菜单栏中选择“设置寄存器值”，即可修改寄存器的值。需要注意的是，某些寄存器的值是只读的，修改后会恢复原值。

7.5.5 外设寄存器窗口

外设寄存器窗口显示了所选芯片的特殊功能寄存器的值，若某寄存器支持位功能，则点击其名称之前的控件符号（ ），即可展开查看支持的位。

在“运行”视图中点击“REGISTERS”一栏即可打开“REGISTERS”视图。如下图所示：

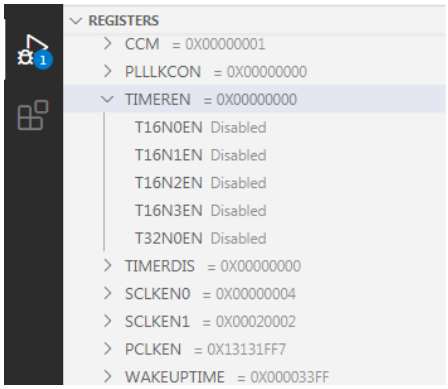


图 7-14 外设寄存器视图

右击寄存器的值，在出现的菜单栏中选择“设置寄存器值”，即可修改寄存器的值，位的值也支持修改。需要注意的是，某些寄存器或位的值是只读的，修改后会恢复原值。

7.5.6 内存窗口

“内存”窗口显示了芯片当前状态下数据存储器的值，仅在中断模式下有效。

查看内存窗口的操作步骤如下图所示：

1. 在源窗口或“反汇编”窗口中，右击并从快捷菜单中选择“显示内存窗口”。如图 6-4 所

示。

2. 在出现的输入框中输入想要查看的内存起始地址并回车，如下图所示。

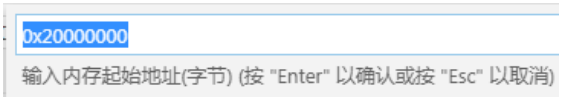


图 7-15 起始地址输入框

3. 再在出现的输入框内输入内存长度并回车，如下图所示。

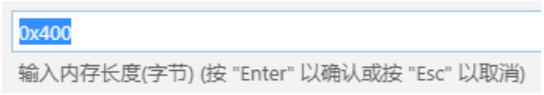


图 7-16 内存长度输入框

通过以上操作步骤，即可打开内存窗口，如下图所示：

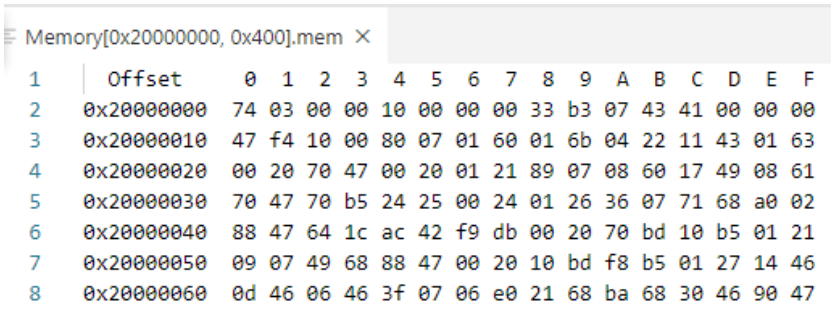


图 7-17“内存”窗口

在“Memory”窗口的上方菜单栏中有可以选择需要的数据显示格式的按钮，有“1”、“2”、“4”分别表示“显示 1 字节”、“显示 2 字节”、“显示 4 字节”，如下图所示：

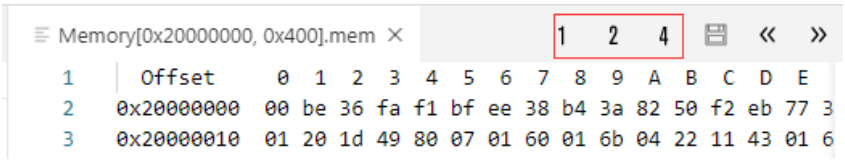


图 7-18“数据显示格式”按钮

7.5.7 反汇编窗口

在源窗口中，右击某一行源代码并从快捷菜单中选择“显示反汇编代码”，若该行源代码有对应的反汇编代码，即可跳转至对应的反汇编代码行，并在该行高亮标注，如下图所示：

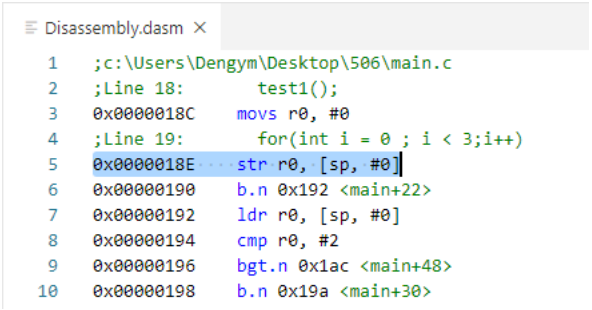


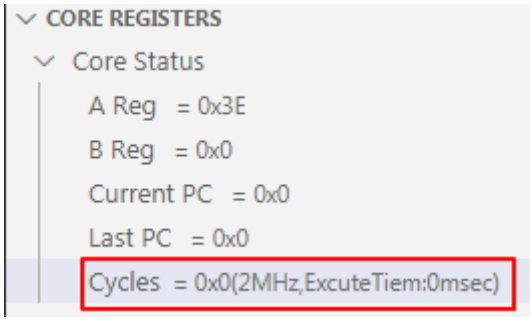
图 7-19“反汇编代码”窗口

7.5.8 跑表窗口

8 位项目支持跑表功能。

“跑表”窗口可以记录芯片执行调试命令花费的指令周期及时间。

在“运行”视图中点击展开“CORE REGISTERS”一栏，在寄存器“Cycles”一栏中，括号内所记录的数值即为每次执性调试命令花费的指令周期和时间。其中，指令周期默认为 2MHz，用户可以右击此栏在出现的菜单中选择设置晶振值，以及重置时间的操作。如下图所示：



7-20“跑表”窗口

“Cycles”栏显示的是执行单次调试命令的周期和时间，每次执行命令时都会重新计算。

7.6 可视化调试

可视化调试功能可在调试过程中输入变量名等进行可视化监视。在调试状态下，打开命令面板（Ctrl+Shift+P），输入“Debug Visualizer:New View”命令，打开一个 Debug Visualizer 窗口，在输入框里输入变量名称，即可在该窗口中以图表的形式监视变量。如下图所示。

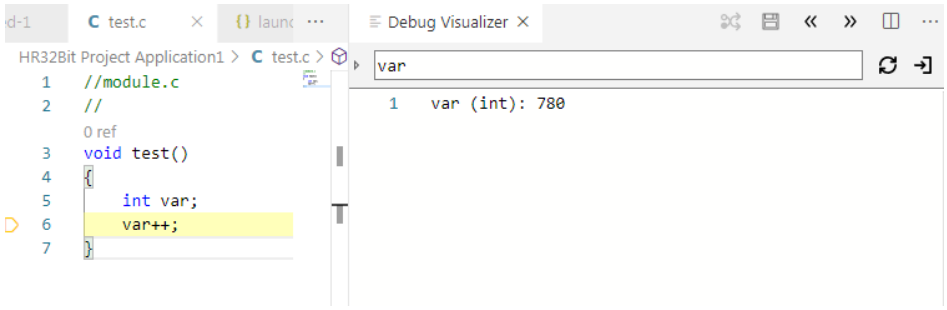


图 7-21 可视化调试窗口

可视化功能支持变量监视及查找指向变量的指针。由于需要分析的变量符号较多，调试性能可能会受到影响。

变量监视

在可视化调试窗口输入框里直接输入变量名称即可对此变量进行可视化监视。其中，对于监视的变量存在以下三种不同情况：

- 若变量是基础类型变量，如 int、float 等，直接显示名称、类型、值。
- 若变量是数组类型，则以表格方式显示。若数组元素是基础类型，显示元素值,如图 7-18 所示；若数组元素是复杂变量，则显示元素类型，如图 7-19 所示。
- 若变量是指针、结构体、联合体，以图的方式显示，展开变量所有成员。若遇到指针成员，会尝试分析所有局部变量、全局变量，找到指针指向的真实变量，并将其显示展开。

注意：8 位芯片不支持表达式计算，监视数组元素只支持常量下标。

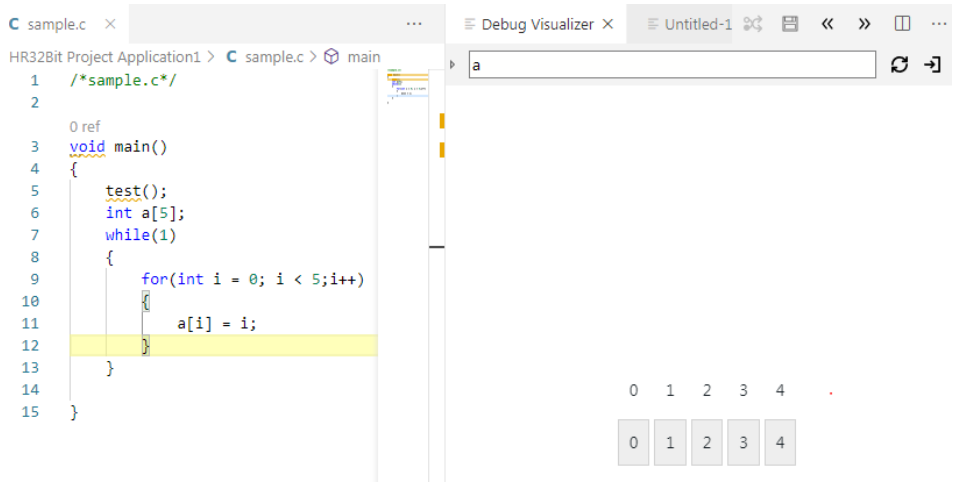


图 7-22 数组元素为简单变量-可视化监视

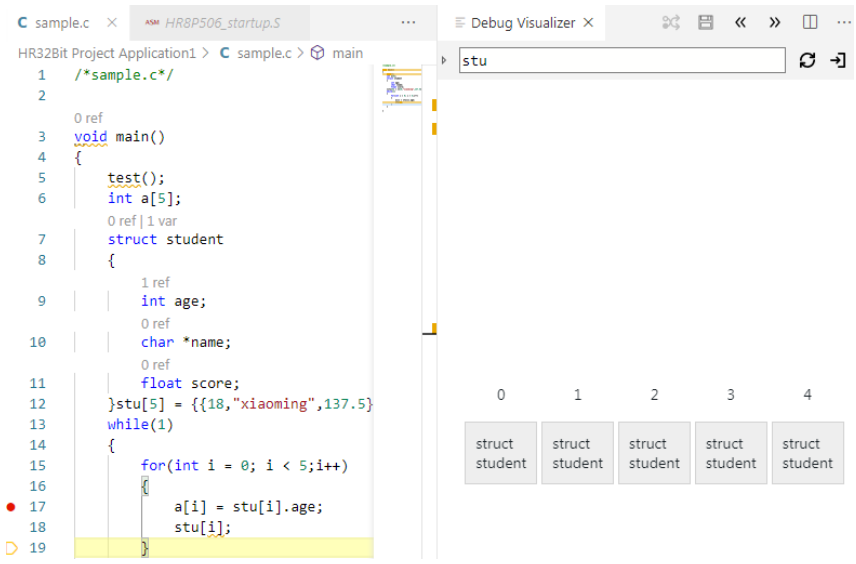


图 7-23 数组元素为复杂变量-可视化监视

查找指针

可视化调试窗口还支持查找所有指向某变量内存的指针变量，若变量是复杂变量，则展开变量成员（指针不进一步展开），并将指针指向具体成员。用户在打开可视化调试窗口后，在输入框里输入：ref(变量名)，即可显示变量并找到所有指向此变量内存的指针变量。如下图所示。

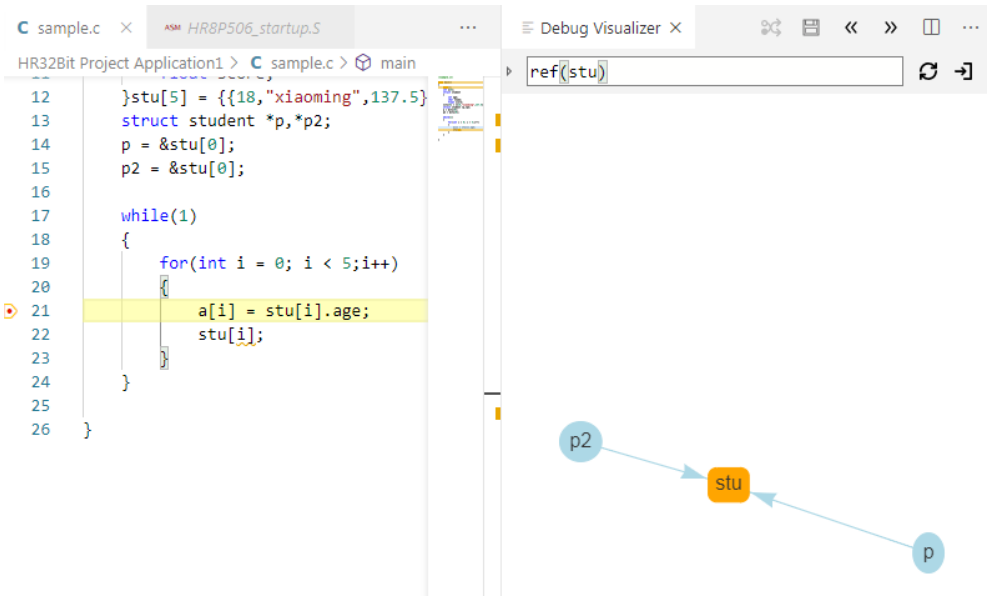


图 7-24 指向变量的指针-可视化监视

8 可视化配置

8.1 可视化配置机制

可视化配置头文件功能用于在文件中生成类似 GUI 的配置控件。使用源代码中的配置向导注释来生成控件。

用户打开头文件编辑配置向导注释，如下图 8-1 所示。关闭后，右击该节点文件，在出现的菜单栏中选择“可视化配置头文件”，即可根据配置向导注释内容显示此文件的可视化配置界面。如下图 8-2 所示。更改可视化配置界面的控件内容，会映射更改指定的数字或字符串。

ConfigWazid.h ×

ConfigWazid > ConfigWazid.h > Setup

```
1   //          <<<Use Configuration Wizard in Context Menu >>>
    0 ref
2   FUNC void Setup (void) {
3
4   // <h> External Bus Interface (EBI)
5   //   <e1.13> Enable Chip Select 0 (CSR0)
6   //     <o1> BA: Base Address
7   //                 <i> Start Address for Chip Select Signal
8   //     <o1.7..8>   PAGES: Page Size      <0=> 1M Byte    <1=> 4M Bytes
9   //                                       <2=> 16M Bytes  <3=> 64M Bytes
10  //                 <i> Selects Active Bits in Base Address
11
12  //     <o1.12>     BAT: Byte Access Type <0=> Byte-write
13  //                                       <1=> Byte-select
14  //     <e1.5>      WSE: Enable Wait State Generation
15  //       <o1.2..4>   NWS: Number of Standard Wait States <1-8><#/0x11>
16  //     </e>
17  //     <o1.9..11>  TDF: Data Float Output Time <0-7>
18  //                 <i> Number of Cycles Added after the Transfer
19  //   </e>
20    _WDWORD(0xFFE00000, 0x10002E29);   // EBI_CSR0: Flash
21  //<e1.7> init config
22  //<i> default 0x00,0x01
23  //</e>
24  #define INIT_CONFIG (0x01,0x81)
25
26
27  //   <q1.4>        DRP: Data Read Protocol
28  //                      <0=> Standard Read
29  //                      <1=> Early Read
30
31    _WDWORD(0xFFE00024, 0x00000010);   // EBI_MCR:  Data Read Protocol
32
33    _WDWORD(0xFFE00020, 0x00000001);   // EBI_RCR:  Remap Command
34
35    // </h>
36
37
38  // <o> Program Entry Point
39    PC = 0x04000000;
40  }
41
42  // <s> Change ID
43  // <s1.30> Change Password String
44  #define ID  "User ID"
    0 ref
45  char pw[] = "My Password";
46  // <<< end of configuration section >>>
```

图 8-1 头文件配置向导注释

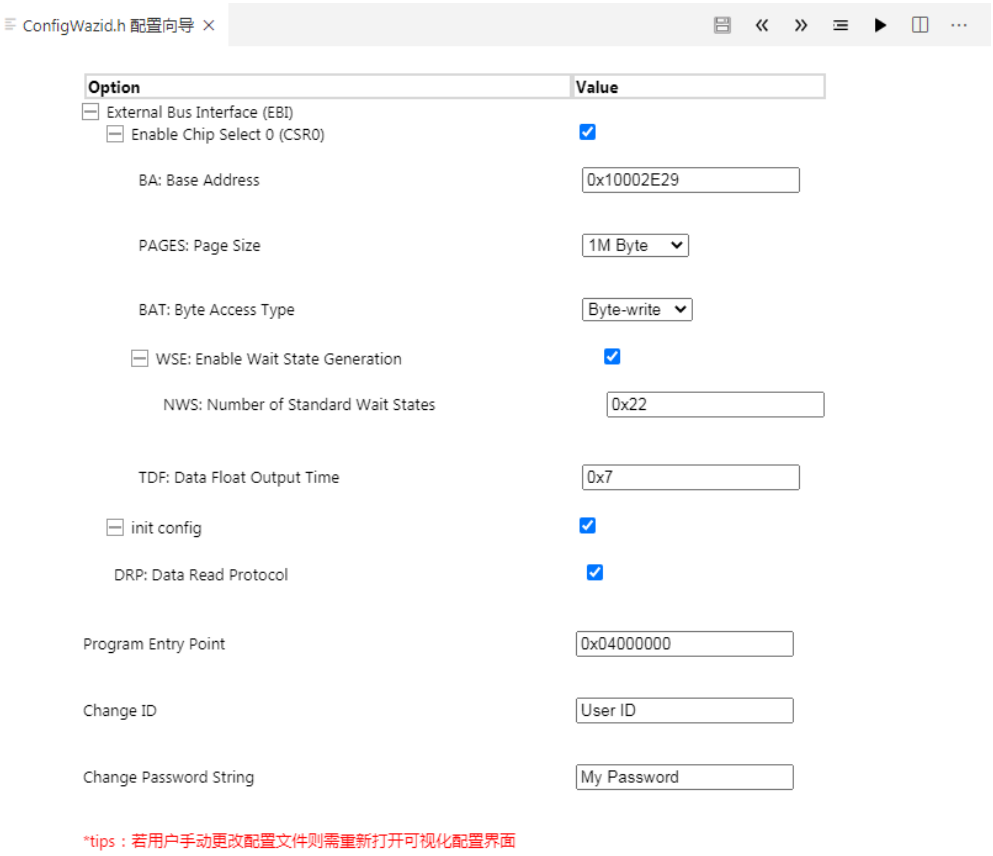


图 8-2 可视化配置界面

8.2 配置向导注释说明

根据头文件中的配置向导注释生成界面所需的项和修饰符。

1. 配置向导注释是创建配置控件所需的项和修饰符，配置向导注释被写成代码注释的形式；
2. 配置向导部分必须以下面的注释行开始；

   // <<< Use Configuration Wizard in Context Menu >>>

3. 配置向导部分以下面的注释行结束；

   // <<< end of configuration section >>>

4. 默认注释后面的下一个数字或字符串将被修改，也可跳过值，修改指定的值。

配置向导注释列表如下：

Item	Description
<h>	标题：紧随其后的选项归为一组。 使<h>到</h>中的选项为一个可展开的树形组。
<e>	可以使能的标题：紧随其后的选项归为一个**可展开的树形组**，可以通过复选框使能： 1. 在<h></h>的基础上**添加复选框**。当选中复选框后，才**可以配**

	置子选项。 2. <e>的主要使用是<e#1.#2>，其中#1 代表第几个数据(从 0 开始)，#2 代表此数据的指定位数(从 0 开始，若不指定则对整个数据操作) eg: //<e1.7>init config #define INIT_CONFIG(0X00,0X01) 其中<e1.7>就代表 0x01 的第 7 位。如果使能此复选框，则 0x01 变为 0x81;
</h> or </e>	标题或使能的结束符。 eg：//<e1.7> init config //<i> default 0x00,0x01 //</e> #define INIT_CONFIG (0x01,0x81) 代表此复选框是针对语句：#define INIT_CONFIG (0x01,0x81)
<c> </c>	产生一个复选框，选中复选框则使代码（<c>到</c>中的代码行）使能，否则被注释
<c#>	#代表数字，表示从定义此选项后的第几行开始，#的取值范围为<c>到</c>的这一段代码的行数范围。 eg://<c1> time option enable #define TIME_VALUE 1000 //第 0 行 #define USING_TIME2 //第 1 行 //</c> 表示此标签仅会对#define USING_TIME2 有定义和未定义之分，不会影响到#define TIME_VALUE 1000
<q>	一个将 bit 值设置为复选框的形式。 主要使用<q#1.#2>,#1 和#2 代表的意义和<e#1.#2>的一样。通过复选框设置数字的位值。 eg: //<q1.4> DRP: Data Read Protocol _WDWORD(0xFFE00024, 0x00000000); 其中<q1.4>就代表 0x00000000 的第 4 位，若勾选复选框，则 0x00000000 变为 0x00000010。 若不使用#1 和#2，直接使用<q> DRP: Data Read Protocol，则代表第 0 个数字第 0 位置，即 0xFFE00024 的第 0 位。
<o>	创建一个能够选择或输入数据的选项； 1. 直接修改一个数的数值（输入框）： eg：//<o> Program Entry Point PC=0x40000000; 若在标签为“Program Entry Point”的输入框输入 0x40000008,则 PC=0x40000008 2. 同时也可以修改数据的某个位或某几位： eg：//<o1.9..11> TDF: Data Float Output Time <0-7><#-1> _WDWORD(0xFFE00000, 0x10002429); 注：若仅修改一位，则界面将会以复选框的形式显示： eg：//<o1.9 > TDF: Data Float Output Time _WDWORD(0xFFE00000, 0x10002429);

	界面：  <o1.9..11>代表 0x10002429 的第 9、10、11 位;**<0-7>**：修饰符，代表输入值取值范围在 0-7（111）;**<#-1>:**修饰符，代表输入值再减 1 才是实际得出的结果。 3. 同修饰符**<#=>**组合使用形成下拉列表： eg：//<o1.7..8> PAGES:page Size <0=> 1M Byte <1=> 4M Bytes // <2=> 16M Byte <3=> 64M Bytes _WDWORD(0xFFE00000, 0x10002429); 其中<o1.7..8>代表 0x10002429 第 7、8 位，<0=> 1M Byte 代表 00，将第 7、8 位修改为 00，以此类推。
<s>	字符串输入选项； 1. 修改字符串： eg：//<s> Change ID #define ID “My user id” 若在标签为“Change ID”的输入框输入 my change id,则#define ID “My user id” 2. 可以使用**<s#1.#2>** ,#1 代表第几个数据，#2 代表字符串的最大长度。 eg：//<s1.1> Change ID #define ID “My user id” Char pw[] = “passwaord”; 其中 s0.1 代表第 0 个字符串“My user id”，此字符串长度限制为 20。 **注： 若#1 不写则默认为第 0 个字符串，#2 不写则默认为不限制长度。**
<i>	标注提示信息使用，提示上一个标签有关信息 eg: //<o1.20..31>BA:BaseAddress <0x0-0xFFF00000:0x100000><#/0x100000> // <i> Start Address for Chip Select Signal _WDWORD(0xFFE00004, 0x040034B5); 代表输入框 BA:BaseAddress 的提示信息为 Start Address for Chip Select Signal

Modifier(修饰符)：主要和输入框结合使用（<o>），对输入值进行操作，modifier 主要如下列表：

Modifier	Description
<num1-num2>	选项字段的取值范围（超过此范围的默认为此范围的最大值）
<num1-num2:step>	选项字段的取值范围 num1-num2，步长 step
<#=>	用于选择的值和文本（形成下拉列表框使用）
<#+num> <#-num> <#*num> <#/num>	对要修改的值先与 num 进行+-*/处理，然后赋值给定义

9 注意事项

9.1 环境问题

1) 未安装工具链 ES32CC 所需的 vc2015 分发包时，对于 8 位的项目进行头文件分析时会报错。因此，针对 8 位的项目，Windows 下需要安装 vc2015 分发包（vc_redist2015.x86.exe）。

2) VSCode 在某些环境下可能会出现黑屏、卡顿等现象。若用户遇到这种情况，可以尝试以兼容性模式运行此程序。

3) 若用户遇到设置的断点信息不准确，比如：文件名全部变为大写，同一行可设置两个断点等异常信息，如下图所示。是由于 VSCode 版本问题，用户点击菜单栏的帮助按钮，选择“检查更新”，下载 VSCode 最新版本的安装包并安装，即可解决此问题。

9-1 断点异常

4) 若用户电脑安装的有 ESBurner，需要将 VSCode 设置为以管理员模式启动，否则的话，可能会进入调试失败。

9.2 智能编码注意事项

1) 用户可手动重新启动智能编码辅助功能，键盘输入“Ctrl+Shift+P”打开命令面板，在命令输入栏中输入命令“esls：Restart language server”，即可重新启动语言服务器，对打开的项目文件内容重新进行语法分析。