用于 C 系列的 TivaWare™ 入门信息

User’s Guide

Literature Number: ZHCU861A
MARCH 2021 – REVISED AUGUST 2022

目录

插图清单

关于本手册

德州仪器 (TI) 用于 Tiva-C 的 TivaWare™ 软件开发套件 (SDK) 为用户提供了器件驱动程序、头文件、应用示例、实用程序等，可加快 TM4C 系列微控制器器件的开发进程。本用户指南概述了 TivaWare SDK 提供的功能，并介绍了一些基本内容，包括如何利用 TivaWare 示例工程、如何调试 TivaWare 库以及可避免常见嵌入式编程错误的通用软件最佳实践。本指南中的所有示例均使用 TI 的 Code Composer Studio™ (CCS) 集成开发环境 (IDE)，但相同的概念也适用于其他 IDE。

术语表

TI 术语表　　本术语表列出并解释了术语、首字母缩略词和定义。

德州仪器 (TI) 提供的相关文档

有关这些器件的相关文档和开发支持工具的完整列表，请访问德州仪器 (TI) 网站 http://www.ti.com.cn。

支持资源

TI E2E™ 支持论坛是工程师的重要参考资料，可直接从专家获得快速、经过验证的解答和设计帮助。搜索现有解答或提出自己的问题可获得所需的快速设计帮助。

链接的内容由各个贡献者“按原样”提供。这些内容并不构成 TI 技术规范，并且不一定反映 TI 的观点；请参阅 TI 的《使用条款》。

This page intentionally left blank.

章节 1

TivaWare SDK 简介

本文的所有示例均使用 TivaWare 2.2.0.295 版，但提供的信息也适用于更高和更低版本的 TivaWare。在本文档中，提及的所有目录路径均以 TivaWare 安装文件夹 TivaWare_C_Series-2.2.0.295 为父目录。

1.1 TivaWare SDK 文件夹细目分类

本节简要介绍了 TivaWare SDK 中提供的内容，并提供了多个文件夹所关联的 TivaWare 文档。通过阅读这些文档，您可以进一步了解 SDK 中提供的内容。

在基础目录中，提供了 13 个文件夹和一些文件。图 1-1 中突出显示的 12 个文件夹中存储了所有相关的配套资料。基础目录中仅有的说明文件是 3 个 .txt 许可文件。基础目录中其余的文件以及 *.metadata* 文件夹是构建构件和 TI Resource Explorer 信息，无需理会。

boot_loader 文件夹包含执行 TivaWare 闪存引导加载程序所需的源代码。*TivaWare™ 引导加载程序用户指南 (SW-TM4C-BOOTLDR-UG)* 中介绍了引导加载程序功能的全部细节，包括 ROM 和闪存引导加载程序的区别。

docs 文件夹包含与 TivaWare 示例、库和实用程序相关的所有技术文档。其中还包含 TivaWare 官方版本说明，该说明重点介绍了每个版本的 SDK 中对 TivaWare 所做的更改 (*用于 C 系列的 TivaWare™ 版本说明 (SW-TM4C-RLN)*) 。

driverlib 文件夹包含 TivaWare 驱动程序库 (DriverLib) 源代码，用户可以借助它来利用经过 TI 验证的函数。借助 DriverLib 函数，所有程序员无需在寄存器级进行操作，即可轻松配置器件并控制外设。有关 DriverLib 源代码的全部细节，请参阅 *TivaWare™ 外设驱动程序库用户指南 (SW-TM4C-DRL-UG)*。

examples 文件夹包含 TivaWare 提供的所有示例工程。有关这些示例工程的详细信息，请参阅章节 2。

inc 文件夹包含每个 TM4C 器件的器件头文件以及硬件头文件。器件头文件提供针对特定器件的所有寄存器、偏移和位字段的定义。通用硬件头文件是 .h 文件，该类文件以 'hw_' 前缀开头，包含针对所有寄存器偏移和位字段的定义。TivaWare 库利用通用硬件头文件，以确保所有库函数都与器件无关，但针对特定应用，通用硬件头文件可由特定器件头文件代替。

> *如果代码文件同时使用器件头文件和通用硬件头文件的 #include，将出现指示 #define 重叠的编译器错误。头文件中只能使用一个 #define。*

grlib 文件夹包含 TivaWare 图形库源代码，该代码提供一套图形基元和小工具集，用于在各种 LCD 显示屏上创建图形用户界面。图形基元用于绘制定义的形状和图案。小工具用于绘制各种用户界面元素。此外，所有图形应用均需要显示驱动程序，在 TivaWare 示例文件夹中会提供应用级别的驱动程序。有关图形库源代码的全部细节，请参阅 *TivaWare™ 图形库用户指南 (SW-TM4C-GRL-UG)*。

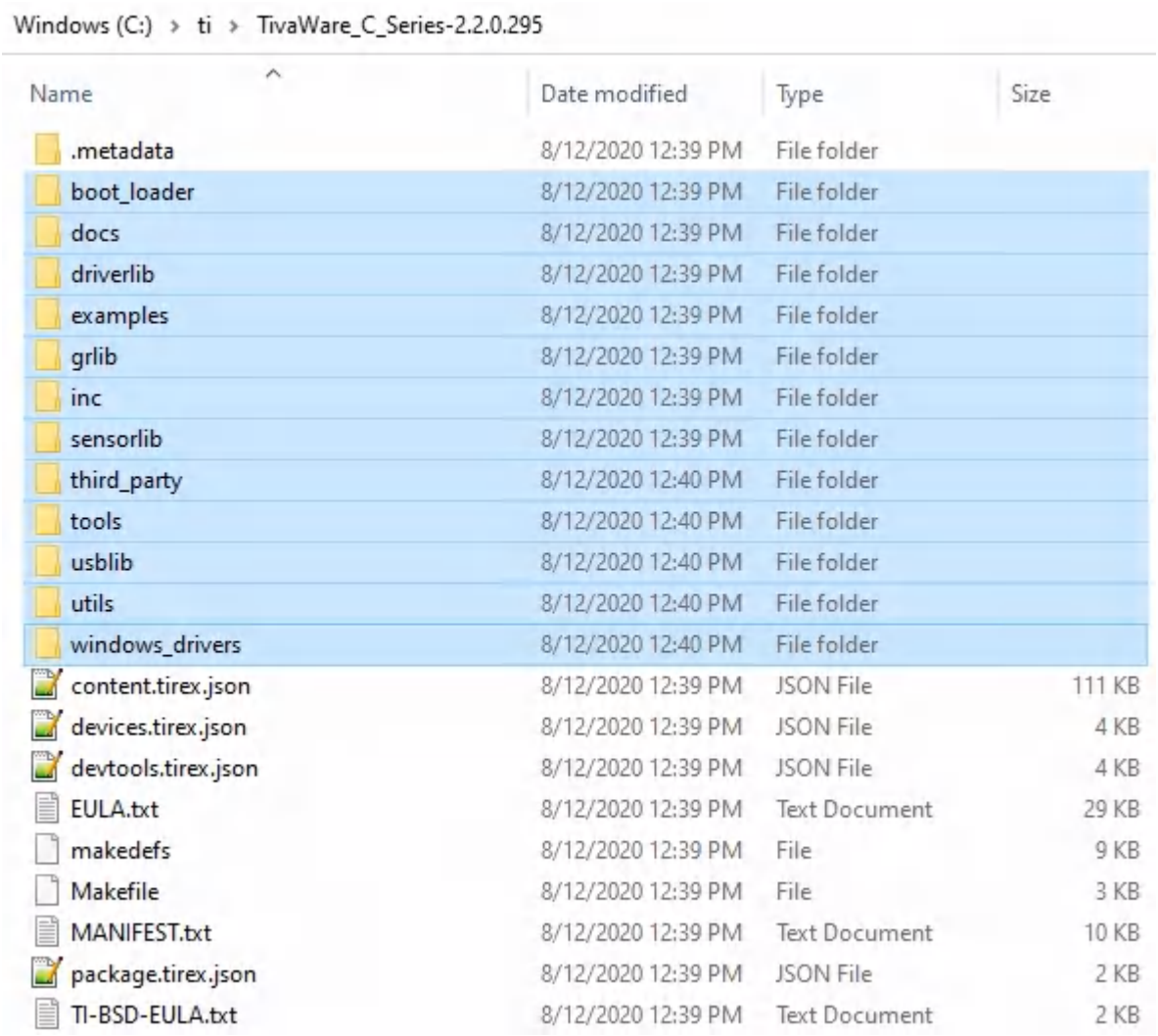

图 1-1. 完整的 TivaWare SDK 安装目录

sensorlib 文件夹包含 TivaWare 传感器库源代码，该代码用于各种主要来自原始 BOOSTXL-SENSHUB BoosterPack 的受 TivaWare 支持的传感器。有关相关产品的全部细节，请参阅 *TivaWare™ 传感器库用户指南 (SW-TM4C-SENSORLIB-UG)*。

third_party 文件夹包含来自第三方的各种开源资源，可用于为 TivaWare 示例提供支持。这些资源包括有关 Exosite 云物联网演示（*使用 EK-TM4C1294XL LaunchPad 的物联网演示应用报告*）、通用 FAT 文件系统、FatFs、SD 卡、FreeRTOS 支持和各种以太网协议相关应用程序（包括 lwIP 1.4.1）的文件。

tools 文件夹包含在开发系统上而不是 TM4C 器件上运行的 TivaWare 主机工具。提供的这些工具有助于开发适用于 TM4C 应用的固件。有关相关工具的全部细节，请参阅工具用户指南 (SW-TM4C-TOOLS-UG)。

usblib 文件夹包含 TivaWare USB 库源代码，该代码向用户提供了适用于器件、主机和移动模式的基本 USB 功能。USB 库提供了器件枚举、端点管理和特定类模块的描述符。有关 USB 库源代码的全部细节，请参阅 *TivaWare™ USB 库用户指南 (SW-TM4C-USBL-UG)*。

utils 文件夹包含 TI 开发的各种实用程序，可帮助创建应用程序。这些实用程序与器件无关，有助于构建更高级的用户应用程序。有关这些实用程序的全部细节，请参阅实用程序库用户指南 (SW-TM4C-UTILS-UG)。

windows_drivers 文件夹包含所有 TivaWare USB 器件示例中都需要的已签名 Windows USB 设备驱动程序。

章节 2

TivaWare 示例工程

examples 中包含三个子文件夹，本节将详细介绍 *boards* 文件夹和 *peripherals* 文件夹。*project* 文件夹只包含一个基于 TM4C123GH6PM 的基本 TivaWare 工程，供开发人员启动在 IDE (例如 Code Composer Studio) 中的软件开发。TM4C123x 和 TM4C129x MCU 的 boards 文件夹中也提供了类似工程。

2.1 TivaWare 开发板示例

examples/boards 文件夹内包含以下各项的专用文件夹：提供的每个 TM4C LaunchPad 评估套件、TM4C129x 开发套件、可连接开发板的多个 BoosterPack 的组合。

每个独立 LaunchPad 或开发套件的文件夹都包含 *project0* 示例。建议您在进行新的 TivaWare 开发工作时将该示例用作基准代码工程。

每个示例工程包括以下集成开发环境 (IDE) 的工程文件：Code Composer Studio (德州仪器 (TI)) 、IAR Embedded Workbench (IAR Systems) 和 µVision® IDE (Keil)。

2.2 TivaWare 外设示例

examples\peripherals TivaWare 文件夹包含常见 TM4C 外设的基本示例代码。这些示例全部以 .c 源文件提供，不含相应的特定 IDE 的工程。这些文件配置为支持 TM4C123x 和 TM4C129x 器件。

若要利用这些示例，首先导入所需 EVM 的 *project0* 示例，重新命名 IDE 中的工程，然后复制并粘贴外设原代码文件的完整内容，以代替 *project0.c* 的完整内容。

通过将 *project0* 用于正确电路板，为 TM4C123x 或 TM4C129x 预定义的符号将自动加载到指定 IDE 的工程中。

如果工程使用 **UARTprintf** 等附加实用程序，则可能需要链接附加文件。有关如何将附加文件链接添加到 TivaWare 工程中的详细信息，请参阅 节 3.1。

2.3 如何将示例工程导入 CCS

Code Composer Studio 具备简单易用的导入功能，依次点击“Project”→“Import CCS Projects”即可找到该功能。若要导入任何 TivaWare 工程，请点击“Select search-directory”选项的“Browse”按钮。此功能将查找所提供的目录中的**所有** CCS 工程。TivaWare SDK 中提供多个示例工程，因此如果不是导航至准确的目标工程，则强烈建议至少导航至具体板级。图 2-1 显示了导航至 *examples\boards\ek-tm4c123gxl* 时的选项。

用户一次可以导入多个工程。只需滚动浏览列表并选择应导入的所有工程，然后点击“Finish”。系统会自动将所有 TivaWare 示例工程作为新工程复制到工作区，因此原始工程不会改变，可随时重新导入。

由于是将工程复制到工作区，如果从 CCS 工作区中移除了之前导入的工程，但未将其从文件系统中删除，那么可能出现错误提示，指示工程仍然存在于工作区中。在这种情况下，可以从工作区目录中将工程重新导入到 CCS，也可以从工作区目录中删除工程文件夹，然后再次尝试从 TivaWare 目录导入工程。

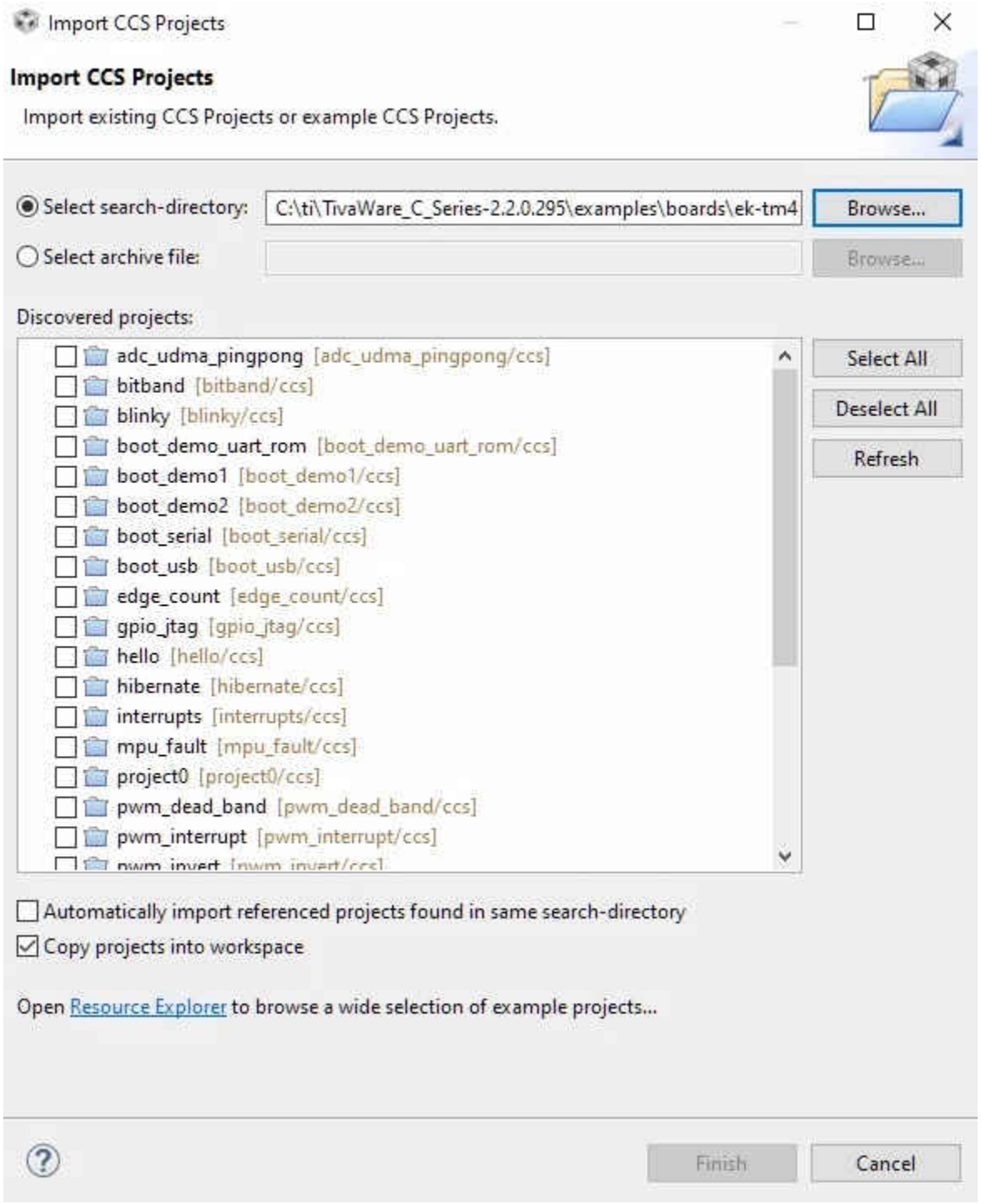

图 2-1. 导入 CCS 工程实用程序的示例

章节 3

在 Code Composer Studio 中将文件和库关联到 TivaWare 工程中

在扩展 TivaWare 工程以包含附加文件或库（例如添加新实用程序或包含 *usblib* 等库）时，需要采取特定步骤来确保正确添加文件和库，否则工程将返回构建错误。本节将介绍如何避免这种常见问题。

3.1 在 CCS 中链接文件

在本节中，将使用 *examples/peripherals/adc* 下的 *single_ended* 外设示例作为参考。

按照 节 2.2 中的步骤，将 EK-TM4C123GXL LaunchPad 的 project0 用作起始 TivaWare 工程。将工程名重命名为“single_ended”，以便后续能够重新导入 project0。将源代码复制到 *project0.c* 文件并尝试编译工程后，会出现图 3-1 中所示的构建错误。

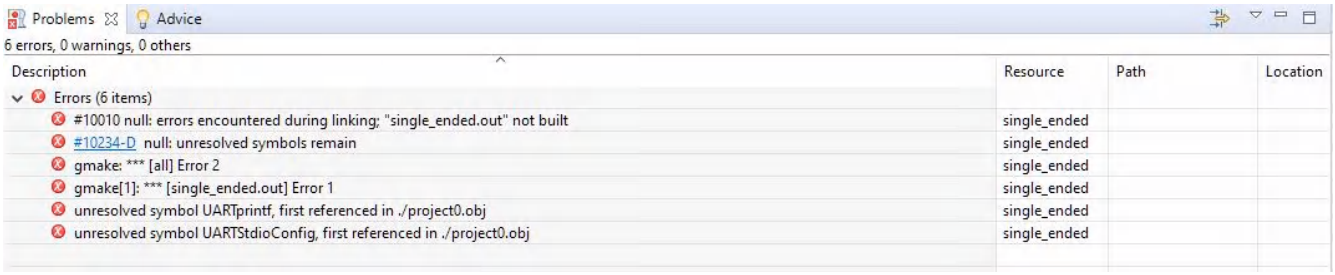

图 3-1. CCS 构建错误 - 未解析的符号

如果在使用函数调用时没有在工程中添加合适的 .c 源文件，通常会出现“未解析的符号”错误。如果工程使用除了所提供的 TivaWare 库（DriverLib 或 UsbLib）以外的文件，通常会出现该错误。如果使用 #include 从 TivaWare 库目录中添加了头文件，那么应进行检查以确保已将相应的 .lib 文件添加到工程中。如果已添加，则无需执行进一步操作。节 3.2 将介绍在工程中缺失 .lib 文件的情况下如何添加该文件。

在本例中，以下两个 UART 相关函数导致了这个问题：**UARTprintf** 和 **UARTStdioConfig**。由于本示例用于现有 TivaWare 示例工程，了解缺少哪个文件的最快捷方式是向上滚动到代码工程顶部的 #include 语句。执行此操作后，可以找到与 UART 相关的以下行：

```
#include "driverlib/uart.h"
#include "utils/uartstdio.h"
```

如前文所述，由于 .lib 文件已经包含 *driverlib* 文件，接下来要检查是否缺少 *uartstdio.c* 文件。仔细查看工程中的文件列表后，发现工程中不包含 *uartstdio.c* 文件（图 3-2）。通过搜索 TivaWare 安装 *utils* 目录中 *uartstdio.c* 文件的内容，可以发现该文件中也有缺少的函数。

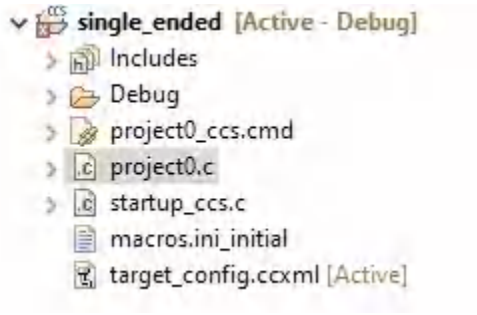

图 3-2. single_ended 工程的文件列表

在识别出文件之后，需要通过以下步骤将文件链接到 CCS 工程。

1. 转至 CCS 工程，右键点击工程名称以打开工程的选项菜单（图 3-3）。
2. 点击“Add Files”。

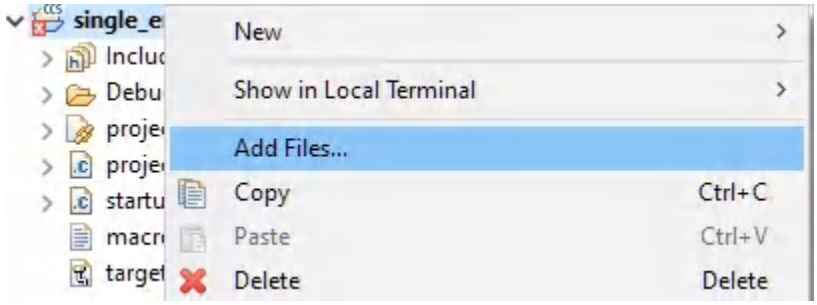

图 3-3. CCS 工程的“Add Files”选项

3. 此时将出现文件浏览器，如前文所示，需要从 *utils* 目录中添加 *uartstdio.c* 文件。导航到该目录并选择该文件。

   按下 CTRL 键并点击每个必须链接的文件，以选择多个文件。

4. 添加文件时，CCS 将询问是复制还是链接文件。选择“Link to files”选项（图 3-4）。如果从 TivaWare SDK 中添加另一个文件，建议将“Create Link locations relative to:”路径更改为“SW_ROOT”。**SW_ROOT** 是计算机上基础 TivaWare SDK 目录的 TivaWare 定义路径。

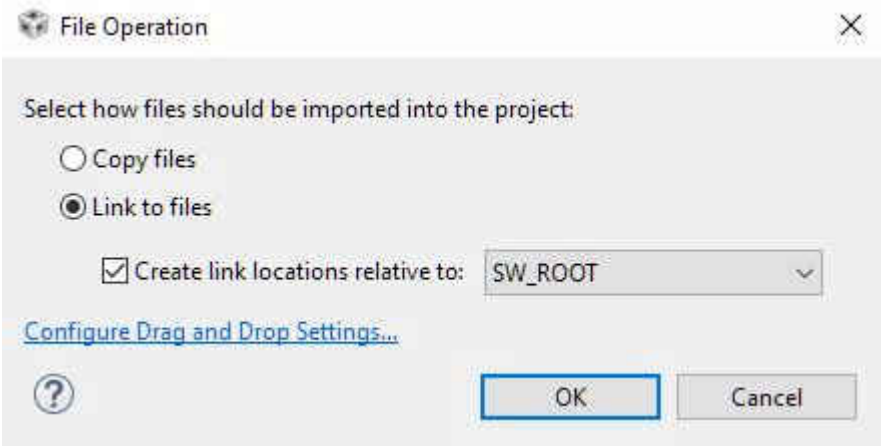

图 3-4. 针对其他 TivaWare 文件的建议链接位置路径

可以在“Resources”→“Linked Resources”→“Path Variables”选项卡视图下的 CCS 工程属性中找到 **SW_ROOT** 定义。

现在，*uartstdio.c* 文件应显示在工程的文件列表中（图 3-5），现在将能够成功地重新构建工程！

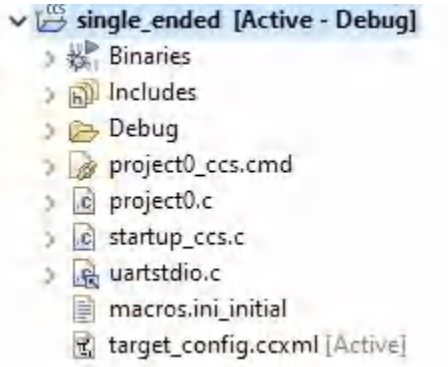

图 3-5. *uartstdio.c* 现已包含在 CCS 工程中

检查是否已正确完成该过程的另一个方法是转至 CCS 工程属性，导航到“Resource”→“Linked Resources”页面，然后选择“Linked Resources”选项卡。在此选项卡下，找到所列的“Variable Relative Location”，如图 3-6 所示。

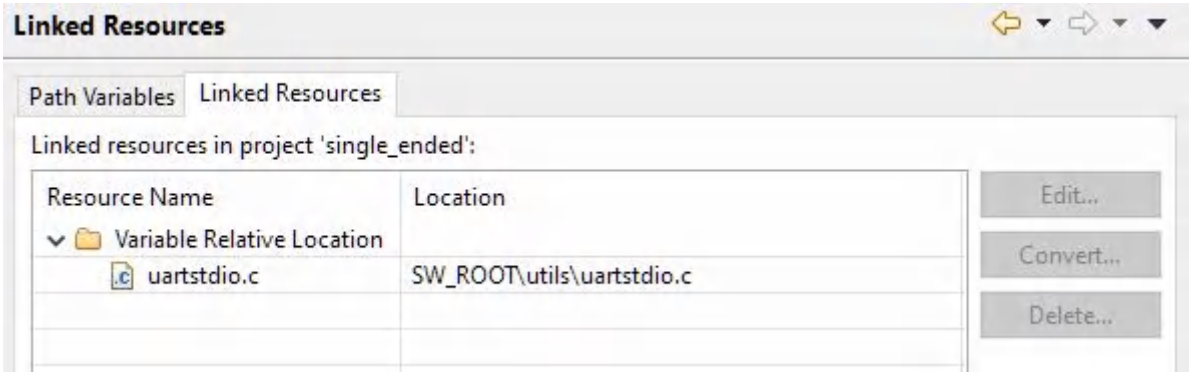

图 3-6. CCS 工程属性中的链接资源

3.2 在 CCS 中链接库

TivaWare 示例工程仅包含与提供的示例工程相关的 TivaWare 库的链接。这意味着，虽然所有工程将包含 DriverLib 链接，但 UsbLib 或 GrLib 等其他库可能无法链接到工程，因此需要执行一些步骤来链接库。

需要链接的库文件是 .lib 文件，可在 TivaWare 安装程序的库文件夹中找到它。在此示例中，将使用 *usblib.lib* 文件。在编译库时，库的 .lib 文件是构建输出，可在 *usblib\ccs\Debug* 下找到。

为了在现有工程中添加库，应转到“Project Properties”，并导航到“Build”→“ARM Linker”→“File Path Search”。此时将出现“Include library file or command file as input”部分。点击图 3-7 中圈出的“Add…”按钮。

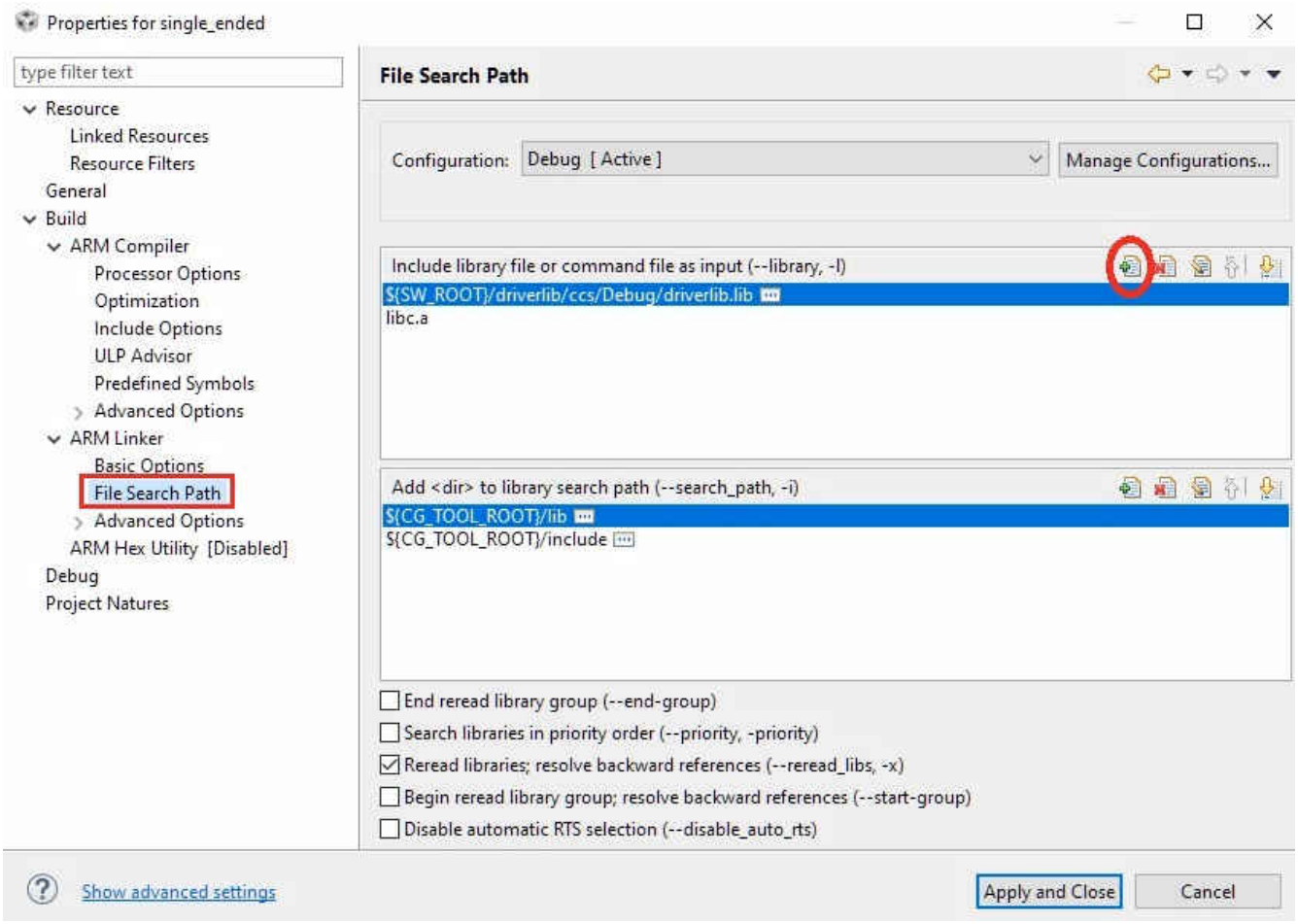

图 3-7. 如何在 CCS 工程属性中添加库链接。

接下来，点击“Browse…”按钮，并导航至包含 .lib 文件的文件夹，选择库文件并点击打开，或双击文件。显示的路径应与图 3-8 类似。点击“OK”，然后在工程属性页面点击“Apply and Close”，库即可链接到 CCS 工程。

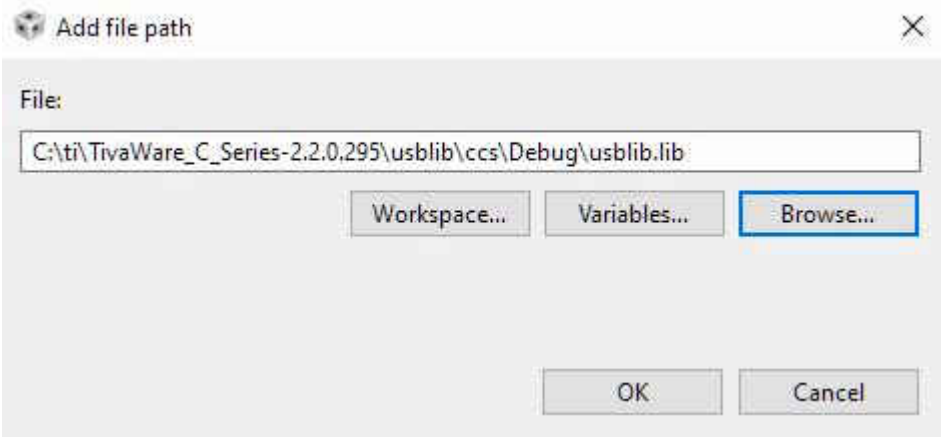

图 3-8. UsbLib 库文件的文件路径

虽然不是必需的，但为了使用 **$SW_ROOT** 链接资源路径，可以进一步编辑链接位置。如图 3-7 所示的 *driverlib.lib* 链接路径，只需用 **$SW_ROOT** 代替 TivaWare 目录即可。

This page intentionally left blank.

TI 库函数很少出现问题，但如果需要在库级别调试程序，则需要执行一系列操作来实现。提供的 .lib 文件不允许 IDE 调试功能为了单步执行库函数而查看源代码。TI 提供了几种调试库代码的方法。最简单的方法是将库文件夹中的源文件导入到工程中，如 节 3.1 中所示。本章中将介绍的其他两个方法是在尝试 C 源调试或重新编译整个库时，将 IDE 指向源文件，以便包含本地系统的正确路径信息，从而让 IDE 找到 C 源文件。

4.1 如何引导 Code Composer Studio 找到源文件

如果在调试代码时需要访问库调用的源文件，那么需要将 CCS 指向源代码文件。若要实现这一点，可在调试过程中将断点和步进代码调试技术结合使用，从而提示 IDE 尝试并找到文件（之后可以手动放入本地文件系统）。

第一步是在需要调试的准确函数调用中设置断点。设置方法是双击这行代码旁边的灰色区域，或者右键点击代码行并选择弹出菜单顶部的“Breakpoint”。设置断点后，代码行旁边的灰色区域出现了一个蓝色球体，如图 4-1 所示。

> 如果调用函数时使用了 MAP 或 ROM 前缀，应删除前缀以避免任何 ROM 库调用，因为调试器无法访问这些调用。

```
123     //
124     // Enable the GPIO pins for the LED D1 (PN1).
125     //
126     GPIOPinTypeGPIOOutput(GPIO_PORTN_BASE, GPIO_PIN_1);
```

图 4-1. 在 DriverLib 调用中设置断点

设置断点后，执行代码，直至断点被触发。断点触发后，将停止执行代码，顶部的工具栏将出现图 4-2 中所示的选项。使用图中突出显示的“Step Into”功能或键盘快捷键 F5 以触发 IDE 搜索源代码文件。

图 4-2. CCS 调试控制选项

提示 IDE 查找源文件后，将无法找到文件，并弹出一个通知（图 4-3）。在此弹出窗口中，用户可以选择“Locate File...”，此选项用于引导 IDE 找到正确的文件。点击“Locate File...”后，导航至本地文件系统中的 DriverLib 文件夹，然后选择它。选择整个文件夹，而不是一个单独的文件，因此无需准确知道需要哪个源代码文件。IDE 将搜索文件夹中的所有文件，并自动弹出正确的文件。

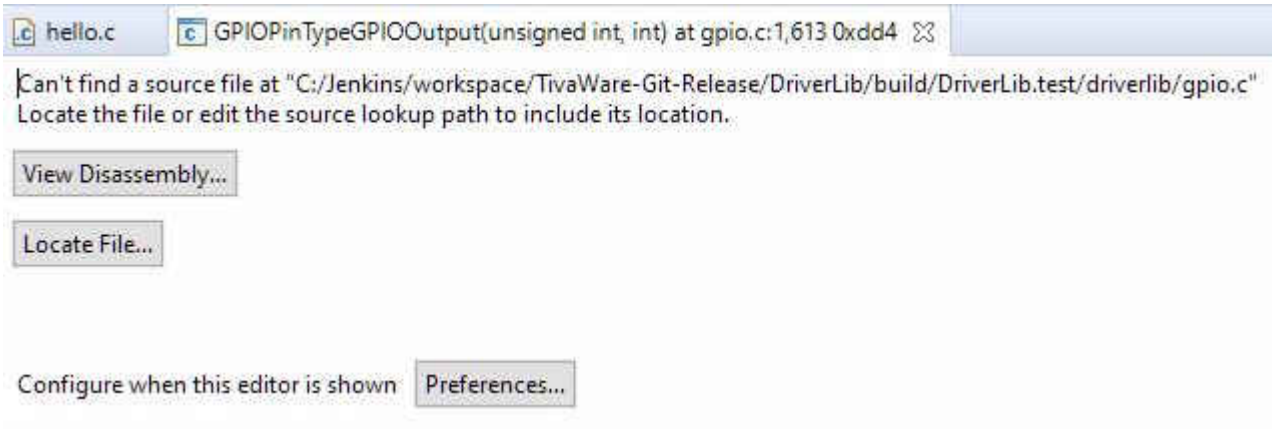

图 4-3. “Locate File”弹出窗口

提供源代码的目录后，弹出窗口会消失，这时将打开源代码文件，光标位于逐步执行的函数内部，如图 4-4 所示。可以从此处继续进行正常的调试。

```
1610 //*****************************************************************************
1611 void
1612 GPIOPinTypeGPIOOutput(uint32_t ui32Port, uint8_t ui8Pins)
1613 {
1614     //
1615     // Check the arguments.
1616     //
1617     ASSERT(_GPIOBaseValid(ui32Port));
1618
1619     //
1620     // Set the pad(s) for standard push-pull operation.
1621     //
1622     GPIOPadConfigSet(ui32Port, ui8Pins, GPIO_STRENGTH_2MA, GPIO_PIN_TYPE_STD);
1623
1624     //
1625     // Make the pin(s) be outputs.
1626     //
1627     GPIODirModeSet(ui32Port, ui8Pins, GPIO_DIR_MODE_OUT);
1628 }
```

图 4-4. 成功查找并加载 DriverLib 源代码文件

4.2 如何重新构建 TivaWare 库

在有些情况下，需要重新编译库文件以获取新 .lib 文件输出。一种情况是本章开始时提到的启用库调试的情况。更常见的情况是根据速度/规模的权衡结果更改库的优化。在极少数情况下，需要体现对库文件所做的更改，这时重新编译 .lib 文件是为了体现所做的更改。使用这种方法而不是将修改的源文件保存在工程中所面临的风险是，它需要长时间保留库，以确保各个版本的更新不会丢失。

为了重新编译 TivaWare 库，必须将其导入 CCS 并重新构建。除了一个关键区别之外，操作步骤与 节 2.3 中所述的步骤相同。必须取消选中“Copy projects into workspace”复选框。这样，将保留位于所有 TivaWare 工程所关联的 TivaWare 文件夹中的库工程和构建输出。

导入之后，只需重新构建工程，更改就会应用于 TivaWare 目录中的新 .lib 文件中。

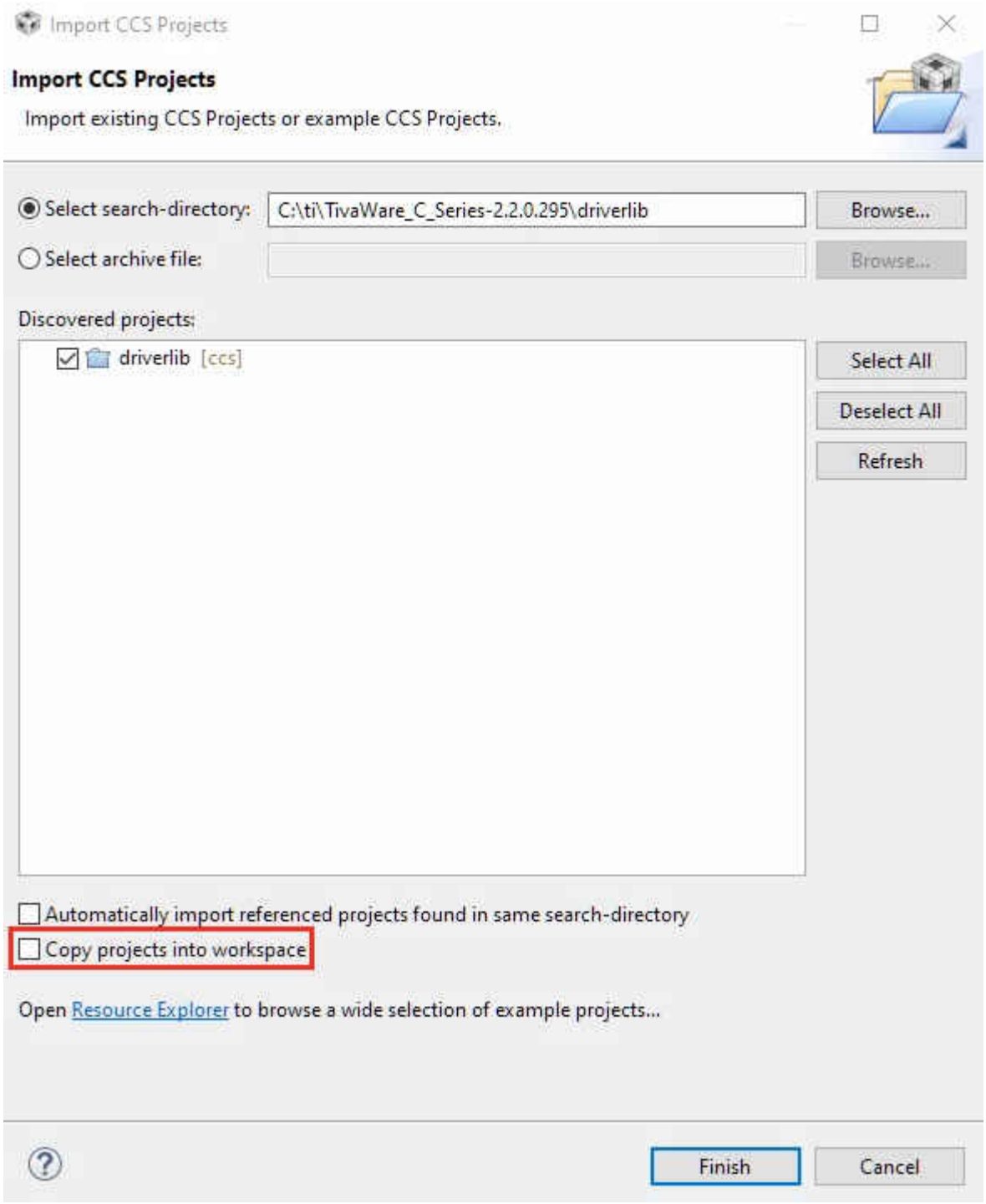

图 4-5. 将 DriverLib 导入 CCS 的正确方法

This page intentionally left blank.

章节 5

如何将 TivaWare 添加到现有 CCS 工程中

TI 始终建议通过现有 TivaWare 示例开始构建新工程，但有时需要修改现有工程才能将 TivaWare 添加到工程中。与仅关联库相比，这个过程涉及的操作更多，本节将介绍需要注意的所有其他事项。

执行这些步骤的过程中显示的图像是针对定制 CCS 工程的，该工程尝试包含外设示例 *single_ended* 的功能性。每一项更改都将在 CCS 工程属性中进行。如果您不确定参考位置，可以从工程属性开始，然后执行所示的步骤。

5.1 路径变量

如果尝试将 TivaWare 添加到现有工程中，出现的第一个错误是特定文件和文件夹的路径不能正确链接在工程属性中。在每个 TivaWare 工程中，编译器可以在几个包含路径中查找具体文件的位置。最重要的路径是 TivaWare 根目录。它被定义为 TivaWare 工程中的路径变量 **SW_ROOT**，但不一定要将它添加为路径变量。可以通过导航到文件夹来添加包含选项。但是，**SW_ROOT** 可以在其他多种情况下重复使用，因此建议添加路径变量。

若要添加路径变量，请转到“Project Properties”，然后导航至“Resource”→“Linked Resources”。应该有一份如图 5-1 中所示的现有链接文件夹列表。若要添加 **SW_ROOT** 路径，请点击“New”。将路径命名为“SW_ROOT”，然后点击“Folder”。导航至 TivaWare 安装文件夹（在 Windows 中，默认为 C:\ti\TivaWare_C_Series-2.2.0.295），然后点击“Select Folder”。核实路径是否正确，然后点击“OK”。

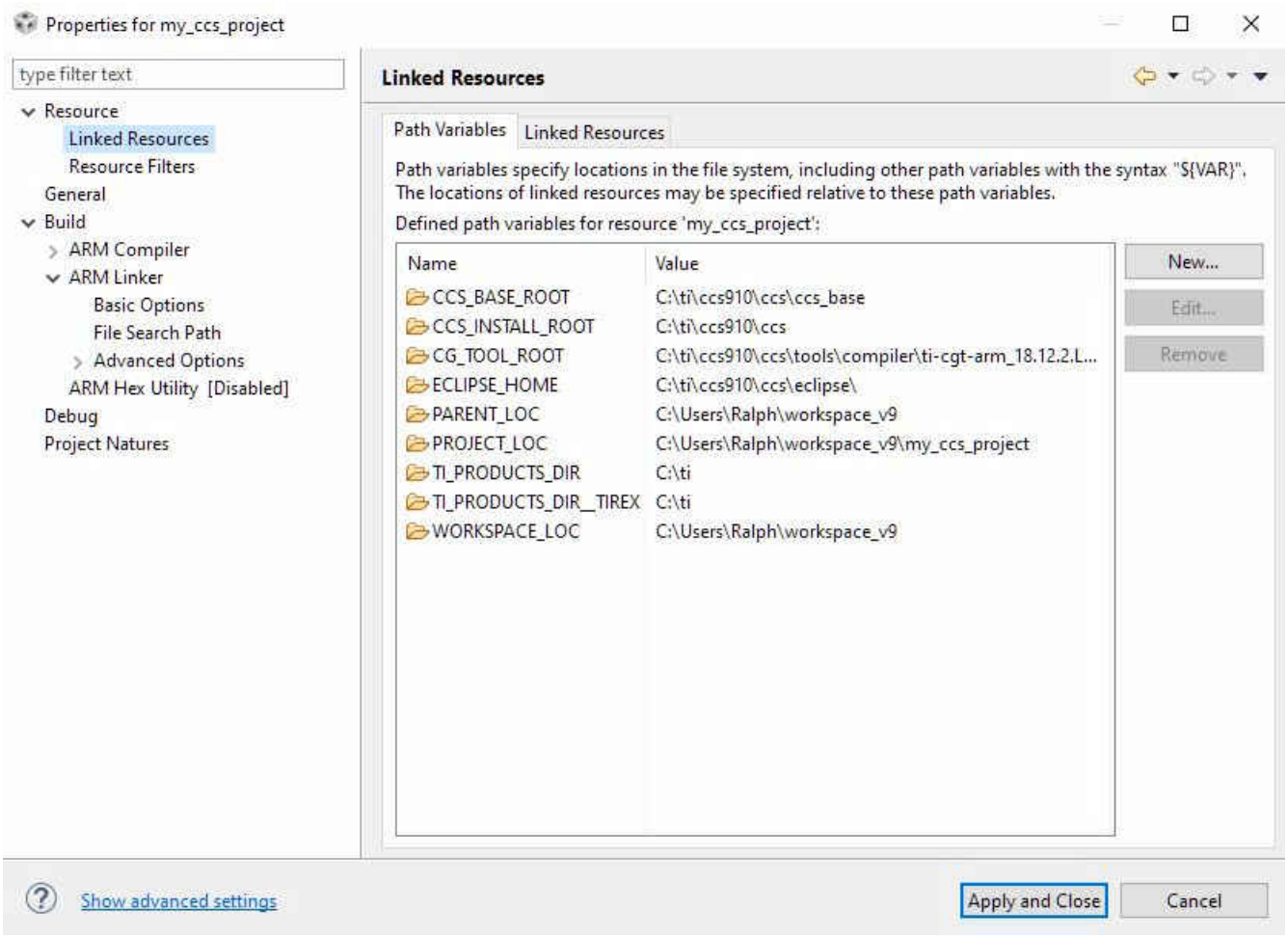

图 5-1. CCS 工程属性中的路径变量菜单

5.2 包含路径

添加 **SW_ROOT** 的路径变量后，即可轻松添加各种包含路径的链接。需要的第一个包含路径是 **SW_ROOT**，它是 TivaWare 基础目录。它可以解决与文件链接相关的很多构建错误。

在 CCS 工程属性中，导航至“Build”→“ARM Compiler”→“Include Options”。这里有两个框，但是将不会使用“preinclude file”框。查看“Add dir to #include search path”框，应显示当前路径版本，通常至少包含一个 **PROJECT_ROOT** 版本和一个 **CG_TOOL_ROOT** 版本。若要将 **SW_ROOT** 添加到其中，请点击“Add…”按钮（图 5-2），然后在“Directory”字段中输入 *${SW_ROOT}*，并点击“OK”。

一些工程可能还需要添加 LaunchPad 文件夹。要添加的路径形式如下：*${SW_ROOT}/examples/boards/ek-tm4c123gxl*，最终文件夹名称是所使用的 LaunchPad 文件夹。

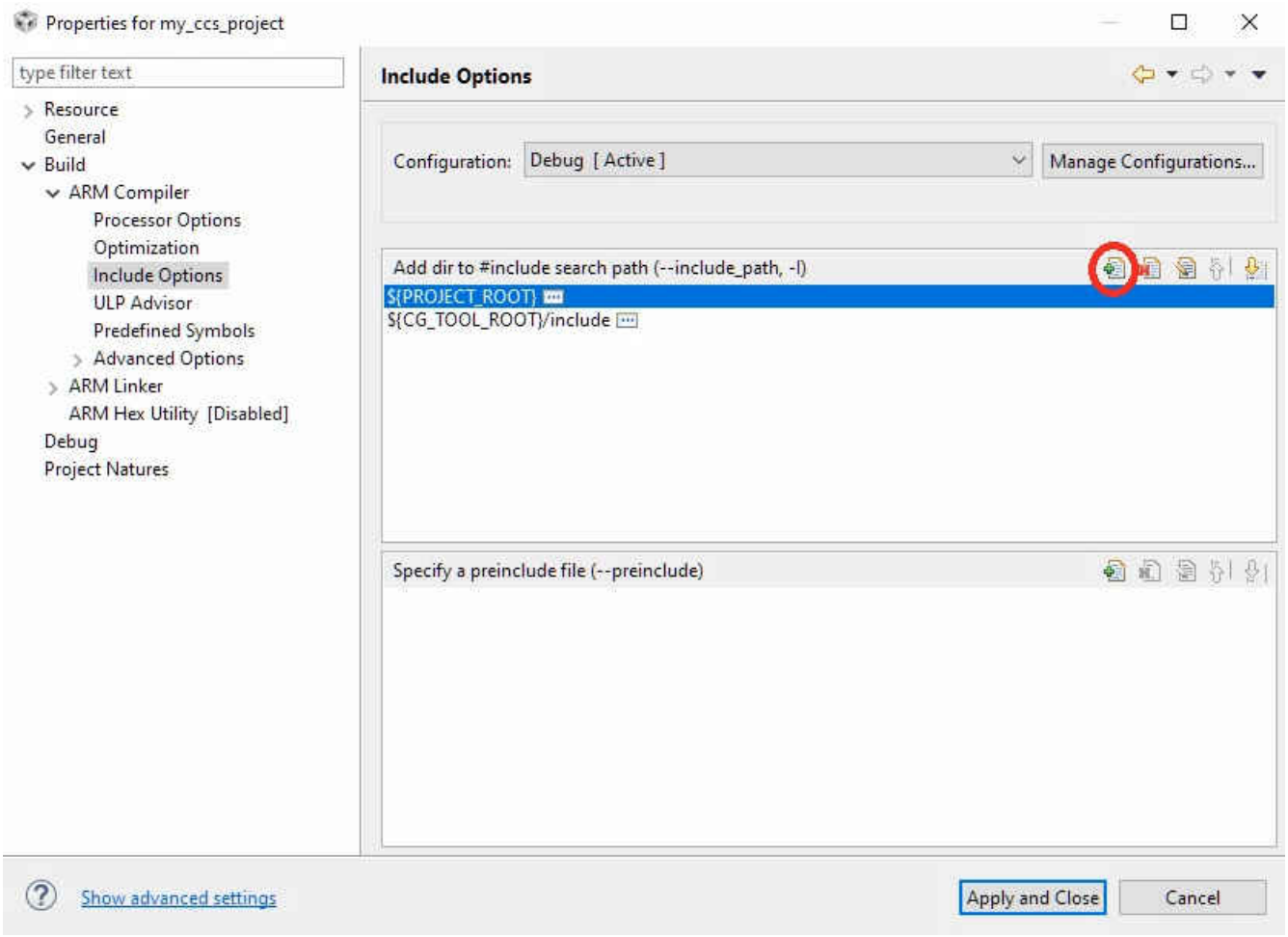

图 5-2. 如何在 CCS 工程属性中添加新包含路径

5.3 预定义变量

下一步是在工程中添加预定义符号。TivaWare 文件（包括 DriverLib）根据所用的 TM4C MCU 系列使用预定义符号来选择代码块。若要添加符号，首先转到 CCS 工程属性，然后导航至“Build”→“ARM Compiler”→“Predefined Symbols”。现在，应该已经为正在使用的 TM4C 微控制器定义了至少一个符号，如图 5-3 中所示的 **PART_TM4C123GH6PM**。如果由于某种原因它不存在，也必须添加。应该在“**PART_**”后使用的器件型号是在工程属性的“General”部分选定的器件。

若要添加新符号，请点击“Add…”按钮，并在弹出的框中键入值。需要添加的符号取决于要使用的 MCU 系列。如果使用的是 TM4C123x 器件，则添加 **TARGET_IS_TM4C123_RB1**。如果使用的是 TM4C129x 器件，则添加 **TARGET_IS_TM4C129_RA2**。点击“OK”以添加新符号。

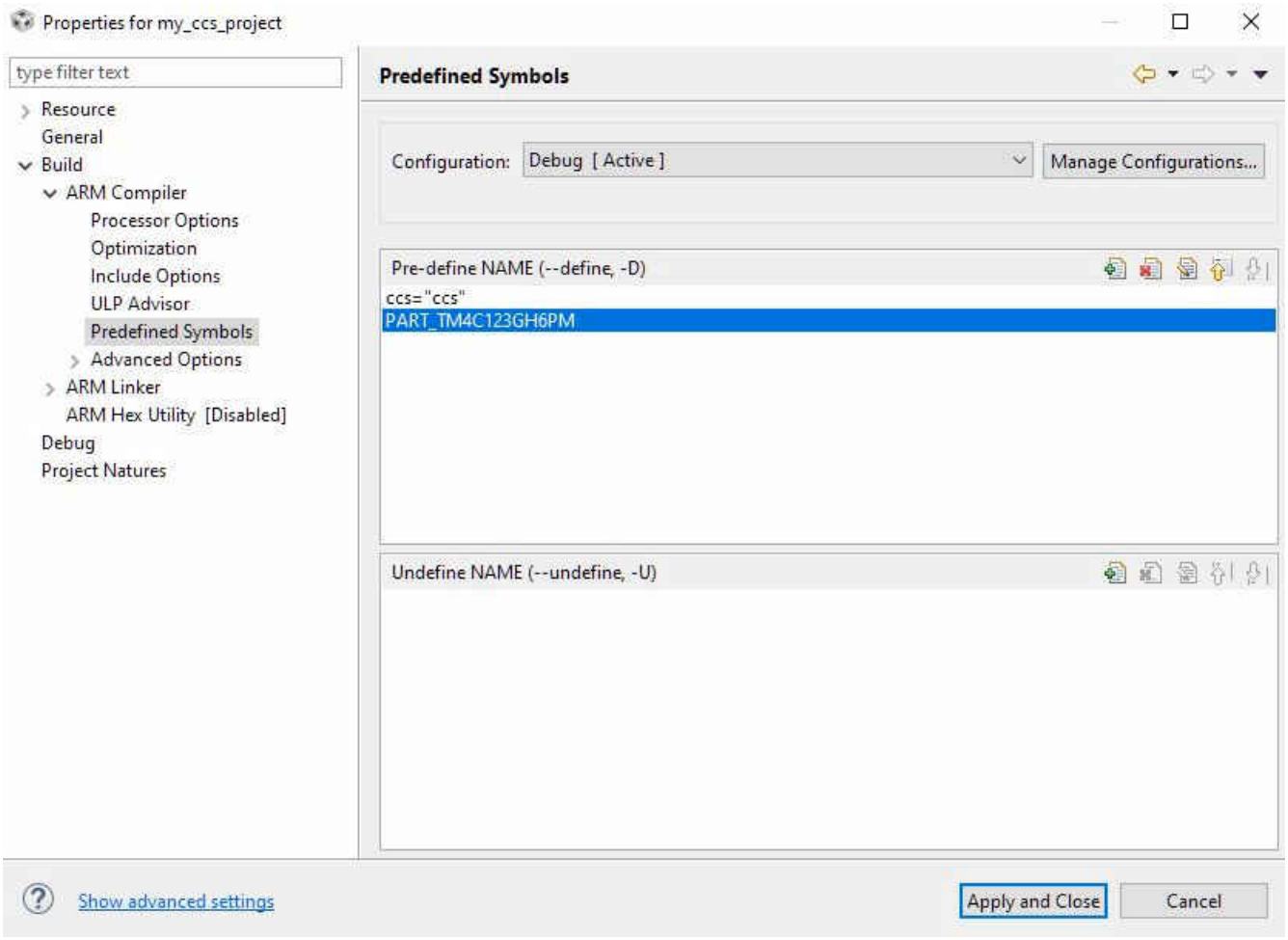

图 5-3. CCS 工程属性中的预定义符号菜单

5.4 库链接

最后一步是按照 节 3.2 中的说明将所有相关 TivaWare 库链接到 CCS 工程。至少需要 *driverlib.lib*，可能还需要其他库。根据所添加的库，还应记住可以按照 节 3.1 中所述链接文件。

接下来，应完成需要的所有步骤。开始尝试构建工程，如果出现错误，检查是否缺失需要链接的文件。如果问题持续存在，应检查正式 TivaWare 工程的链接位置以进行比较。

This page intentionally left blank.

章节 6

TivaWare 引导加载程序

TM4C 微控制器提供的一项重要功能是能够使用多个接口，以便通过引导加载程序更新器件上的固件。提供了 ROM 和闪存引导加载程序选项。可以在 *TivaWare™ 引导加载程序用户指南 (SW-TM4C-BOOTLDR-UG)* 中找到这些模式之间的差异比较、ROM 引导加载程序的使用方法以及所支持接口的所有详细信息。

本节将重点介绍如何使用闪存引导加载程序，以便在回顾可以使用的工具之前，回顾如何在 CCS 中配置引导加载程序，然后执行引导加载流程。

6.1 在 Code Composer Studio 中修改要用于引导加载的 TivaWare 工程

在本节中，将使用适用于 EK-TM4C123GXL LaunchPad 开发套件的 TivaWare *boot_serial* 示例作为引导加载程序固件，并将使用 *hello* 示例作为应用程序代码。

boot_serial 示例仅包含 *bl_config.h* 文件，因为闪存引导加载程序本身是 TivaWare *boot_loader* 文件夹的一部分。对引导加载程序的任何修改可通过 *bl_config* 文件完成，可在文件中选择串行接口并进行修改，包括选择要使用的模块和引脚 (不同于锁定至特定引脚的 ROM 引导加载程序) 。另一项重要功能是能够设置起始地址。起始地址定义为 **APP_START_ADDRESS**，它用于确定由引导加载程序加载的应用程序在闪存中的起始地址。需要将引导加载程序置于从 0x0000.0000 地址开始的闪存中，因此需要起始地址。**APP_START_ADDRESS** 的大小是根据引导加载程序代码的长短定义的。它用于在引导加载流程结束时跳至应用程序代码的起始位置，并开始执行它。

```
#define APP_START_ADDRESS       0x2800
```

TM4C123x 器件和 TM4C129x 器件的 **APP_START_ADDRESS** 有所不同，因为不同系列的闪存结构存在差异。通常，确定起始地址时有两个主要考虑因素。第一个是矢量表必须与 1024 字节页的边界对齐。第二个因素是在对应用程序代码进行编程时，不能擦除引导加载程序，因为闪存是按扇区擦除的。

对于 TM4C123x 器件，闪存扇区的大小为 1kB，但对于 TM4C129x 器件，大小为 16kB。为了避免引导加载程序和应用程序位于同一扇区中，最简单的解决方案是从引导加载程序结束后的第一个扇区边界开始存储应用程序。这还可以满足矢量表与 1024 字节页边界对齐的要求。对于使用随附的 TivaWare 闪存引导加载程序的 TM4C123x 器件，起始地址设置为 0x2800，这是不存在任何引导加载程序代码的第一个扇区。对于 TM4C129x 器件，起始地址设置为 0x4000，因为扇区大小为 16kB。

定义起始地址后，下一步是编译将要引导加载的应用程序代码，因此起始地址将体现在生成的二进制文件中。可以通过修改 CCS 工程的链接器命令文件来实现此操作。打开 *hello* 工程或将其导入 CCS 后，找到 *hello_ccs.cmd* 文件并双击，即可在 CCS 链接器命令文件编辑器中打开该文件。本文件中有两个重要的代码部分，即 #define 语句设置的地址基址和系统存储器映射。

```
#define APP_BASE 0x00000000
#define RAM_BASE 0x20000000

/* 系统存储器映射 */

MEMORY
{
     /* 应用程序存储在内部闪存中并通过其执行 */
     FLASH (RX) : origin = APP_BASE, length = 0x00040000
     SRAM (RWX) : origin = RAM_BASE, length = 0x00008000
}
```

为了准备与引导加载程序搭配使用的 *hello*，请首先更新 **APP_BASE** 存储器地址，体现 *bl_config* 中的 **APP_START_ADDRESS**。引导加载程序会占用从 0x0000.0000 到 **APP_BASE** 的存储空间，因此必须更新系统存储器映射中的 *length* 字段，以体现应用程序可用的闪存空间。length 字段是总器件闪存减去引导加载程序所需的空间。在本例中，*0x0004.0000 – 0x0000.2800* 等于 *0x0003.D800*。一种能最大限度地减小误差的简单方法是直接从定义的 *length* 值中减去 **APP_BASE**。

```
#define APP_BASE 0x00002800
#define RAM_BASE 0x20000000

/* 系统存储器映射 */

MEMORY
{
     /* 应用程序存储在内部闪存中并通过其执行 */
     FLASH (RX) : origin = APP_BASE, length = 0x00040000 - APP_BASE
     SRAM (RWX) : origin = RAM_BASE, length = 0x00008000
}
```

完成这些更改后，保存更新的 .cmd 文件，然后重新编译工程，在 Debug 文件夹中生成更新的 .out 和 .bin 文件。现在，*hello* 工程已准备就绪，可通过 *boot_serial* 引导加载程序加载。

使用 Code Composer Studio 的最后一步是使用 *boot_serial* 工程对目标器件进行编程。由于引导加载程序没有 *main* 函数，IDE 将不提供在编程后运行固件的选项，这意味着引导加载程序已成功编程。

6.2 如何使用 LM 闪存编程器引导加载

TI 提供的 LM 闪存编程器实用程序工具可以通过 UART、USB 和以太网接口支持 TM4C 器件的引导加载。为了使用引导加载工具，请选择“Configuration”选项卡中“Quick Set”部分的“Manual Configuration”选项（图 6-1）。选择“Manual Configuration”后，可在“Interface”部分选择 *Serial (UART)*、*Ethernet* 和 *USB DFU* 这三个引导加载选项中的一个。

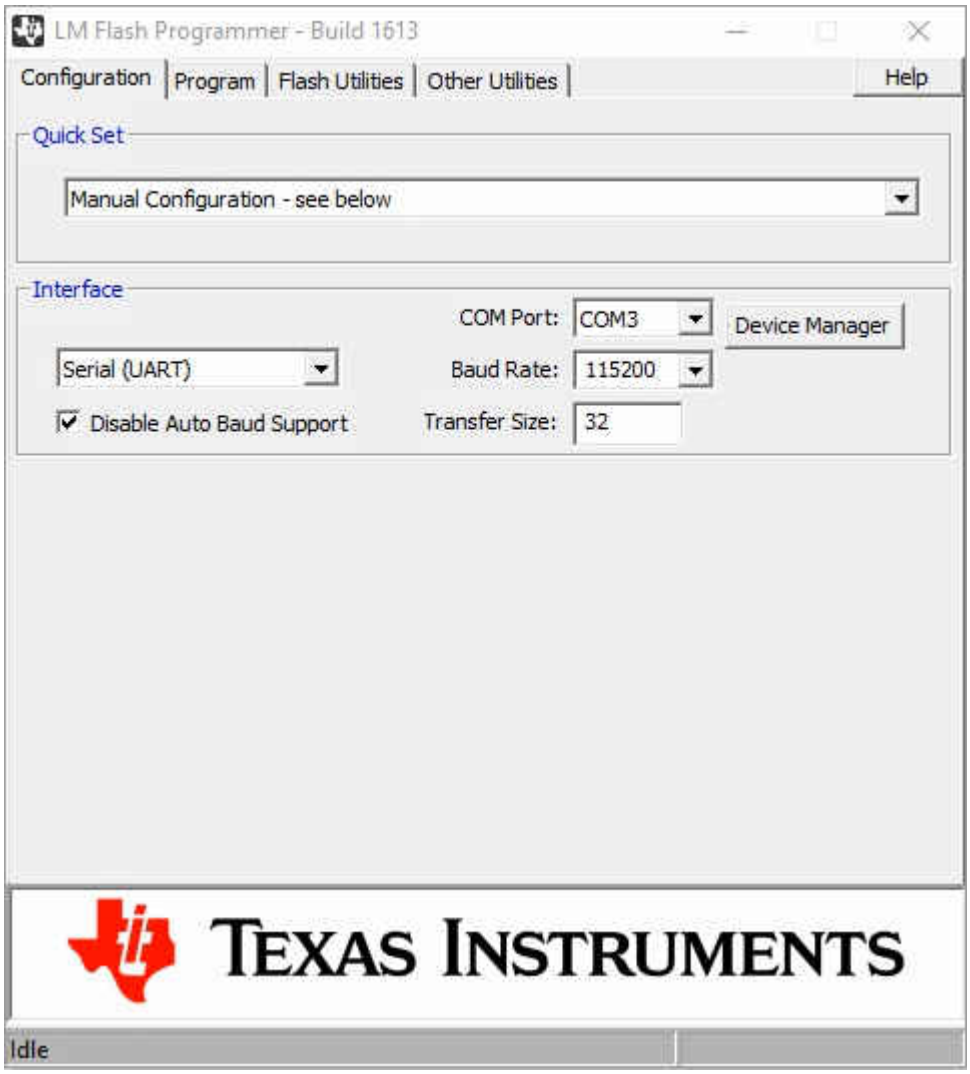

图 6-1. LM 闪存编程器手动配置

对于以太网引导加载程序，应务必确保输入正确的客户端 MAC 地址。如果使用 *USB DFU* 模式，TM4C 微控制器必须处于这样的状态：USB 引导加载程序正常运行，以便 PC 可以识别 USB DFU 端口。如果列表中未显示任何器件，请在 TM4C 微控制器上调用 USB 引导加载程序，然后使用刷新按钮触发 LM 闪存编程器以再次搜索 USB DFU 接口。

设置完所需的接口后，转至“Program”选项卡以选择二进制文件进行编程。浏览本地文件系统以查找工程的 .bin 文件（图 6-2）。该文件始终位于 CCS 工程的 Debug 文件夹中。

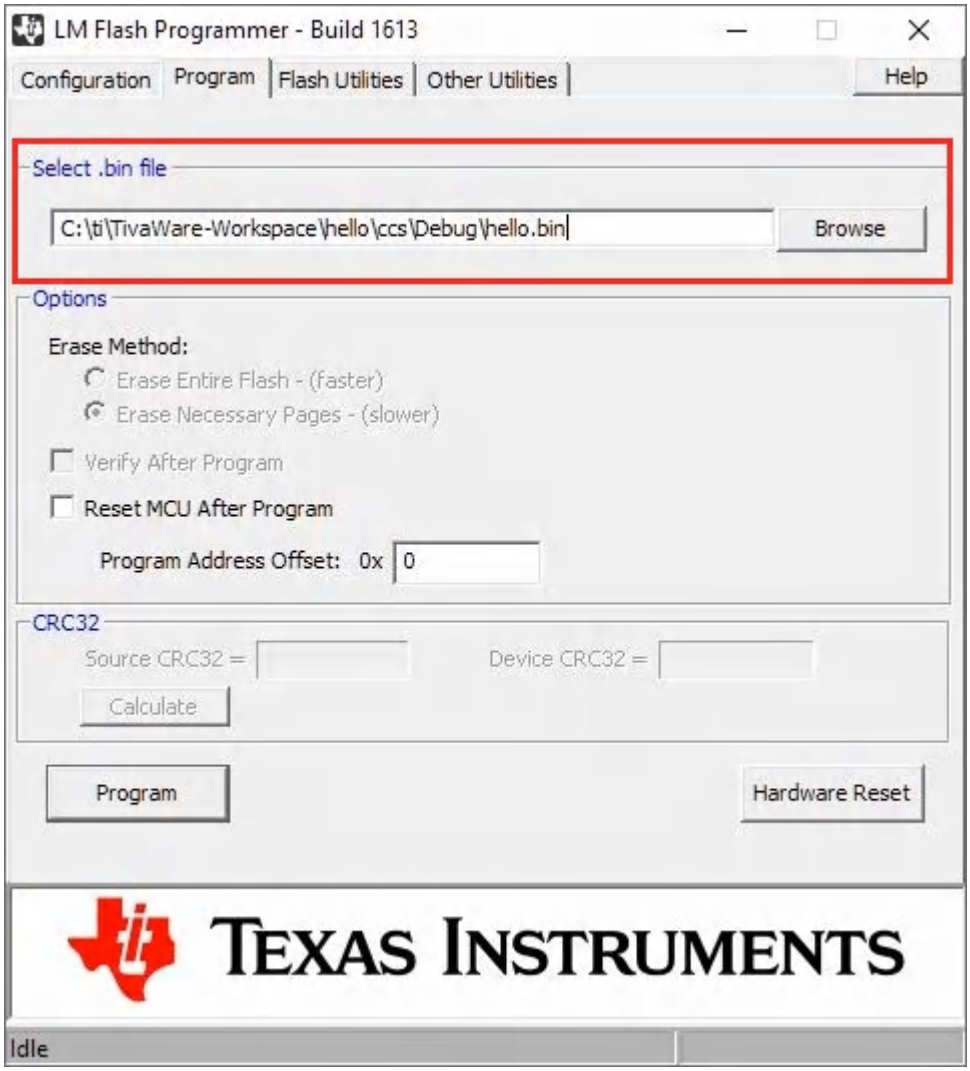

图 6-2. 浏览 .bin 文件

选定文件后，设置正确的 *Program Address Offset*（图 6-3）。这等同于 **APP_START_ADDRESS**。完成上述步骤后，按下“Program”按钮以触发引导加载流程并将应用程序固件上传至器件。

图 6-3. 设置 Program Address Offset 值

在设计功能完整的嵌入式代码工程时，必须将很多要素整合在一起，因此很容易遗漏一两个关键详细信息。本节将介绍嵌入式系统中常见的问题以及如何避免这些问题。

7.1 栈/堆设置和栈溢出

若要确保程序不易受到栈溢出的影响，需要确保程序拥有充足的栈空间，这点非常重要。程序栈位于 RAM 内存中，用于在程序执行过程中存储当前寄存器值和返回地址等信息。栈溢出是一个编程错误，是指栈指针超出了为程序栈分配的空间。出现该错误时，栈读回的数据可能已被应用程序覆盖，并且数据损坏会导致程序崩溃。

若要调整 Code Composer Studio 工程的栈设置，请打开“Project Properties”，然后转到“Build”→“ARM Linker”→“Basic Options”。在列表底部，有一个“Set C system stack size”框（图 7-1）。为了确保对齐内存，请务必为栈设置使用 4 字节的倍数。

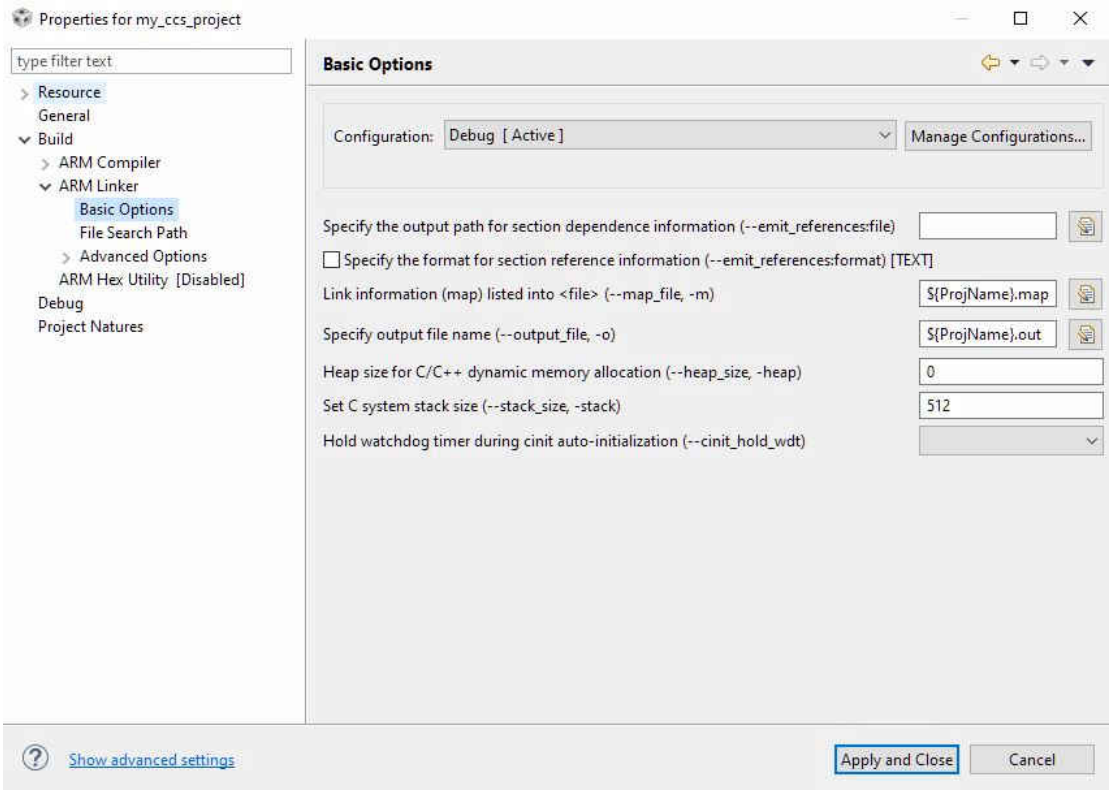

图 7-1. CCS 工程属性中的堆栈设置

另一个修改栈大小的地方是工程的链接器命令文件 (.cmd)。在文件底部，应该有一个 *__STACK_TOP* 条目，可以通过编辑它后面的数值来调整栈大小。

```
__STACK_TOP = __stack + 512;
```

对于使用动态存储器分配的程序（如 *malloc*），以及在 RTOS 中使用动态创建的任务时，还需要分配堆存储器。堆是为支持动态存储器分配而保留的存储器中的一个独立段。可以直接在栈选项的上方找到 Code Composer Studio 工程的堆设置“Heap size for C/C++ dynamic memory allocation”。在 TivaWare 示例中，此项通常默认设置为 0。

7.2 中断服务例程

作为任何嵌入式程序的基本构建块，中断服务例程 (ISR) 是一个因系统硬件中断而输入的代码块。程序员使用 ISR 的方法不一定都正确，错误使用 ISR 会导致很多系统级错误。本节将介绍一些最佳做法，然后说明如何调试用户遇到的关于 ISR 的几个基本错误。

7.2.1 最佳做法

顾名思义，中断服务例程将中断程序当前执行的操作，以执行例程内的代码。在最简单的程序中，无论 ISR 是如何编写的，行为通常都是无害的，但是随着程序规模和范围增加，编写高效的 ISR 对于更大程度地缩短中断典型程序执行所需的时间越来越重要。

最好使用 ISR 快速接收和存储数据或获取和发送数据，并/或体现程序其余部分的状态变化。接收或发送数据时，目标应该是处理来自仅触发 ISR 的硬件外设的数据，并使用标志或状态变化来通知主要程序应用数据已经过处理，以便主程序控制该数据在 ISR 外部执行哪些操作。

关于标志和最佳实践，使用 ISR 内部的全局变量作为标志、计数器或任何其他将在 ISR 内部修改的变量时，必须将全局变量声明为 **volatile** 变量。如果未使用 **volatile** 关键字，编译器可以确定在 ISR 中未使用该变量，并在 ISR 外部对它进行优化。这样，当 ISR 执行以及程序未正常运行时，可以防止修改变量。使用中断时，全局变量缺少 volatile 声明是常见的编程错误之一，因此在代码审核清单中应始终包含此项。

导致 ISR 问题的主要常见错误包括：在 ISR 中进行数据处理、在 ISR 中运行阻止快速退出的循环或者使用 **UARTprintf** 等阻塞命令发送数据（在这种情况下，在数据正确发送前，ISR 无法退出）。其中每个错误都会导致 ISR 无法快速执行，并造成影响系统性能的延迟，在每种情况下，都应该在主程序执行中处理这些问题。

如果系统出现异常行为，并怀疑这个问题是 ISR 速度过慢造成的，最简单的方法是使用示波器评估 ISR 的持续时间。通过在进入和退出 ISR 时切换 GPIO，可以评估每次执行 ISR 所用的时间。如果 ISR 有多个分支，可添加附加 GPIO，以便弄清楚正在执行哪个分支。

7.2.2 TivaWare 矢量表和 IntDefaultHandler

在每个工程中包含的用于每个编译器的 TivaWare *startup* 文件中，微控制器的中断矢量表有一个数据部分。将根据器件中提供的中断填充矢量表条目，中断可在器件数据表的*异常类型*部分中找到。*startup* 文件中的备注指示哪一行对应于哪种硬件中断。然后将在器件内部映射此矢量表，将应用程序内部的所有 ISR 链接到 TM4C 微控制器上的嵌套矢量中断控制器 (NVIC)。

矢量表开始处由默认 ISR 处理程序填充。除了用于对器件执行软重置的 **ResetISR**，每个其他默认 ISR 处理程序都包含一个空的 while(1) 循环。**FaultISR** 表明 MCU 中出现故障，需要通过调试来检测故障点。TI 提供了一份重点介绍软件故障诊断的应用报告，其中显示了系统故障的调试流程。处理程序收到 NMI 时，将输入 **NmiSR**。每个其他中断都与包罗万象的 **IntDefaultHandler** 相关联。卡在 **IntDefaultHandler** 中的程序表明已启用中断，但 ISR 处理程序并未正确地与 NVIC 矢量表关联。

可以使用两种方式将应用程序 ISR 与 NVIC 矢量表中的合适条目相关联。第一种方式是使用 **IntRegister** API，它会将存储在闪存中的矢量表复制到 RAM 中，然后在代码执行过程中添加新的 ISR 关联。这样，就能通过动态方法创建新的 ISR 关联，因而灵活性更高。第二种方法是在 *startup* 文件中为应用程序的 ISR 处理程序手动添加 **extern** 调用，并用 ISR 处理程序代替目标外设的 **IntDefaultHandler** 实例。这是一种在编译时生成关联的静态方法，它的优势包括减小内存使用量和减少代码执行，因此这种方法通常更受欢迎。

为了将由应用程序定义的 ISR 处理程序添加到 *startup* 文件中，请在声明前使用 **extern** 关键字。以下代码来自 EK-TM4C123GXL LaunchPad 上的 *interrupts* 示例。如下所示，每个 ISR 处理程序都在声明的其余部分之前添加了 **extern** 关键字。

```
//*****************************************************************************
//
// External declarations for the interrupt handlers used by the application.
//
//*****************************************************************************
extern void IntGPIOa(void);
extern void IntGPIOb(void);
extern void IntGPIOc(void);
```

将 ISR 处理程序正确添加到 *startup* 文件后，查找与 ISR 对应的矢量表中的外设。例如，通过搜索矢量表查找端口 A、B 和 C 的 GPIO 外设，可以发现矢量图中的以下区段。

```
    IntDefaultHandler,                      // The PendSV handler
    IntDefaultHandler,                      // The SysTick handler
    IntDefaultHandler,                      // GPIO Port A
    IntDefaultHandler,                      // GPIO Port B
    IntDefaultHandler,                      // GPIO Port C
    IntDefaultHandler,                      // GPIO Port D
    IntDefaultHandler,                      // GPIO Port E
    IntDefaultHandler,                      // UART0 Rx and Tx
```

并不是所有外设都是按顺序分组的。使用 Ctrl+F 搜索文件是浏览整个表格的快捷方法。

使用由目标应用程序定义的所需 ISR 处理程序代替中断矢量表中每个对应条目的 **IntDefaultHandler**，然后保存文件。

```
    IntDefaultHandler,                      // The PendSV handler
    IntDefaultHandler,                      // The SysTick handler
    IntGPIOa,                               // GPIO Port A
    IntGPIOb,                               // GPIO Port B
    IntGPIOc,                               // GPIO Port C
    IntDefaultHandler,                      // GPIO Port D
    IntDefaultHandler,                      // GPIO Port E
    IntDefaultHandler,                      // UART0 Rx and Tx
```

7.3 TivaWare 硬件头文件

合并示例或尝试将新功能添加到现有工程中之后，编译代码时发生的常见问题是出现诸如“未解析符号”错误等构建错误，如 节 3.1 所述。导致出现构建错误的情况有很多，该节仅介绍了一些基于 TivaWare API 的基本错误。在使用难以跟踪的基于硬件的声明时，也可能出现此类问题。

TM4C 硬件访问宏、寄存器偏移和硬件位字段的声明在一系列硬件头文件中提供。这些文件位于 TivaWare SDK 的 *inc* 文件夹中，文件名以“hw_”前缀开头。然后，TivaWare API 使用这些声明在寄存器编程级控制 MCU。尽管 TivaWare 为每种外设提供一系列 API，但在有些情况下，比如要优化进程速度或者在 API 未提供正确设置时，可能需要一些寄存器级编程调用。

大多数硬件头文件是特定于外设的，并包含与特定外设的寄存器相关的信息（偏移和位字段）。在接收信息（如需要解析的外设状态标志）时，应用中偶尔需要这些文件。如果添加了附加代码，并且未解析符号似乎是特定外设中的定义，请检查工程中是否包含外设的硬件头文件，或在 TivaWare 中搜索 *inc* 文件夹，以查看哪些文件包含该定义。

除了外设特定文件，有四个附加硬件头文件可提供更多通用声明：

必须在每个 TivaWare 工程中使用 *hw_memmap.h* 文件，因为它为访问的所有外设提供存储器基地址。这是唯一一个必需强制使用的硬件文件。

为了在 NVIC 矢量表中启用外设 ISR，*hw_ints.h* 文件包含所需的定义。使用 ISR 的任何 TivaWare 工程都需要此文件，最好始终包含此文件。

hw_nvic.h 文件为关于 NVIC 的一切提供定义。它包含在每个 TivaWare 工程的 *startup_ccs* 文件中，但如果应用程序代码需要使用任意 NVIC 特定定义（如使用引导加载程序应用时），那么还必须在应用程序文件中包含它。

hw_types.h 文件提供用于直接修改寄存器的宏命令。只有在应用程序代码中直接修改寄存器时，才需要使用此文件。

7.4 ROM 和 MAP TivaWare 前缀

为了更大限度地减少闪存空间，TM4C 微控制器将 TivaWare 的 DriverLib 加载至 ROM 存储器。然而，在 ROM 中包含的 DriverLib 是一个旧版本，因此必须使用最新的 TivaWare DriverLib 通过闪存执行任何更新的函数或新函数。为了更轻松地了解 ROM 中加载的哪些函数是最新的，以及必须从闪存执行哪些函数，TivaWare 包含一个映射文件，用于确定是否使用 ROM 函数或闪存函数。在此设置中，每个 DriverLib 函数可能都有三个函数调用，即泛型函数调用、ROM 前缀函数调用（函数以“ROM_”开始）和 MAP 前缀函数调用（函数以“MAP_”开始）。

对于所有 ROM 前缀函数调用，需要 *rom.h* 头文件，它将从 ROM 存储器映射中选择正确的函数来执行。如果 ROM 中不存在该函数，则会出现编译器错误，指示该函数不可用。在某些情况下，在 *rom.h* 中删除了关联勘误表的旧 ROM 函数以避免滥用，在这些情况下，应使用最新 TivaWare 中的 DriverLib 函数。在其他情况下，ROM 函数可能根本不存在。这就增加了复杂性，不利于程序员使用。

为了简化这一流程，提供了函数调用的第三个选项 - MAP 前缀。所有 DriverLib 函数调用都根据是否存在 ROM 版本在 *rom_map.h* 头文件中定义了一个等效函数。因此，通过使用 MAP 前缀，无需猜测何时使用 ROM 或闪存 DriverLib 函数，同时还可以最大限度地减少 DriverLib 的闪存占用。在应用程序文件中包含 *rom_map.h* 后，只需在所有 DriverLib 函数中添加 MAP 前缀即可利用所有可用 ROM 函数的优势。

This page intentionally left blank.

TEXAS INSTRUMENTS

器件软件
- TivaWare
- 适用于 Tiva-C 的 TI-RTOS

软件工具
- Code Composer Studio
- LM 闪存编程器
- Uniflash

硬件开发套件
- TM4C123x LaunchPad
- TM4C129x LaunchPad
- TM4C129x 开发套件

文档
- 基本信息
  - *TM4C 微控制器产品选型指南*
- 软件 - TivaWare
  - *TivaWare™ 外设驱动程序库用户指南*
  - *TivaWare™ USB 库用户指南*
  - *TivaWare™ 启动加载程序用户指南*
- 软件 - 其他
  - *Stellaris® 微控制器软件故障诊断应用报告*
  - *适用于 TivaC 的 TI-RTOS 2.20 入门指南*
  - *TI-RTOS 2.20 用户指南*
- 硬件设计应用报告
  - *Tiva™ C 系列的 TM4C123x 系列微控制器系统设计指南应用报告*
  - *Tiva™ C 系列的 TM4C129x 系列微控制器系统设计指南应用报告*
  - *通过 JTAG 接口使用 TM4C12x 器件的应用报告*
- 硬件用户指南
  - *Tiva™ C 系列 TM4C123G LaunchPad 评估板 EK-TM4C123GXL 用户指南*
  - *Tiva™ C 系列 TM4C1294 互联 LaunchPad 评估套件 EK-TM4C1294XL 用户指南*
  - *Tiva™ TM4C129X 开发板 DK-TM4C129X 用户指南*

- 应用报告
  - *使用 EK-TM4C1294XL LaunchPad 的物联网演示应用报告*
  - *在 EKTM4C123GXL LaunchPad 上使用 Edde Flex CAN 控制器应用报告*
  - *在 EKTM4C1294XL LaunchPad 上使用 Edde Flex CAN 控制器应用报告*
  - *在 TM4C129x 微控制器上使用 I2C 主机功能集应用报告*
  - *在 EK-TM4C123GXL LaunchPad 上使用 USB 主机模式应用报告*
  - *适用于 Stellaris® 系列微控制器的 ADC 过采样技术应用报告*

技术支持

- E2E 论坛
  - 基于 Arm® 的微控制器论坛
- 常见问题解答
  - TM4C 基本信息和常见问题解答
  - TI-RTOS 常见问题解答完整列表
  - 原 TI-RTOS 研讨会
  - TM4C 的第三方支持

修订历史记录

注：以前版本的页码可能与当前版本的页码不同

Changes from Revision * (March 2021) to Revision A (August 2022)	Page
• 更新了整个文档中的表格、图和交叉参考的编号格式。	7
• 更新了 节 6.2。	29
• 更新了节 7.2.2。	32

This page intentionally left blank.

重要声明和免责声明

TI“按原样”提供技术和可靠性数据（包括数据表）、设计资源（包括参考设计）、应用或其他设计建议、网络工具、安全信息和其他资源，不保证没有瑕疵且不做出任何明示或暗示的担保，包括但不限于对适销性、某特定用途方面的适用性或不侵犯任何第三方知识产权的暗示担保。

这些资源可供使用 TI 产品进行设计的熟练开发人员使用。您将自行承担以下全部责任：(1) 针对您的应用选择合适的 TI 产品，(2) 设计、验证并测试您的应用，(3) 确保您的应用满足相应标准以及任何其他功能安全、信息安全、监管或其他要求。

这些资源如有变更，恕不另行通知。TI 授权您仅可将这些资源用于研发本资源所述的 TI 产品的应用。严禁对这些资源进行其他复制或展示。您无权使用任何其他 TI 知识产权或任何第三方知识产权。您应全额赔偿因在这些资源的使用中对 TI 及其代表造成的任何索赔、损害、成本、损失和债务，TI 对此概不负责。

TI 提供的产品受 TI 的销售条款或 ti.com 上其他适用条款/TI 产品随附的其他适用条款的约束。TI 提供这些资源并不会扩展或以其他方式更改 TI 针对 TI 产品发布的适用的担保或担保免责声明。

TI 反对并拒绝您可能提出的任何其他或不同的条款。

邮寄地址：Texas Instruments, Post Office Box 655303, Dallas, Texas 75265