应用报告
ZHCA506B–January 2012–Revised November 2012

在 Code Composer Studio™ 中，使用针对Stellaris®的 CMSIS DSP 库

Jordan Wills *Stellaris® Microcontrollers*

摘要

本应用报告描述了建立 ARM® CMSIS DSP 库所需的过程，建立此库所使用的工具为 Code Composer Studio™ v5 (CCSv5)。 这份报告还说明了如何使用 CCSv5 来建立、运行和验证 CMSIS 软件包中包含的 11 个 ARM DSP 示例项目。 本文档中描述的项目附属和源代码可从以下德州仪器网站的位置内下载：http://www.ti.com/lit/zip/spmc017。

内容

1 简介

很多基于微控制器的应用可从高效数字信号处理器 (DSP) 库的使用中受益。 为了达到这一目的，ARM 已经开发出了一组被成为 CMSIS DSP 库的函数，它与所有 ARM Cortex™-M3 和 Cortex-M4 处理器兼容并且被专门设计成使用 ARM 汇编指令来快速且方便地处理多种复杂的 DSP 函数。 当前，ARM 提供用于他们的 Keil µVision IDE 中的示例项目，这些项目用来显示如何建立它们的 CMSIS DSP 库并在一个 Cortex-M3 或 Cortex-M4 处理器上运行它们。 这份应用报告描述了在 Code Composer Studio 版本 5 内建立相同 DSP 库所需的步骤，以及正在一个 Stellaris® LM4F232 微控制器上运行这些示例应用的步骤。

2 CMSIS DSP 库

要建立 CMSIS DSP 库，从 ARM CMSIS 网站内下载并提取源代码：http://www.onarm.com/。 针对 DSP 库和示例项目的源代码位于这个目录中：

CMSIS-<*版本*>/CMSIS/DSP_Lib

一个对 DSP 库的完整说明，其中包括对所有示例、所使用的数据结构和针对每个可用函数的 API 的说明，位于以下位置内 ARM 所提供的文档：

CMSIS-<*版本*>/CMSIS/Documentation/DSP_Lib/html/index.html.

如果 ARM 发布一个对 CMSIS 的更新，为了提供对新功能的支持并改正 ARM 在 CMSIS 源代码中发现的任何错误，您也许需要下载且安装一个对 CMSIS DSP 库的补丁。 在您从www.onarm.com网站内下载了补丁文件后，按照以下指令进行安装：

1. 将补丁文件解压缩。
2. 导航至补丁包目录将所有在那个目录中发现的文件复制到 CMSIS DSP 库的相应位置。
3. 当被提示时，写覆盖现有文件。

例如，如果补丁目录包含一个位于 CMSIS/DSP_Lib/Source/CommonTables 目录内的名为 arm_common_table.c 的文件，将这个文件复制到您最初 CMSIS 安装的同一目录内 (CMSIS/DSP_Lib/Source/CommonTables)，将已经存在于最初安装目录内的 arm_common_tables.c 写覆盖。

在 CMSIS 源代码已被下载后，您必须下载并运行德州仪器的 SW01291 安装程序。 这个安装程序位于德州仪器的网站http://www.ti.com/lit/zip/spmc017内。 此安装程序包含一组在 Code Composer Studio 中建立和运行 CMSIS DSP 库所需的支持文件。 在您下载了安装程序后，运行安装程序并选择一个提取文件的位置。

3 在 CCSv5 内建立 DSP 库

这个部分描述了如何从源建立 ARM CMSIS DSP 库。 您可以通过使用一个预编译的 .lib（诸如在CMSIS-<*版本*>/CMSIS/Lib/ARM or CMSIS-<*版本*>/CMSIS/Lib/GCC 中找到的一个文件），但是这么做要求改变 CCS 编译器设置以使用与缺省 CCS 设置不同的方法调用浮点函数。 这要求重建所有 .lib 文件，这些文件被使用在具有 DSP 库的项目中，最常见的 DSP 库是 StellarisWare® driverlib 库。 不建议采用这个方法并且此过程未在应用报告中进行说明。

3.1 将要求 CCS 的头文件添加到 DSP 库中

为了使用 Code Composer Studio 编译 CMSIS DSP 库，您必须修改 DSP 库包含文件。 这些包含文件位于 CMSIS-<*版本*>/CMSIS/Include 目录内。 对于 core_cmInstr.h，core_cmFunc,h 和 core_cm4_simd.h 文件，请找到用来确定哪个编译器正在建立库的 #if define statements 块。 将以下代码添加到针对 ICCARM 和 GNU 编译器的语句之间：

```
#elif defined ( __TMS470__ ) /*-------- TI CCS Compiler --------*/
#include <cmsis_ccs.h>
```

此代码高速编译器 CCS（使用 TMS470 编译器）被用来建立代码，所以您必须包含 cmsis_ccs.h 头文件，它定义了调用不同编译器内部函数所需的句法。

您还必须修改 core_cm3.h 和 core_cm4.h 文件。 在针对 ICCARM 和 GNUC 编译器的语句之间，重新在用于确定使用哪个编译器的 #if define statements 块中添加以下的代码行。 这个语句定义了进行直接汇编调用所需的句法。

```
#elif defined ( __TMS470__ )
#define __ASM __asm
```

最后，通过在 string.h 和 math.h 已经被包含在内后添加以下代码块来修改 arm_math.h 文件来重新定义用于 Code Composer Studio 的 __INLINE 宏：

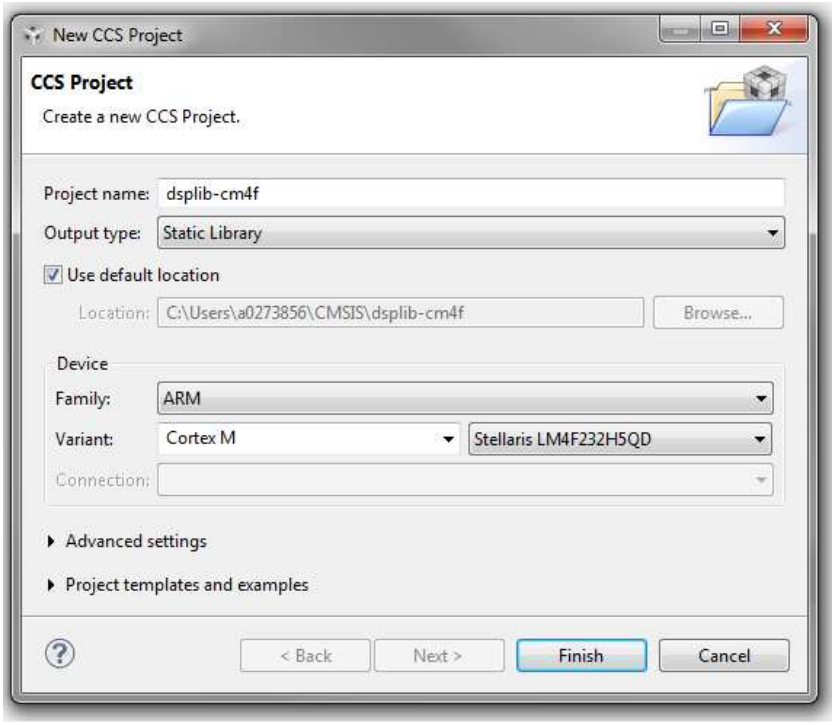
```
#if defined (__TMS470__)
    #undef __INLINE
    #define __INLINE inline
#endif
```

SW01291 安装程序创建一个*Include*目录，此目录包含这些头文件的预先修订的版本，当前作为 CMSIS 2.1 补丁 4。如果您使用预先修订的头文件，您应该重命名最初的 Include 目录，此目录与 CMSIS 一起从 ARM 下载至 Include_backup，然后将 Include 目录从这个应用报告复制到它的位置。

3.2 创建 dsplib 项目

在 CCS 中建立 DSP 库之前，您必须为库创建一个项目。 您可通过完成以下步骤来建立一个项目：

1. 启动 CCSv5 并选择一个空的工作区。
2. 选择 File > New > CCS Project。 新的 CCS 项目窗口显示
3. 在*项目名称*字段中，敲入 dsplib-cm4f（或者，如果使用一个 Cortex-M3 部件的话，敲入 -cm3）
4. 从 Output（输出）下拉菜单中选择 Static Library（静态库）。 项目的位置无关，所以在这个示例中使用缺省位置。
5. 在 Device（器件）区域内，在下拉菜单中进行以下选择：
   - 系列：ARM
   - 变量：Cortex M
   - 部件：Stellaris LM4F232H5QD
6. 单击**Finish**（完成）来创建项目。 dsplib-cm4f 项目出现在 Project Explorer（项目浏览器）中。

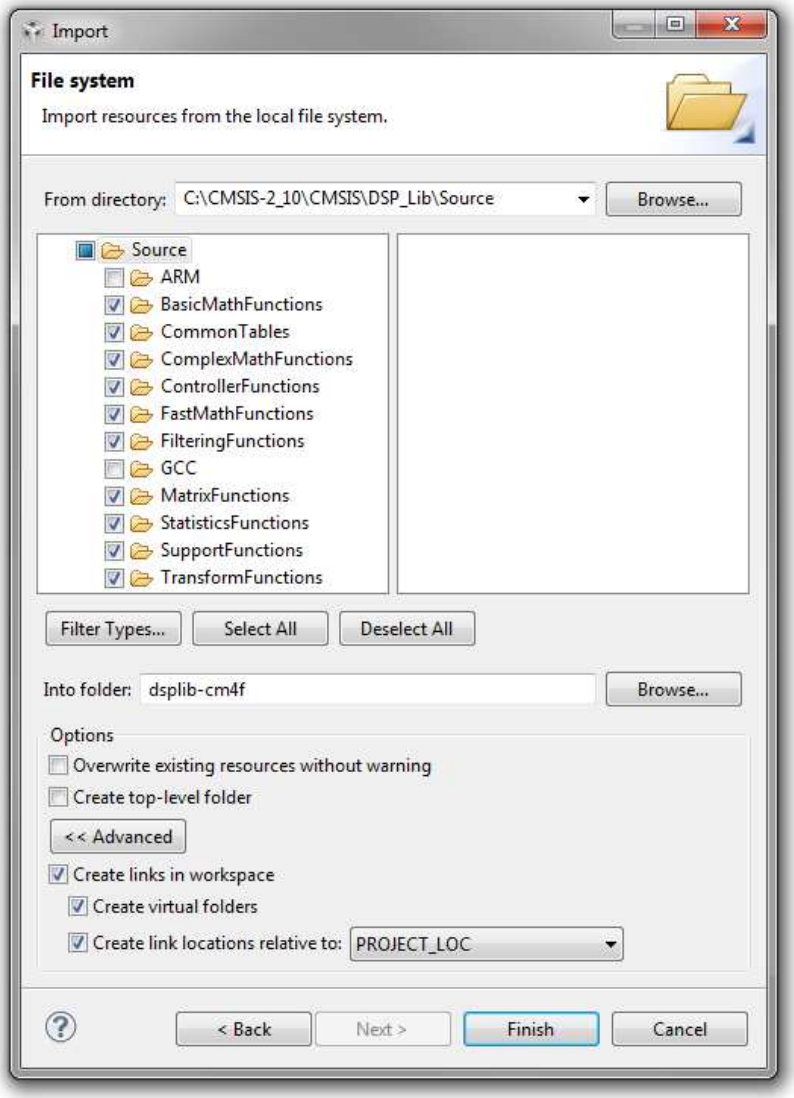


3.3 添加 dsplib 源代码

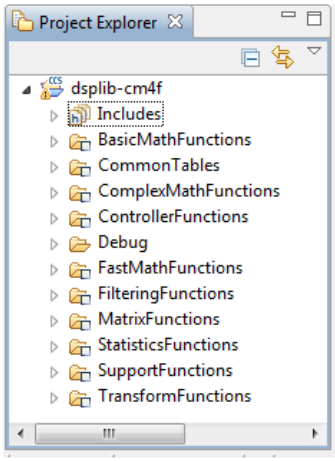
在把 dsplib 源代码添加到项目中，您应该熟悉 CMSIS 库结构。

1. 使用文件浏览器来导航至从 ARM 下载的 .zip 文件被提取的目录中。
2. 导航至 CMSIS-<*版本*>/CMSIS/DSP_Lib/Source/ 目录。 ARM 目录包含在具有 ARM 编译器的 µVision 中建立 DSP 库所需的项目文件， 所有其它目录包含建立目录名称所表示的函数类别所需的源代码。 这个示例将针对 DSP 库的源文件连接至项目工作区。
3. 右键单击 Project Explorer 中的 dsplib-cm4 项目并单击*Import...（导出）*
4. 单击*General*来展开，然后单击*File System（文件系统）*。 单击**Next**。
5. 单击**Browse（浏览）**按钮并导航至 CMSIS DSP 库源代码的位置。
6. 选择您希望导入的文件夹并单击**OK**。
7. 当所选择的文件夹出现在 Import 窗口中时，单击文件夹复选框来选择那个文件夹导入的所有文件。
8. 单击*Source*复选框左侧的箭头并单击 ARM 和 GCC 文件夹来取消选定。
9. 请确保*Into Folder：*文本字段包含您希望导入此文件的 DSP 库项目的名称。
10. 单击*Create top-level folder（创建顶级文件夹）*复选框来取消选定。
11. 单击**Advanced**按钮，然后单击*Create links in workspace（在工作区创建连接）*复选框。
12. 如果*Create link locations relative to:（创建相对连接位置）*复选框未被选中，则单击复选框。
13. 从环境变量的下拉菜单中选择 PROJECT_LOC。 如果下拉菜单中没有列出的变量，选择*Edit Variables...（编辑变量）*并添加一个变量来代表 dsplib 项目文件的位置。

14. 单击**Finish**来将文件夹连接至项目。



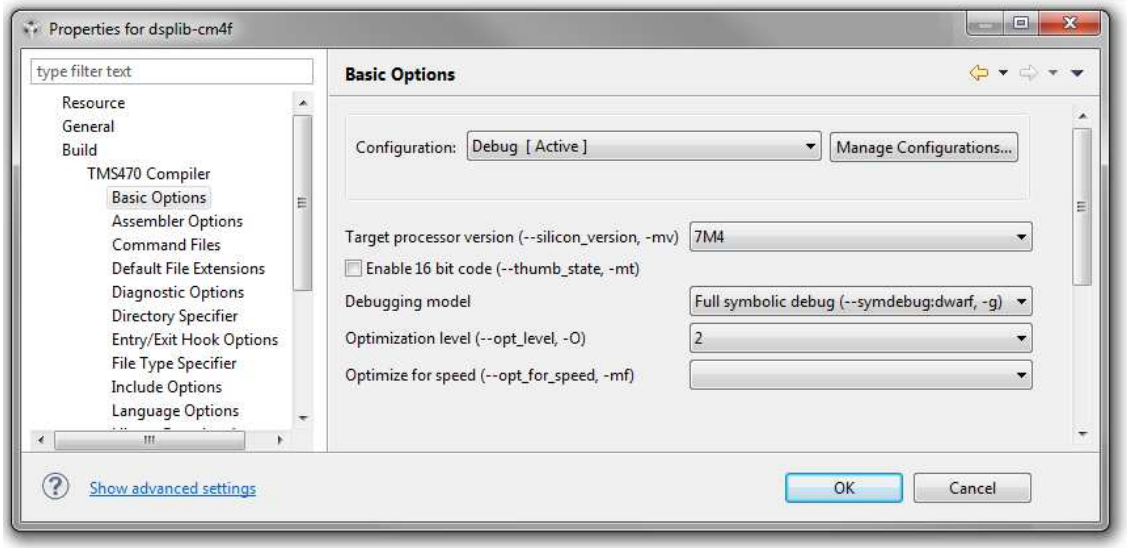
15. 在导入 DSP 源后，检查 Project Explorer 并验证 10 个源目录已被正确地导入到项目工作区中。 您应该在 Project Explorer 窗口中看到这一点。



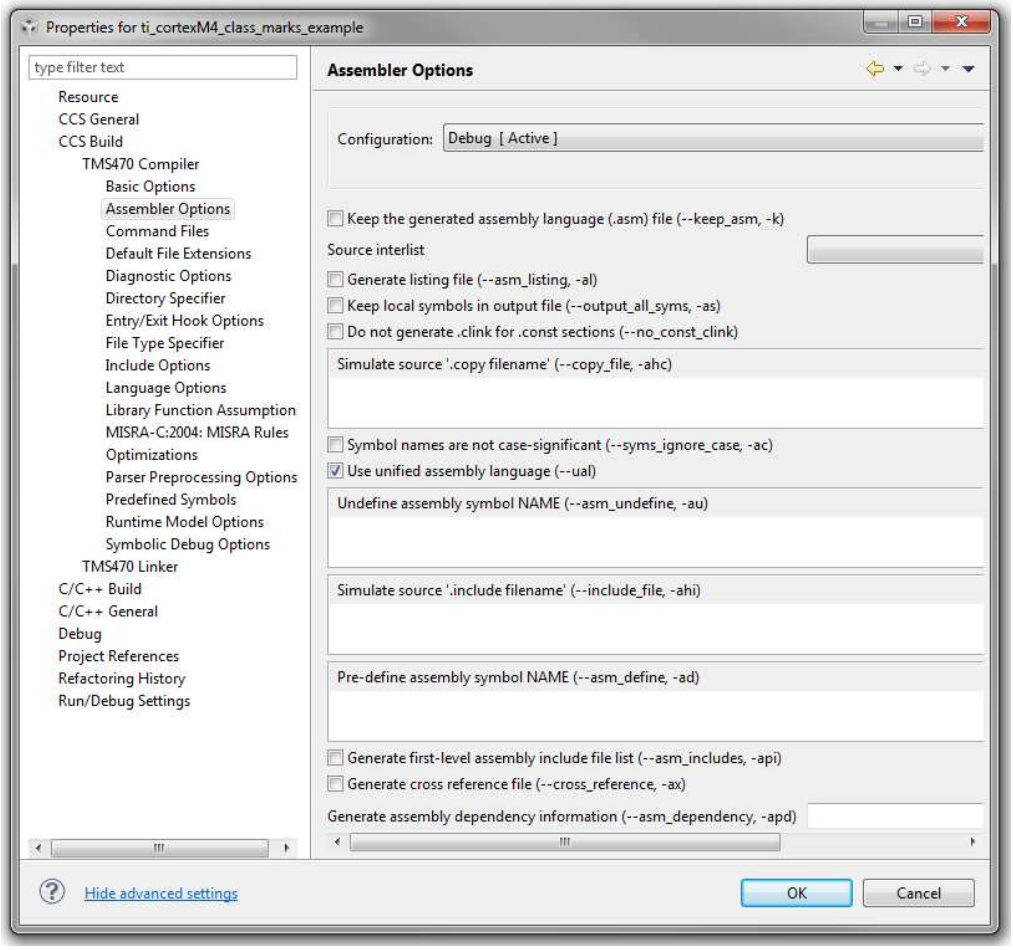
3.4 编辑 dsplib 项目设置。

在连接至所有源文件后，改变以下的缺省 CCS 项目设置。

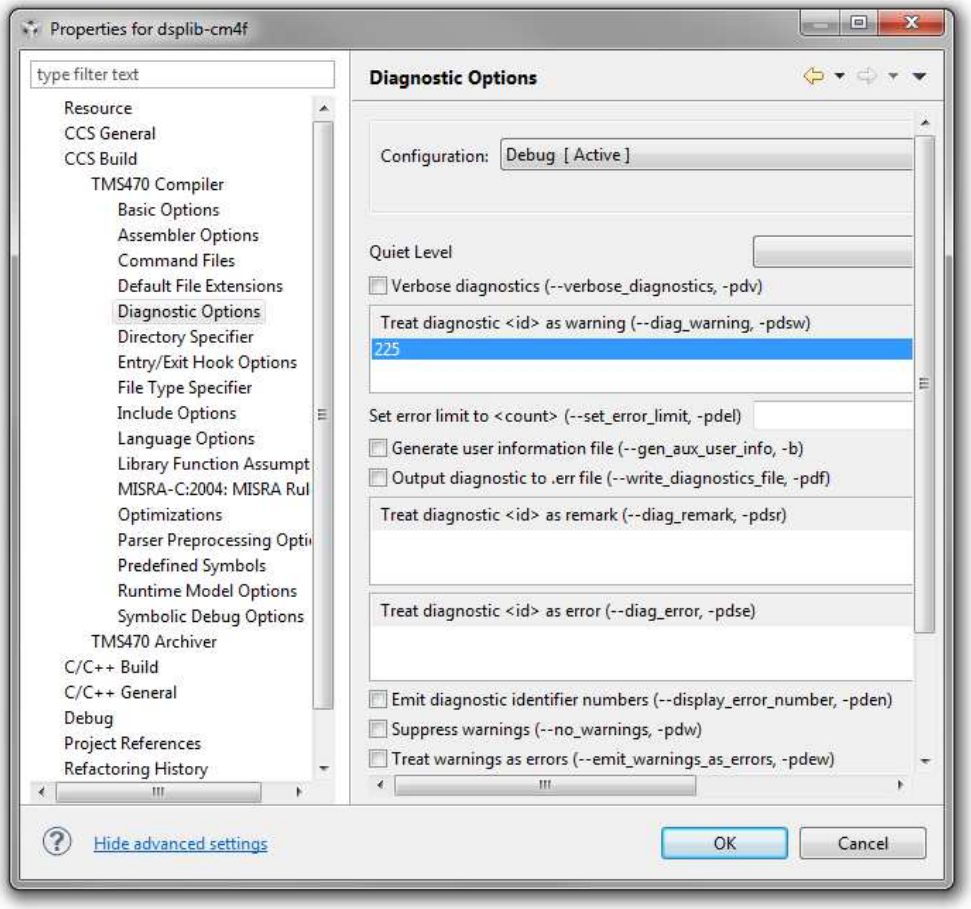
1. 右键单击 Project Explorer 中的 dsplib-cm4f 项目并选择*Properties*（*属性*）。
2. 展开*Build*条目，然后展开*TMS470 编译器*条目。
3. 在 Basic Options（基本选项）面板中，请确认*Target processor version (--silicon_version，-mv)*（*目标处理器版本*）条目与您的处理器相匹配。 对于本示例，目标处理器应该为 7M4。
4. 单击*Optimization level (--opt_level，-O)*（*优化级别*）下拉菜单并选择 2。



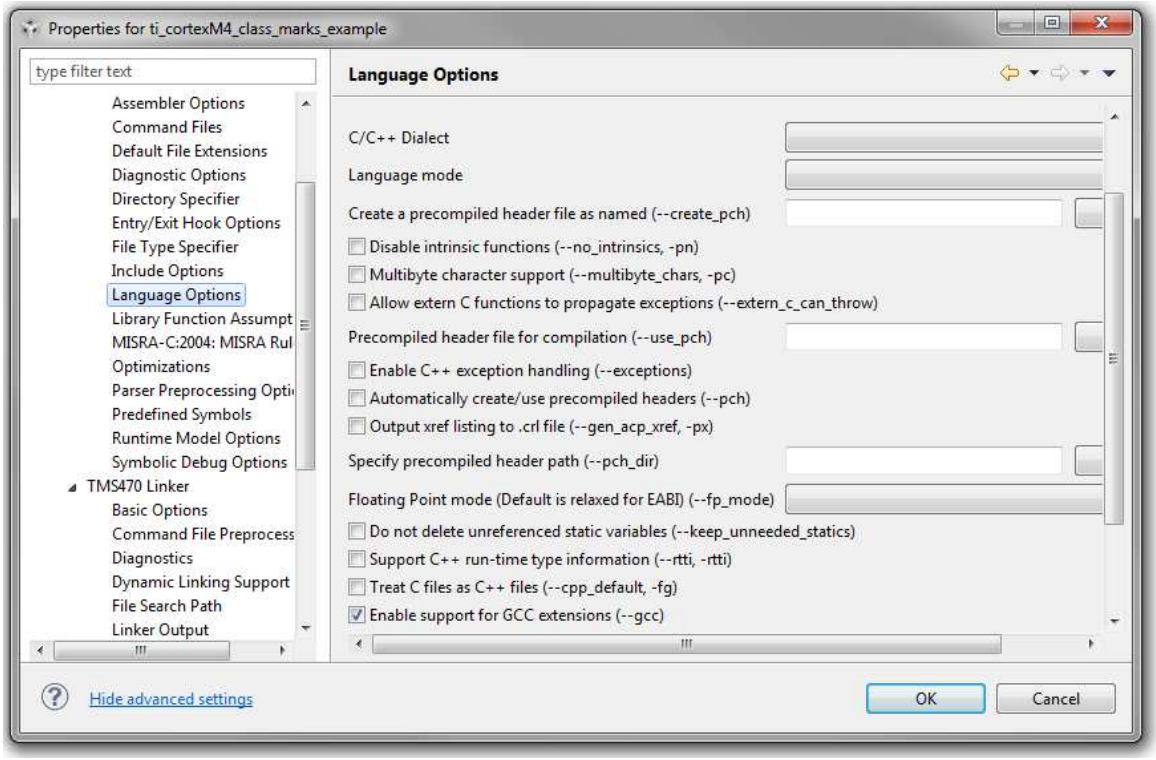
5. 在 Assembler Options（汇编程序选项）面板中，*使用统一汇编语言 (--ual)*复选框来选择。



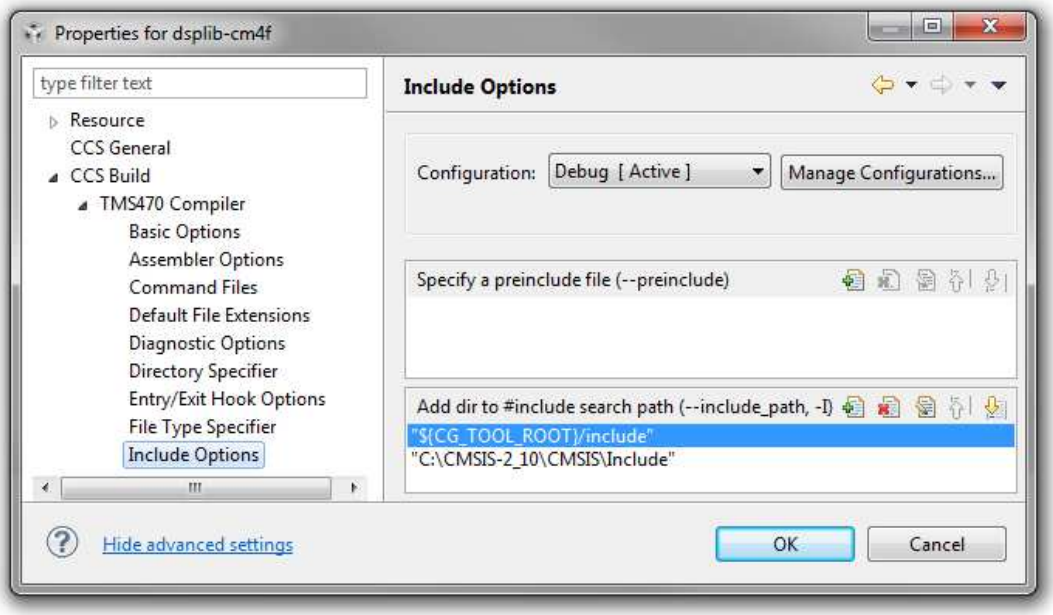
6. 在 Diagnostic Options（诊断选项）面板中，单击*Emit diagnostic identifier numbers (--display_error_number，-pden)*（发出诊断标识符数）复选框来取消选择。



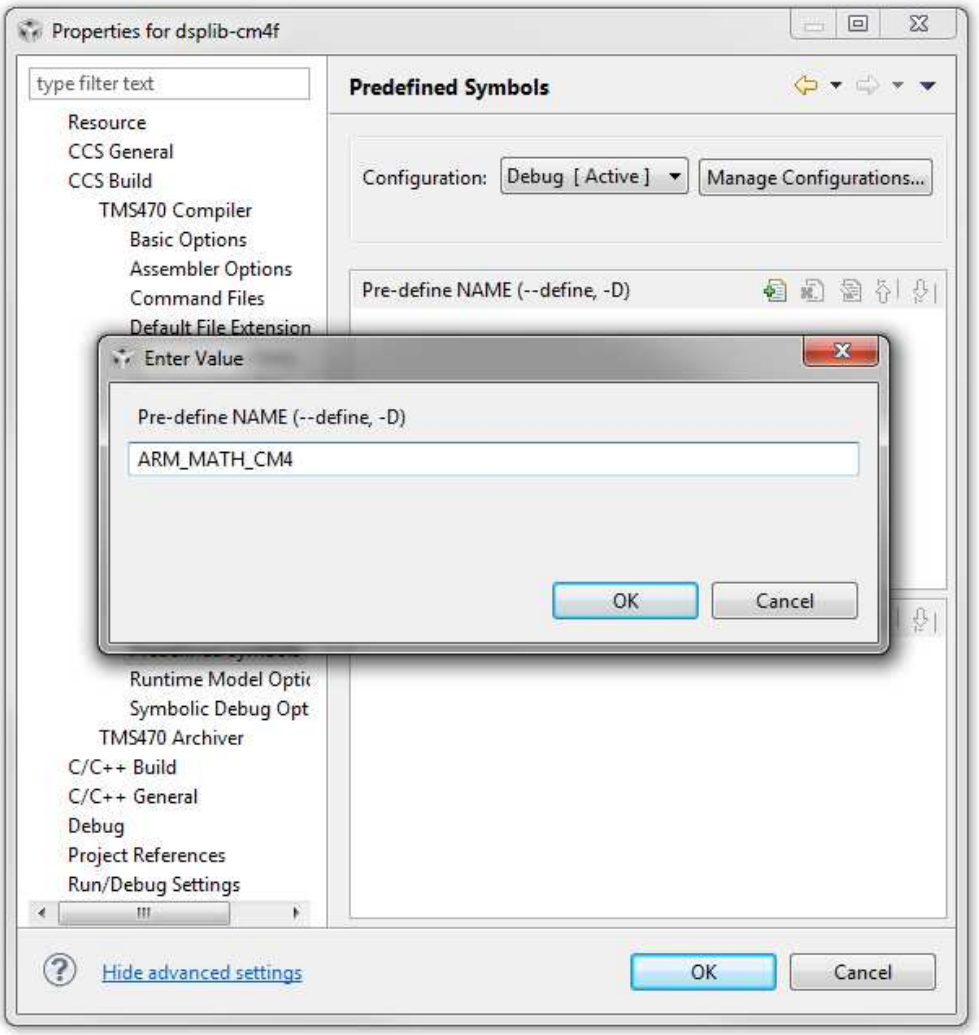
7. 在 Language Options（语言选项）面板中，单击*Enable support for GCC extensions (--gcc)*（*启用针对 GCC 扩展的支持*）复选框来选择。



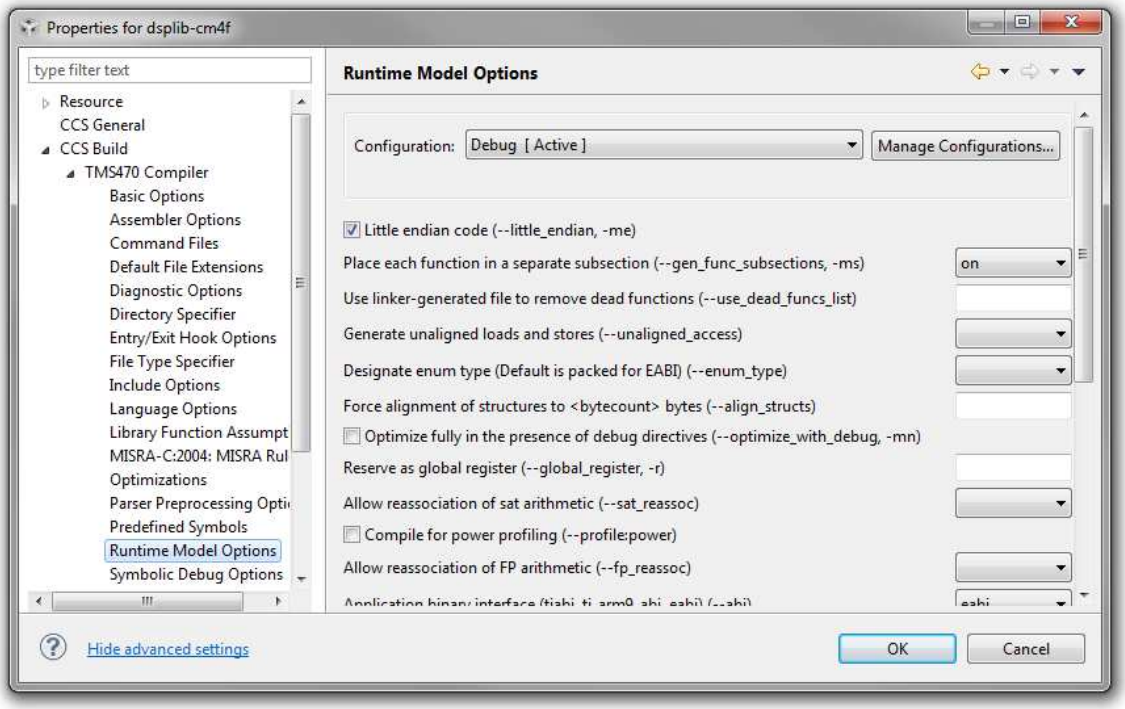
8. 在 Include Options（包含选项）面板中，将 DSP 库 CMSIS-<*版本*>/CMSIS/Include 路径添加到目录列表来添加到 #include search path (--include_path，-I)。



9. 在预先定义的符号面板内，创建一个符号来告诉 DSP 库来使用基于 Cortex-M3 或 Cortex-M4 的数学函数。 单击**Add...**按钮，此按钮位于*Pre-define NAME (--define，-D)*（*预先定义名称*）区域内。 在*Enter Value*（*输入值*）对话框内，在*Pre-define NAME (--define，-D)*字段内敲入 ARM_MATH_CM4 并单击 OK。



10. 在 Runtime Model Options（运行时间模式选择）面板内，从*Place each function in a separate subsection (--gen_func_subsections，-ms)*（*将每个函数放置在一个单独的子部分中*）下拉菜单中选择***on***选项。



3.5 建立 *dsplib* 源代码

通过右键单击 Project Explorer 内的 dsplib-cm4f 来建立 CMSIS DSP 库并选择*Build Project*。根据硬件的不同，这个建立有可能最多花费十分钟来完成。 在建立完成后，在项目工作区的调试文件夹中创建生成的 dsplib-cm4f.lib。

注: 当前 CMSIS 文件结构包含一个位于 CMSIS-<*版本*>/CMSIS/Lib 内的目录，此目录用于存储已编译的库文件。 为了使用这个目录，创建一个 CCS 子目录，然后将用以下步骤创建的 .lib 复制到 CCS 子目录中。

4 ARM 示例项目

ARM CMSIS 下载包含十一个示例项目，这些项目演示了如何使用所包含的 DSP 库函数。 这个部分描述了如何在一个 Stellaris LM4F232 评估板上的 Code Composer Studio v5 创建、编译和运行同样的项目。 您能够轻松修改此指令，使之与其它 Cortex-M4 和 Cortex-M3 微控制器一起运行。

4.1 *创建 ARM 示例项目*

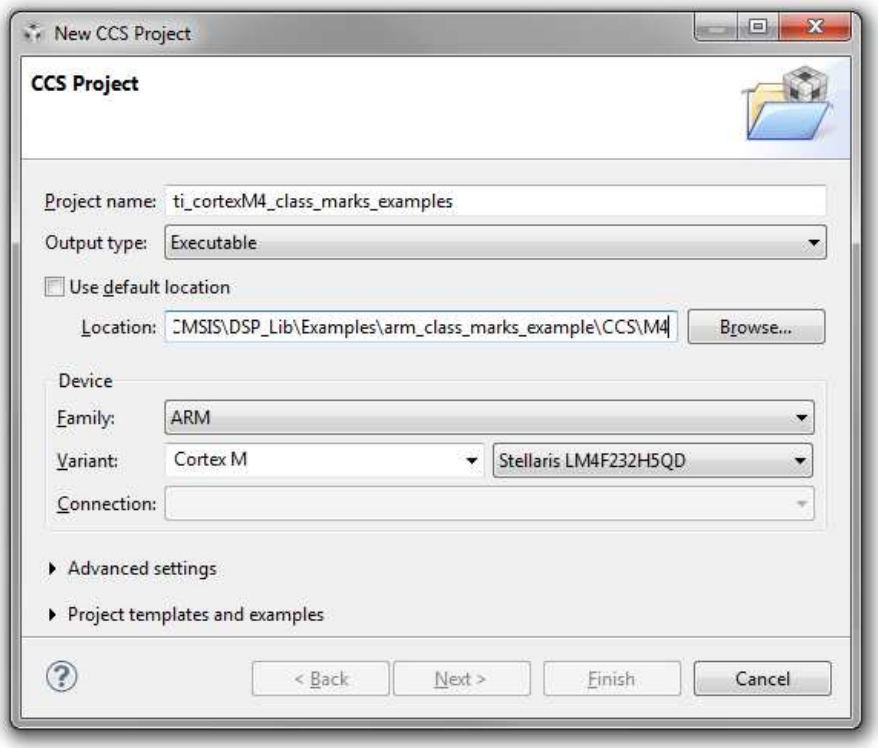
针对所有示例项目的源代码位于以下目录中：

CMSIS-<*版本*>/CMSIS/DSP_Lib/Examples

按照以下步骤来创建 ARM 示例项目：

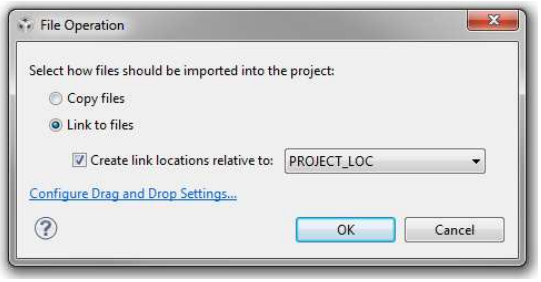
1. 导航至包含将被创建的示例的目录并创建一个名为 CCS 的子目录。
2. 在 CCS 目录中，创建另外一个名为 M4 的子目录。
3. 启动 CCSv5 并选择3.2 节中使用的工作区，*创建 dsplib 项目*，（如果使用一个预先编译的 .lib）或者一个空的工作区。

4. 单击 File > New > CCS project 来显示新的 CCS 项目窗口。
5. 在*Project Name*字段中敲入 ti_cortexM4_<*示例_名称*>_example。
6. 从*Output type*（*输出类型*）下拉菜单中选择可执行。
7. 单击*Use default location*（*使用缺省位置*）复选框来取消选择，然后单击 Browse 按钮来导航至 M4 目录 (CMSIS/DSP_Lib/Examples/<*示例*>/CCS/M4)。
8. 在 Device 区域内，从下拉菜单中做出以下选择：
   - 系列：ARM
   - 变量：Cortex M
   - 部件：Stellaris LM4F232H5QD
9. 单击**Finish**。



4.2 *添加示例源代码*

1. 在项目被创建后，删除随每个空白项目一同生成的缺省 main.c 文件。
2. 右键单击 Project Explorer 内的项目并选择*Add Files...*（*添加文件*）
3. 导航至 CMSIS-<*版本*>/CMSIS/DSP_Lib/Examples/<*example*>目录。
4. 选择 example_f32.c 源文件并单击**Open**。
5. 在 File Operation（文件操作）对话框中，选择**Link to files**（连接至文件）单选框，然后单击*Create link locations relative to:*（*单击相关连接位置*）复选框，并从下拉菜单中选择 PROJECT_LOC。
6. 单击**OK**。



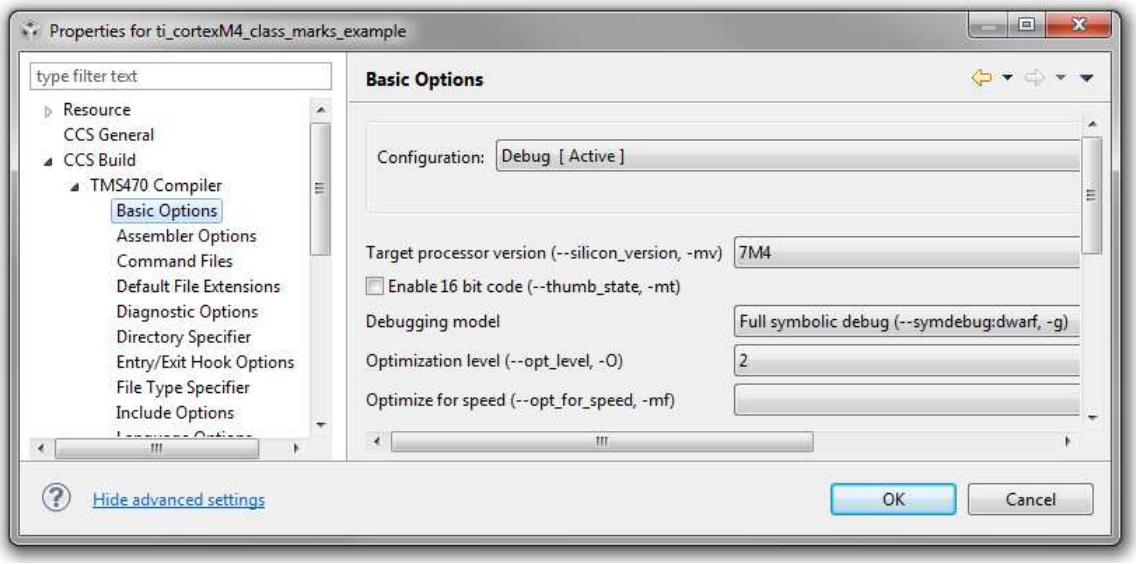
7. 将 math_helper.c 文件连接至项目工作区，这一操作可按照上面列出的用于 example_f32.c 内连接的步骤来完成。 这个文件包含一组由很多示例项目引用的帮助程序函数，并位于这个目录中：

   CMSIS-<*版本*>/CMSIS/DSP_Lib/Examples/Common/Source/math_helper.c

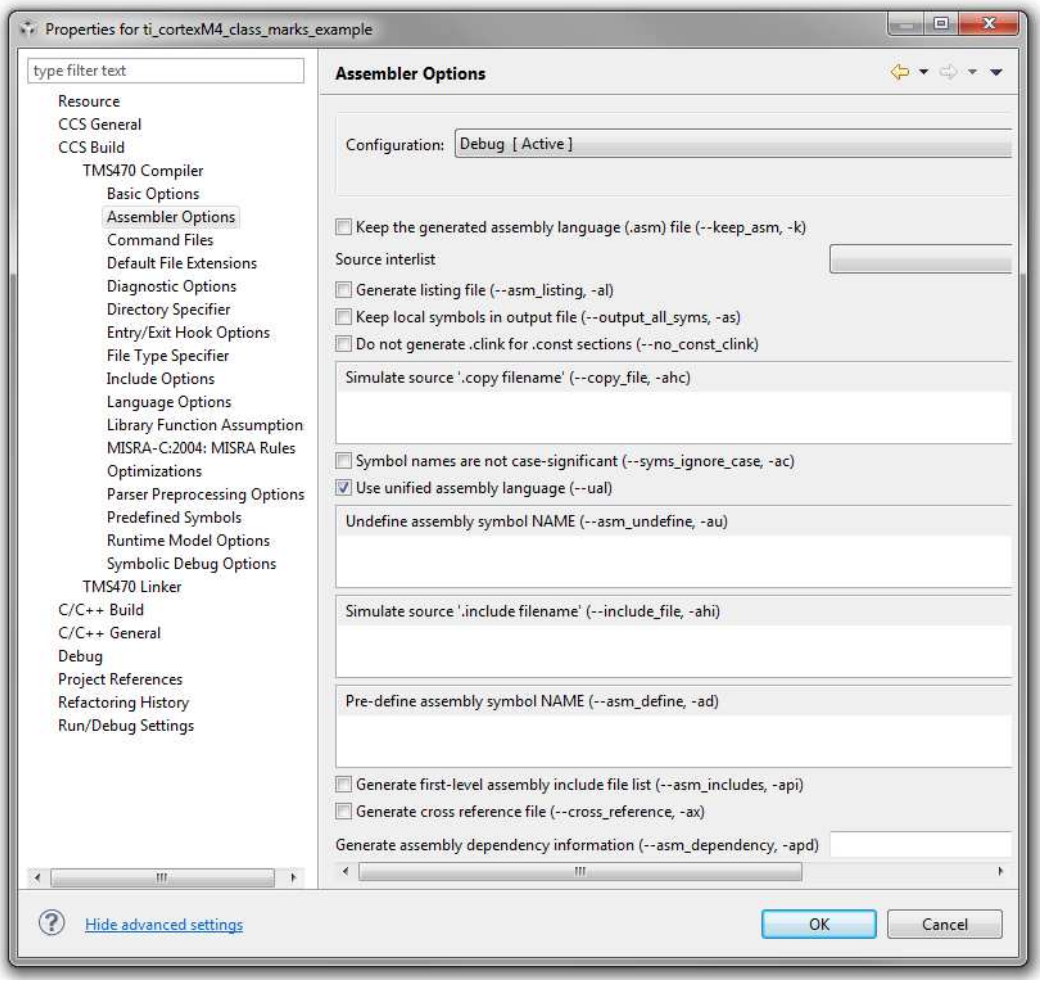
8. 对于包含 arm_<*示例*>_data.c 文件的项目，将此文件连接至工作区。
9. 复制用于 Code Composer Studio v5 的微控制器启动代码文件。 startup_ccs.c 文件位于这个文档所在的同一个 zip 文件内。 将这个文件复制到项目被创建的同一个目录或者使用上面列出的步骤来连接此文件。

4.3 编辑示例项目设置

1. 右键单击 Project Explorer 中的 ti_cortexM4_class_marks_example 项目并选择 Properties。
2. 单击*CCS Build*条目和*TMS470 Compiler*条目。
3. 在 Basic Options 面板中，请确认*Target processor version (--silicon_version，-mv)*下拉选择被设定至正确的处理器。 这个示例使用一个 Cortex-M4 处理器，这意味着应该选择 7M4 选项。
4. 在*Optimization level (--opt_level，-O)*下拉菜单中选择 2。



5. 打开*TMS470 Compiler*条目中的*Assembler Options*条目。 单击*Use unified assembly language (--ual)*复选框来选择。

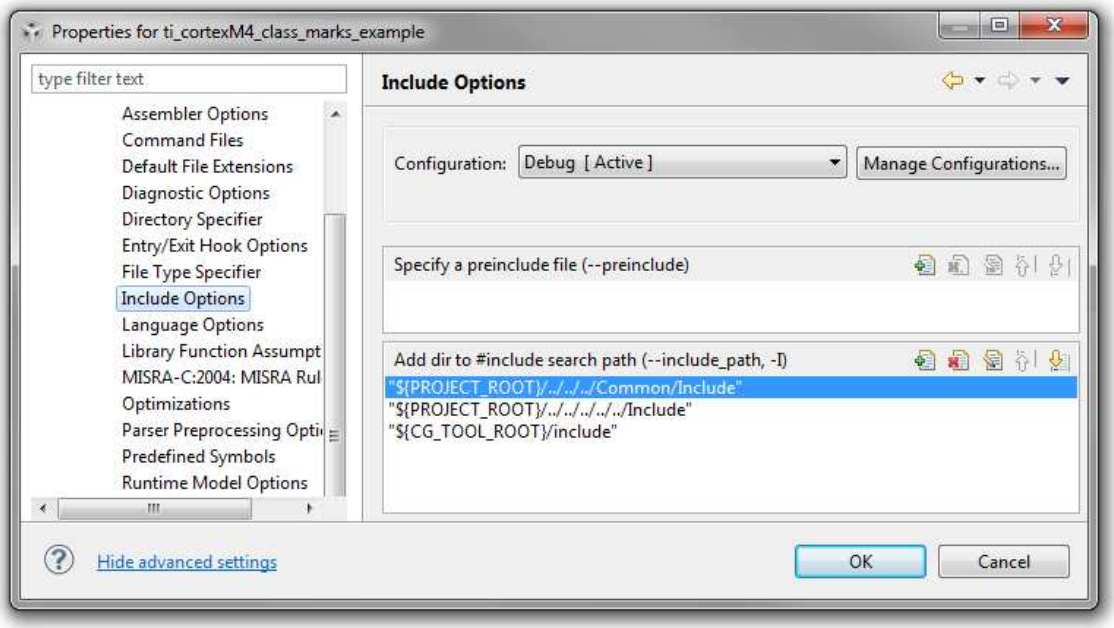


6. 单击*TMS470 Compiler*条目下的*Include Options*条目。
7. 在*Add dir to #include search path (--include_path，-I)*字段中，创建以下两个新条目：

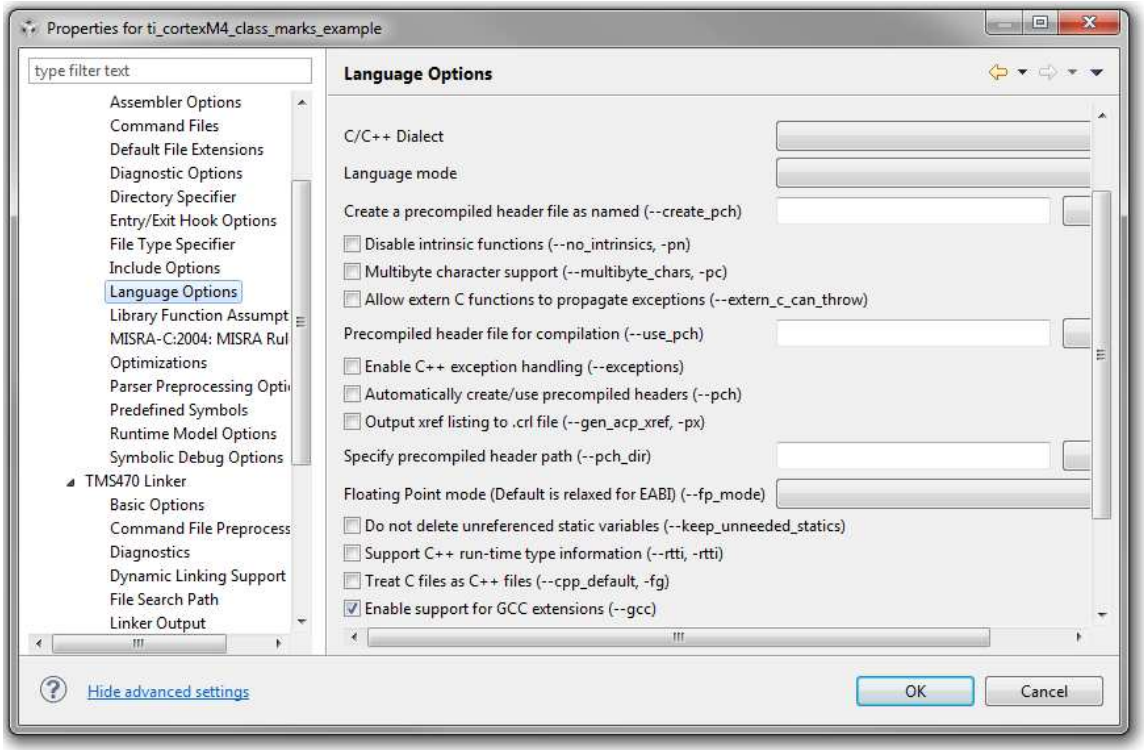
   ARM 示例代码的公共目录：$(PROJECT_ROOT)/../../../Common/Include

   DSP 库的 Include 目录：$(PROJECT_ROOT)/../../../../../Include

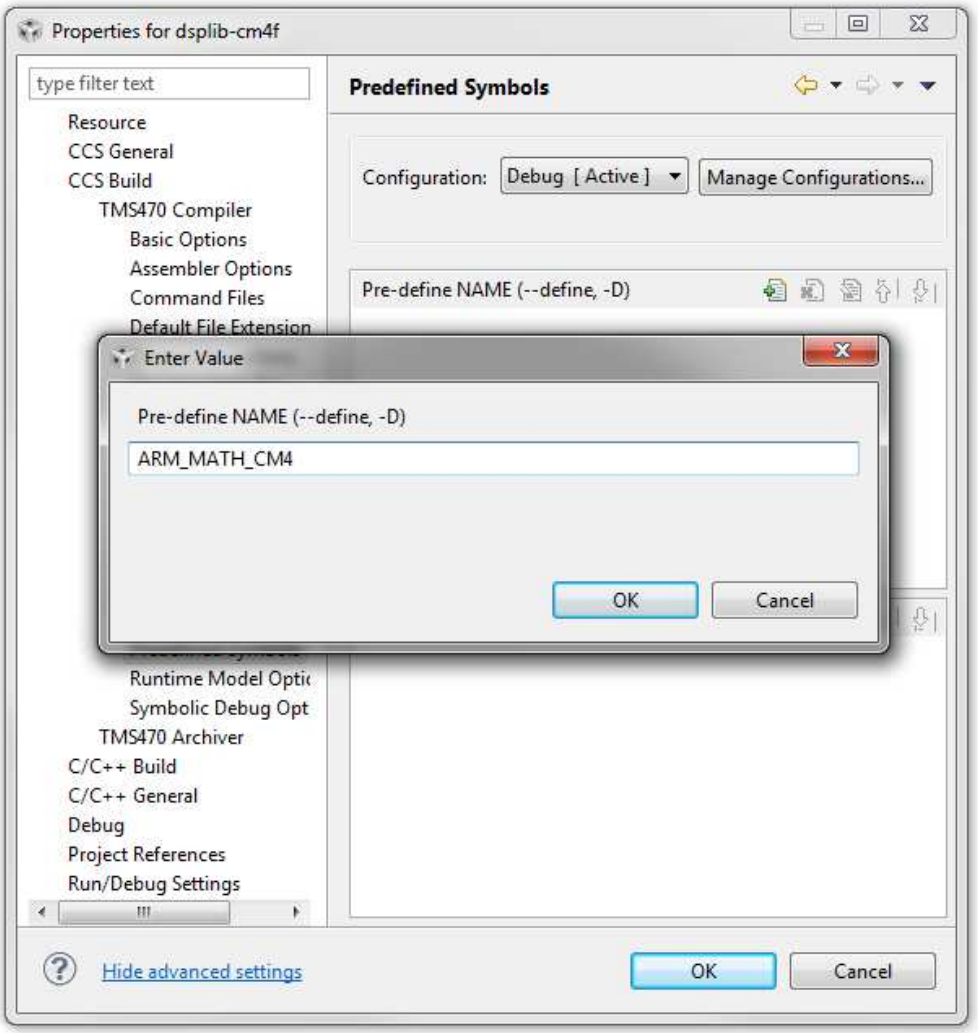
   如果您在尝试使用一个相对路径来连接到示例代码的公共目录和 DSP 库的 Include 目录时遇到问题，您可以添加一个到这些位置的硬连接。 当被提示输入文件查找路径时，请敲入目录的准确位置（例如，C:\CMSIS-2_10\CMSIS\Include）而不是已连接的位置。



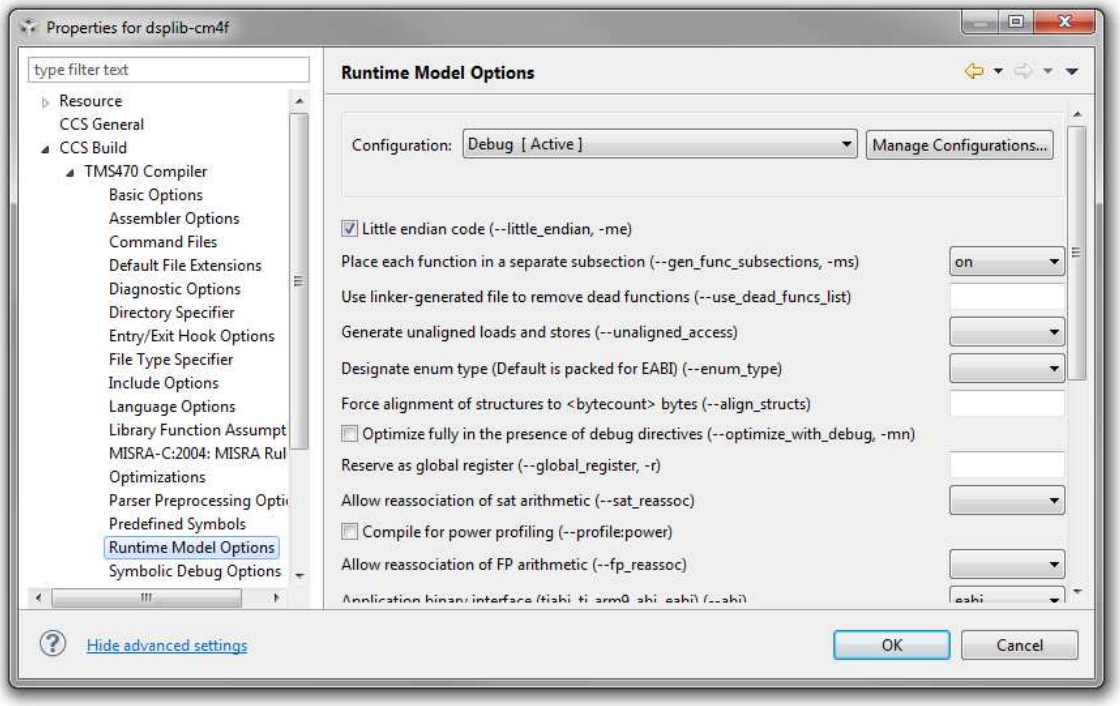
8. 单击*TMS470 Compiler*条目下的*Language Options*条目。 单击*Enable support for GCC extensions (--gcc)*（*启用对 GCC 扩展的支持*）复选框来选择。



9. 在预先定义的符号面板内，创建一个符号来告诉 DSP 库来使用基于 Cortex-M3 或 Cortex-M4 的数学函数。 单击**Add...**按钮，此按钮位于*Pre-define NAME (--define，-D)*区域内。 在*Enter Value*对话框内，在*Pre-define NAME (--define，-D)*字段内敲入 ARM_MATH_CM4 并单击 **OK**。

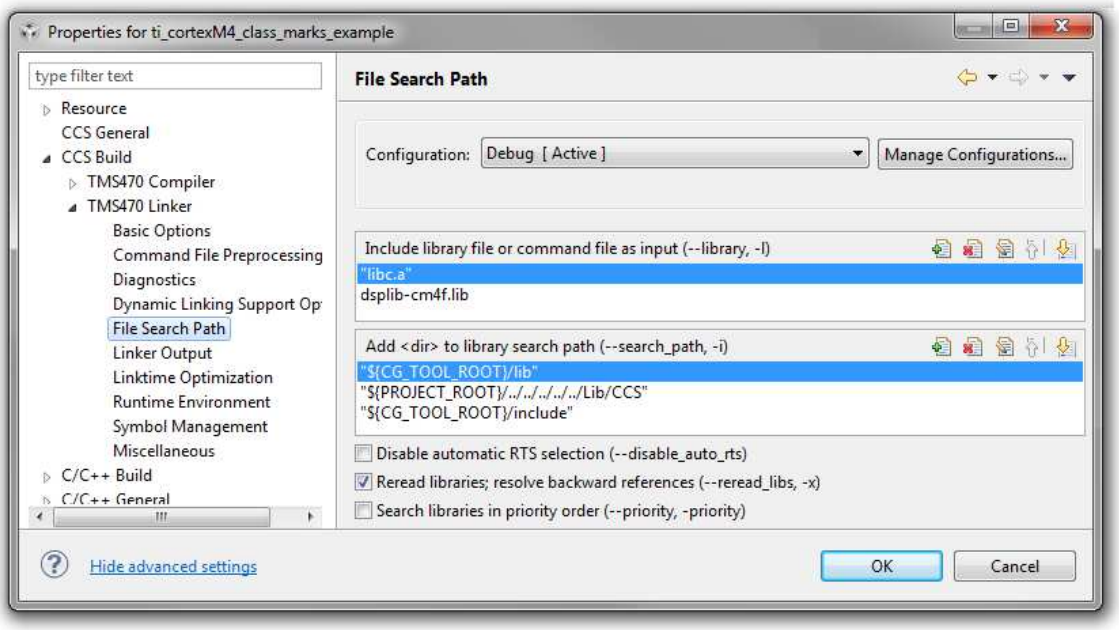


10. 在 Runtime Model Options 面板内，从*Place each function in a separate subsection (--gen_func_subsections，-ms)*下拉菜单中选择***on***选项。



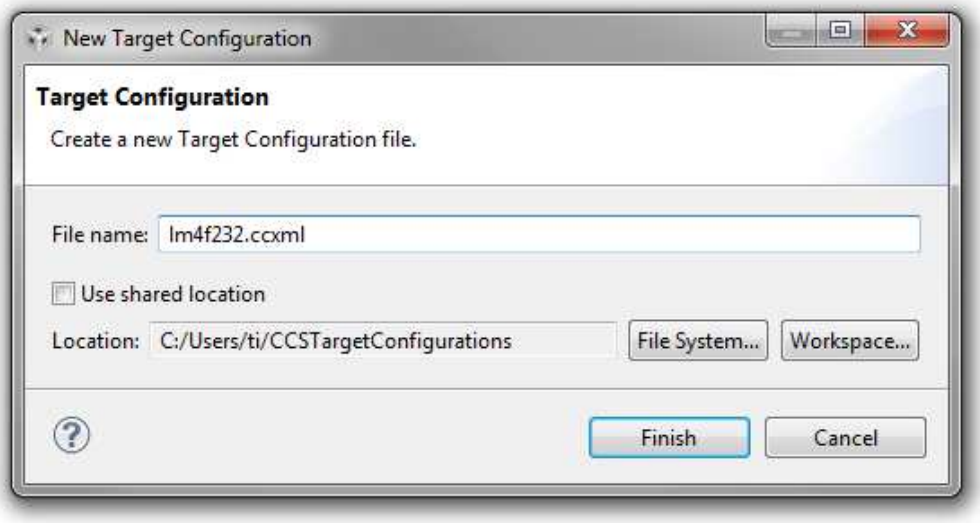
11. 单击*TMS470 Linker*条目并打开*File Search Path*（*文件查找路径*）条目。
12. 在将被使用的*Include library file or command file as input (--library，-l)*（*包含库文件或命令文件作为输入*）中创建一个针对 DSP 的条目。 对于这个示例，在部分三中创建的库文件将被使用，所以单击创建新条目图标并为此文件敲入 dsplib-cm4f.lib。
13. 在包含 DSP .lib 位置的*Add <dir> to library search path (--search_path，-I)*（*将目录添加到库查找路径*）部分中创建一个条目。 对于这个示例，得出自部分 3 的已编译二进制数被放置在新创建的文件夹内，此文件夹所处的目录与 CMSIS 提供的预编译二进制数位于同一目录内，CMSIS-<*版本*>/CMSIS/Lib，所以您必须添加一个到那个目录的引用。 要添加一个引用，单击创建新条目图标并敲入**$(PROJECT_ROOT)/../../../../../Lib/CCS**。

    如果您在尝试使用一个相对路径来连接 DSP 库位置时遇到问题，您可以添加一个到那个位置的硬连接。当被提示输入库查找路径时，请敲入目录的准确位置（例如，C:\CMSIS-2_10\CMSIS\Lib\CCS）而不是已连接的位置。



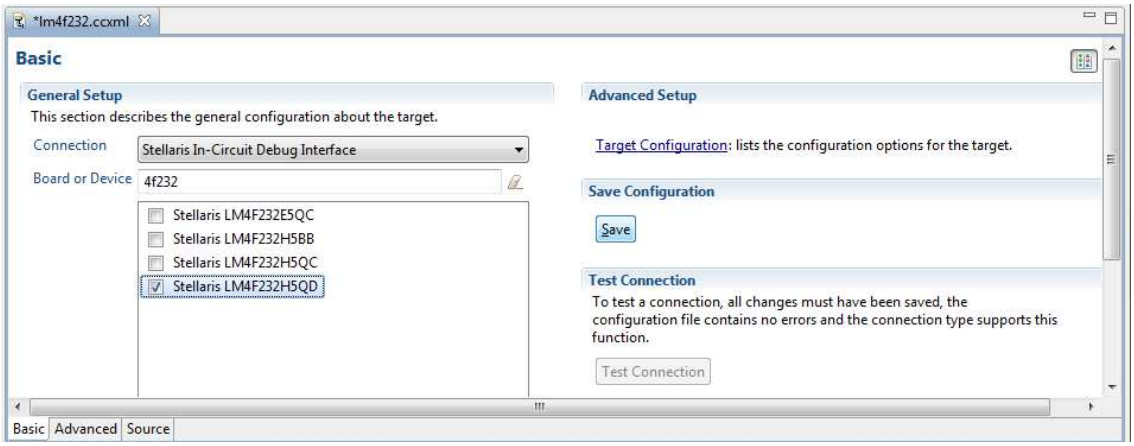
4.4 建立、运行和验证项目

1. 右键单击 Project Explorer 内的项目并选择 Build Project。
2. 通过单击 Debug 图标来使用 Code Composer Studio 调试程序来运行项目。 如果这是在工作区中首次打开调试会话，会出现一个弹出窗口以确定需要一个目标配置文件来启动一个调试会话。 单击**Yes**来打开 New Target Configuration（新目标配置）窗口。
3. 在 New Target Configuration 窗口中，在文件名称文本字段中敲入您将运行代码的主板的名称（在这个情况下，为 lm4f232）。
4. 单击**Finish**，这将打开目标配置编辑器。



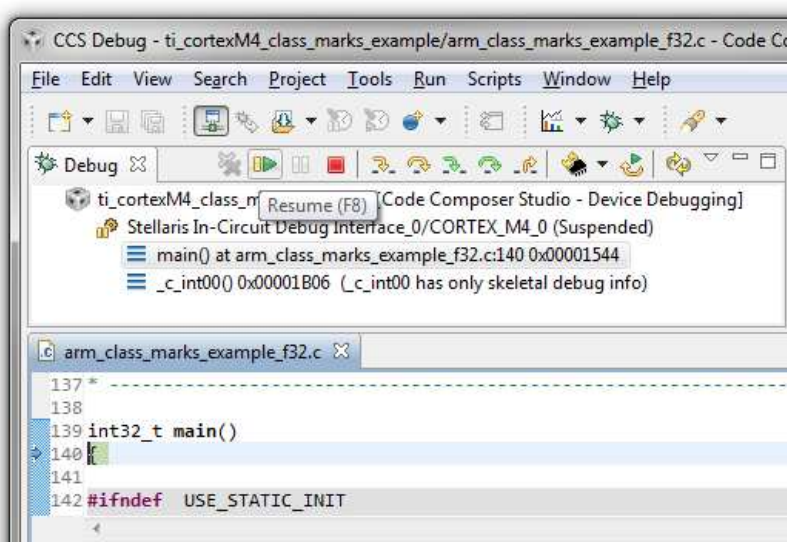
5. 在目标配置编辑器中，单击 Connection（连接）下拉菜单并选择与微控制器进行通信的方法。 这个示例使用一个 Stellaris LM4F232 评估套件，所以*Stellaris In-Circuit Debug Interface（Stellaris 电路内调试接口）*被选中。
6. 从*Board or Device（主板或器件）*列表中选择将被编辑的器件。

7. 在 Save Configuration（保存配置）区域中单击**Save**来保存目标配置。



8. 单击**Debug**按钮来启动一个调试会话，此调试会话使用刚刚被创建的 .ccxml 配置文件。 您也许需要单击 Connect Target（连接目标）按钮来启动一个到微控制器的连接。 一旦连接被建立并且闪存已被编程，微控制器运行，直到它运行至项目主函数。
9. 单击**Resume (F8)**（恢复）按钮来使程序运行。

   请注意：如果这是您第一次使用 Stellaris 电路内调试接口来连接目标，您也许需要安装适当的驱动程序来连接到目标。 您可以在*CCSv5 快速启动指南*中找到完成此操作的指令，此指南在http://processors.wiki.ti.com/index.php/Category:Code_Composer_Studio_v5内提供。



10. 几秒钟后单击 Suspend（挂起）按钮来停止微控制器并显示源代码执行在何处停止。 对于每个类别标记示例以外的函数，程序进入两个 WHILE 循环中的一个。 如果程序并未成功执行，程序被保留在 WHILE 循环内，此循环被带有一个测试条件 (status != ARM_MATH_SUCCESS) 的 IF 语句所包围。 如果程序成功执行，程序被留在 WHILE 循环内，此循环立即出现在之前提到的 IF 语句之后。 对于类别标记示例，没有内置的方法来验证微控制器执行状态。

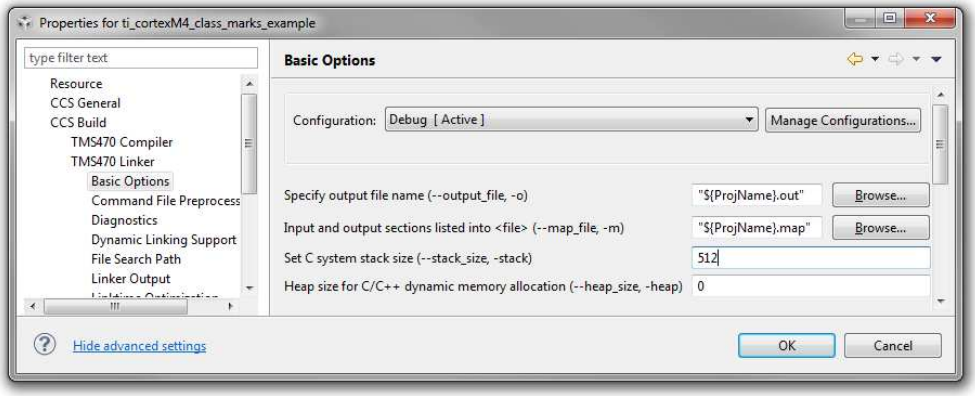
4.5 源代码修改

对于大多数 ARM 示例项目，上述的步骤可被用来建立和运行 ARM 提供的源代码。 然而，图形均衡器示例和线性内插示例要求使用 Code Composer Studio v5 和德州仪器的微控制器对源代码进行修改以正常建立和运行。

4.5.1 图形均衡器示例修改

如果图形均衡器示例与缺省 CCS 设置一起运行，由于发生的一个堆栈溢出错误，微控制器在执行到一半时进入其故障处理程序。 为了改正这个故障，请进行以下操作：

1. 打开那个项目的 Properties 菜单并将*CCS Build*条目和*TMS470 Linker*条目展开。
2. 单击*Basic Options*条目将*Set C system stack size (--stack_size，-stack)*（*设定 C 系统堆栈尺寸*）文本字段改为 512（而不是缺省值 256）。
3. 单击**OK**，现在图形均衡器示例项目现在可以正常运行。



4.5.2 线性内插示例修改

线性内插示例在使用缺省 CCS 设置时也不能正常运行。 线性内插示例包含一些显示在表格中的值，这些值代表 x 从负 π 变成 2*π 时的 sin(x) 的波形（增量为 0.00005）。 这个间隔尺寸使得得出的已编译二进制数对于大多数 Stellaris 评估套件来说太大。 ti_linear_interp_data_37968.c 数据文件代表增量为 0.00025 的同一个阵列并与本应用报告一同提供。 使用这个替代的数据文件将使已编译的二进制数足够小以适合于具有 256KB 闪存存储器的部件。 这也使得对线性内插示例代码的改变成为必然（这是因为静态分配阵列的尺寸已被改变），所以当与本示例的源代码相连接时要使用 ti_linear_interp_example_f32.c 数据文件。

线性内插示例还包含一个有可能使示例在执行时出现故障的错误。 此示例的用途是为了显示使用 CMSIS DSP 库的线性内插 sin 函数可实现的准确度的不同，此 sin 函数使用三次内插和线性内插来得到其返回值，而库的标准 sin 函数只使用三次内插。 相对于预先计算得出已知为正确的信号，用来比较这两个函数准确度的方法计算两个信号的信噪比。 遗憾的是，使用线性内插的方法给出了一个与预先计算得出信号完全匹配的结果，这使得 SNR 函数尝试去获取一个除以 0 的日志记录值。这样，函数的自检方法不被认为是可信的。此外，您应该使用调试程序来验证当使用线性内插函数时，代表 sin 值的 10 项长数组比使用标准函数时更加准确。

5 结论

您可以使用本文档中提供的信息以及 ARM 的 CMSIS 网站内提供的资源来轻松且快速地执行不同的复杂 DSP 算法。 虽然这些函数可被独立编码，但是所获得的结果有可能导致更长的开发时间以及生成效率较低的代码。 当使用德州仪器的 Stellaris 微控制器用于要求复杂 DSP 功能的应用时，请按照本应用报告中的指令进行操作已确保准确、可靠且高效的代码。

6 参考

以下相关文档和软件可从www.ti.com/stellaris上的 Stellaris 网站内下载：

- Stellaris LM4F232H5QD 微控制器数据表（文献编号SPMS319）

额外参考资料包括：

- TI 处理器维基网站：http://processors.wiki.ti.com/index.php/Category:Code_Composer_Studio_v5内的 *德州仪器 Code Composer Studio v5 入门指南*
- 针对 CMSIS DSP 库和示例代码的源代码可从 ARM CMSIS 网站：www.onarm.com内下载

重要声明

德州仪器(TI) 及其下属子公司有权根据 JESD46 最新标准, 对所提供的产品和服务进行更正、修改、增强、改进或其它更改， 并有权根据 JESD48 最新标准中止提供任何产品和服务。客户在下订单前应获取最新的相关信息, 并验证这些信息是否完整且是最新的。所有产品的销售都遵循在订单确认时所提供的TI 销售条款与条件。

TI 保证其所销售的组件的性能符合产品销售时 TI 半导体产品销售条件与条款的适用规范。仅在 TI 保证的范围内，且 TI 认为 有必要时才会使用测试或其它质量控制技术。除非适用法律做出了硬性规定，否则没有必要对每种组件的所有参数进行测试。

TI 对应用帮助或客户产品设计不承担任何义务。客户应对其使用 TI 组件的产品和应用自行负责。为尽量减小与客户产品和应 用相关的风险，客户应提供充分的设计与操作安全措施。

TI 不对任何 TI 专利权、版权、屏蔽作品权或其它与使用了 TI 组件或服务的组合设备、机器或流程相关的 TI 知识产权中授予 的直接或隐含权限作出任何保证或解释。TI 所发布的与第三方产品或服务有关的信息，不能构成从 TI 获得使用这些产品或服 务的许可、授权、或认可。使用此类信息可能需要获得第三方的专利权或其它知识产权方面的许可，或是 TI 的专利权或其它 知识产权方面的许可。

对于 TI 的产品手册或数据表中 TI 信息的重要部分，仅在没有对内容进行任何篡改且带有相关授权、条件、限制和声明的情况 下才允许进行复制。TI 对此类篡改过的文件不承担任何责任或义务。复制第三方的信息可能需要服从额外的限制条件。

在转售 TI 组件或服务时，如果对该组件或服务参数的陈述与 TI 标明的参数相比存在差异或虚假成分，则会失去相关 TI 组件 或服务的所有明示或暗示授权，且这是不正当的、欺诈性商业行为。TI 对任何此类虚假陈述均不承担任何责任或义务。

客户认可并同意，尽管任何应用相关信息或支持仍可能由 TI 提供，但他们将独力负责满足与其产品及在其应用中使用 TI 产品 相关的所有法律、法规和安全相关要求。客户声明并同意，他们具备制定与实施安全措施所需的全部专业技术和知识，可预见 故障的危险后果、监测故障及其后果、降低有可能造成人身伤害的故障的发生机率并采取适当的补救措施。客户将全额赔偿因 在此类安全关键应用中使用任何 TI 组件而对 TI 及其代理造成的任何损失。

在某些场合中，为了推进安全相关应用有可能对 TI 组件进行特别的促销。TI 的目标是利用此类组件帮助客户设计和创立其特 有的可满足适用的功能安全性标准和要求的终端产品解决方案。尽管如此，此类组件仍然服从这些条款。

TI 组件未获得用于 FDA Class III（或类似的生命攸关医疗设备）的授权许可，除非各方授权官员已经达成了专门管控此类使 用的特别协议。

只有那些 TI 特别注明属于军用等级或“增强型塑料”的 TI 组件才是设计或专门用于军事/航空应用或环境的。购买者认可并同 意，对并非指定面向军事或航空航天用途的 TI 组件进行军事或航空航天方面的应用，其风险由客户单独承担，并且由客户独 力负责满足与此类使用相关的所有法律和法规要求。

TI 已明确指定符合 ISO/TS16949 要求的产品，这些产品主要用于汽车。在任何情况下，因使用非指定产品而无法达到 ISO/TS16949 要求，TI不承担任何责任。

	产品		应用
数字音频	www.ti.com.cn/audio	通信与电信	www.ti.com.cn/telecom
放大器和线性器件	www.ti.com.cn/amplifiers	计算机及周边	www.ti.com.cn/computer
数据转换器	www.ti.com.cn/dataconverters	消费电子	www.ti.com/consumer-apps
DLP® 产品	www.dlp.com	能源	www.ti.com/energy
DSP - 数字信号处理器	www.ti.com.cn/dsp	工业应用	www.ti.com.cn/industrial
时钟和计时器	www.ti.com.cn/clockandtimers	医疗电子	www.ti.com.cn/medical
接口	www.ti.com.cn/interface	安防应用	www.ti.com.cn/security
逻辑	www.ti.com.cn/logic	汽车电子	www.ti.com.cn/automotive
电源管理	www.ti.com.cn/power	视频和影像	www.ti.com.cn/video
微控制器 (MCU)	www.ti.com.cn/microcontrollers		
RFID 系统	www.ti.com.cn/rfidsys		
OMAP应用处理器	www.ti.com/omap		
无线连通性	www.ti.com.cn/wirelessconnectivity	德州仪器在线技术支持社区	www.deyisupport.com

邮寄地址： 上海市浦东新区世纪大道 1568 号，中建大厦 32 楼 邮政编码： 200122