

《DSP 应用技术》 实验指导书

上海交通大学
电气工程实验中心

目录

1.实验课程背景任务及主要内容.....	3
2.CCS 使用教程.....	3
2.1 CCS 概述.....	3
2.1.1 调试程序.....	4
2.1.2 分析.....	4
2.1.3 脚本.....	4
2.1.4 图像分析和虚拟化.....	4
2.1.5 编译器.....	5
2.1.6 模拟.....	5
2.1.7 硬件调试（仿真）.....	5
2.1.8 实时操作系统支持.....	6
2.2 CCS 的基本使用方法.....	7
2.2.1 导入 CCS3.x 版本的项目.....	7
2.2.2 导入 CCE 项目或以前版本的 CCSv4 项目.....	10
2.2.3 创建项目.....	12
2.2.4 编译项目.....	18
2.2.5 配置编译设置.....	18
2.2.6 项目调试.....	18
3. OMAPL138 EVM 使用说明.....	31
3.1 简介.....	31
3.2 开发板使用步骤.....	34
4. 实验具体要求.....	39
4.1 实验一.....	39
4.2 实验二.....	41
4.3 实验三.....	42

1.实验课程背景任务及主要内容

实验背景: 以 OMAP L138 EVM 开发板为基础, 以 CCS 为主要的开发环境, 熟悉 DSP 的设计理念及开发过程。

实验任务:

- 1) 了解 DSP 的原理
- 2) 熟悉 CCS 开发环境
- 3) 学会用 C 语言写 DSP 代码
- 4) 结合开发板做简单的实验
- 5) 学会 CCS 环境下的 C 语言调试

实验内容:

- 1) 拨码开关输入与外部存储器读取实验
- 2) 音频发声实验
- 3) FFT 的实现

2.CCS 使用教程

2.1 CCS 概述

Code Composer Studio&™ (CCS 或 CCStudio) 是一种针对 TI 的 DSP、微控制器和应用处理器的集成开发环境。CCStudio 包括一套用于开发和调试嵌入式应用程序的工具。它包括用于各种 TI 设备系列的编译器、源代码编辑器、项目生成环境、调试程序、探查器、模拟器和其他许多功能。CCStudio 提供一个单一用户界面, 指导用户完成应用程序开发流程的每一步骤。类似的工具和界面使用户能够比以前更快地开始使用, 并且能够向他们的应用程序添加功能, 这些都归功于成熟的生产能力工具。

CCStudio 版本 4 (CCSv4) 以 Eclipse 开源软件框架为基础。CCSv4 之所以以 Eclipse 为基础, 是因为 Eclipse 为开发环境提供了一个优异的软件框架, 是众多嵌入式软件供应商所使用的标准框架。CCSv4 将 Eclipse 软件框架的优势和来自

TI 的高级嵌入式调试功能相结合，为嵌入式程序开发人员生成一个功能丰富的吸引人的开发环境。

2.1.1 调试程序

CCStudio 的集成调试程序具有用于简化开发的众多功能和高级断点。条件断点或硬件断点以全 C 表达式、本地变量或寄存器为基础。高级内存窗口允许您检查内存的每一级别，以便您可以调试复杂的缓存一致性问题。CCStudio 支持复杂的多处理器或多核系统的开发。全局断点和同步操作提供了对多个处理器和多核的控制。

2.1.2 分析

CCStudio 的交互式探查器使快速测量代码性能并确保在调试和开发过程中目标资源的高效使用变得更容易。探查器使开发人员能够轻松分析其应用程序中指令周期内或其他事件内的所有 C/C++ 函数，例如缓存未命中/命中率、管道隔栏和分支。分析范围可用于在优化期间将精力集中在代码的高使用率方面，帮助开发人员开发出经过优化的代码。分析可用于任何组合的汇编、C++ 或 C 代码范围。为了提高生产能力，所有分析设备在整个开发周期中都可使用。

2.1.3 脚本

某些任务，例如测试，需要运行数小时或数天而不需要用户交互。要完成此类任务，IDE 应能自动执行一些常见任务。CCStudio 拥有完整的脚本环境，允许自动进行重复性任务，例如测试和性能基准测试。一个单独的脚本控制台允许您在 IDE 内键入命令或执行脚本。

2.1.4 图像分析和虚拟化

CCStudio 拥有许多图像分析及图形虚拟化功能。其中包括以图形方式在能够自动刷新的屏幕上查看变量和数据的能力。CCStudio 还能以本机格式（YUV、RGB）查看主机 PC 或在目标电路板中加载的图像和视频数据。

2.1.5 编译器

TI 已经开发了专门为了最大程度地提高处理器的使用率和性能而优化的 C/C++ 编译器。TI 编译器使用各种各样经典的、面向应用的、成熟的、因设备而异的优化，专为所有支持的结构而优化。其中部分优化包括：

- 消除公共子表达式
- 软件流水
- 强度折减
- 自动增量寻址
- 基于成本的寄存器分配
- 指令预测
- 硬件循环
- 函数内联
- 矢量化

TI 编译器还执行程序级别优化，在应用程序级别评估代码性能。通过程序级别视图，编译器能够像具有完整系统视图的汇编程序开发人员一样生成代码。编译器充分利用此应用程序级别视图，找出能够显著提升处理器性能的折衷。

TI ARM 和 Microcontroller C/C++ 编译器经过专门针对代码大小和控制代码效率的优化。它们具备行业领先的性能和兼容性。

2.1.6 模拟

模拟器向用户提供一种在能够使用开发板之前开始开发的方式。模拟器还具有更加透彻地了解应用程序性能和行为的优势。提供了几种模拟器，让用户能够权衡周期精确性、速度和外围设备模拟，一些模拟器特别适合算法基准测试，而另一些特别适合更加详细的系统模拟。

2.1.7 硬件调试（仿真）

TI 设备包含高级硬件调试功能。这些功能包括：

- IEEE 1149.1 (JTAG) 和边界扫描
- 对寄存器和内存的非侵入式访问
- 实时模式，用于调试与不得禁用的中断进行交互的代码。实时模式允许您在中断事件挂起后台代码，同时继续执行时间关键中断服务例程。
- 多核操作，例如同步运行、步进和终止。其中包括跨核触发，该功能可以让一个核触发另一个核终止。

高级事件触发 (AET)，可在选定设备上使用，允许用户依据复杂事件或序列，例如无效数据或程序内存访问，终止 CPU 或触发其他事件。它能够以非侵入式方式测量性能及统计系统事件数量（例如缓存事件）。

CCStudio 提供有关选定设备的处理器跟踪，帮助客户发现以前“看不到的”复杂实时缺陷。跟踪能够探测很难发现的缺陷-事件之间的争用情况、间歇式实时干扰、堆栈溢出崩溃、失控代码和不停用处理器的误中断。跟踪是一种完全非侵入式调试方法，依赖处理器内的调试单元，因此不会干扰或更改应用程序的实时行为。跟踪可以微调复杂开关密集型多通道应用程序的代码性能和缓存优化。处理器跟踪支持程序、数据、计时和所选处理器与系统事件/中断的导出。可以将处理器跟踪导出到 XDS560 跟踪外部 JTAG 仿真器或选定设备上，或导出到芯片缓存嵌入式跟踪缓存(ETB)上。

2.1.8 实时操作系统支持

CCSv4 具有两个版本的 TI 实时操作系统：

- DSP/BIOS5.4x 是一种为 DSP 设备提供预清空多任务服务的实时操作系统。其服务包括 ISR 调度、软件中断、信号灯、消息、设备 I/O、内存管理和电源管理。此外，DSP/BIOS5.x 还包括调试诊断和加工，包括低系统开销打印和统计数据收集。
- BIOS6.x 是一种高级可扩展实时操作系统，支持 ARM926、ARM Cortex M3、C674x、C64x+、C672x 和基于 28x 的设备。它提供 DSP/BIOS 5.x 没有的若干内核和调试增强，包括更快、更灵活的内存管理、事件和优先级继承互斥体。

注意：BIOS6.x 包括 DSP/BIOS5.x 兼容层，从而使应用程序源代码的迁移非常轻松。

2.2 CCS 的基本使用方法

2.2.1 导入 CCS3.x 版本的项目

以前版本的 Code Composer Studio 使用的项目文件(*.pjt) 包含了所有生成选项以及对源文件的引用，而 CCSv4 使用了新的项目格式，它自动在项目目录下生成每个有效源文件，并将生成选项存储在几个以圆点开头的文件和目录中。

由于这些差异，CCSv4 提供了一个“Import Legacy CCS Project Wizard（导入旧版 CCS 项目向导）”来帮助完成迁移。

操作步骤如下：

① 选择“Project->Import Legacy CCSv3.3Project（项目->导入旧版 CCSv3.3 项目）”，启动向导。

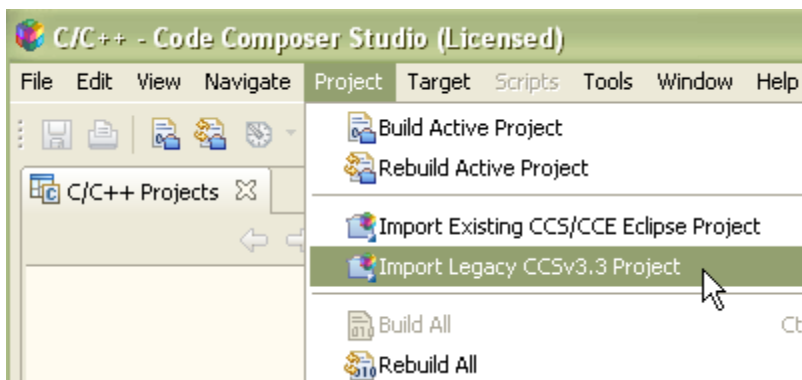


图 2.2.1 导入旧版 CCS 项目向导菜单

②指定要导入的 CCSv3*.pjt 文件。单击“Select a Project File:（选择项目文件:）”，再单击“Browse...（浏览...）”，选择要转换的.pjt 文件。

- 选中“Copy projects into workspace（将项目复制到工作区）”复选框可将项目及关联文件复制到 CCSv4 工作区中。这样做可以依原样保持原始项目。
- 有些示例项目（安装在 C:\tidcs、Stellarisware 和 NDK 中的 C2000 示例）与原始项目位置相对的目录有一定的相关性。在这种情况下，建议选择

“Keep original location for each project（保持每个项目的原始位置）”以保持相对路径不变。

可选：一次可以转换多个项目。选择“Select Search-directory:（选择搜索目录:）”，再单击“Browse...（浏览...）”选择文件夹，在其中递归查找要导入的 CCSv3 项目。任何符合条件、能够导入的项目都将显示在“Discovered legacy projects（已找到的旧版项目）”列表中。

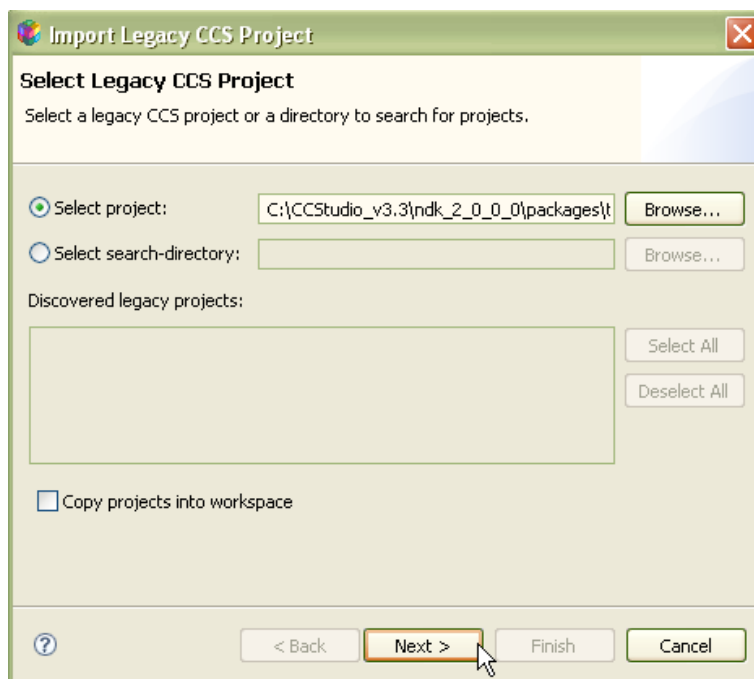


图 2.2.2 导入单个项目

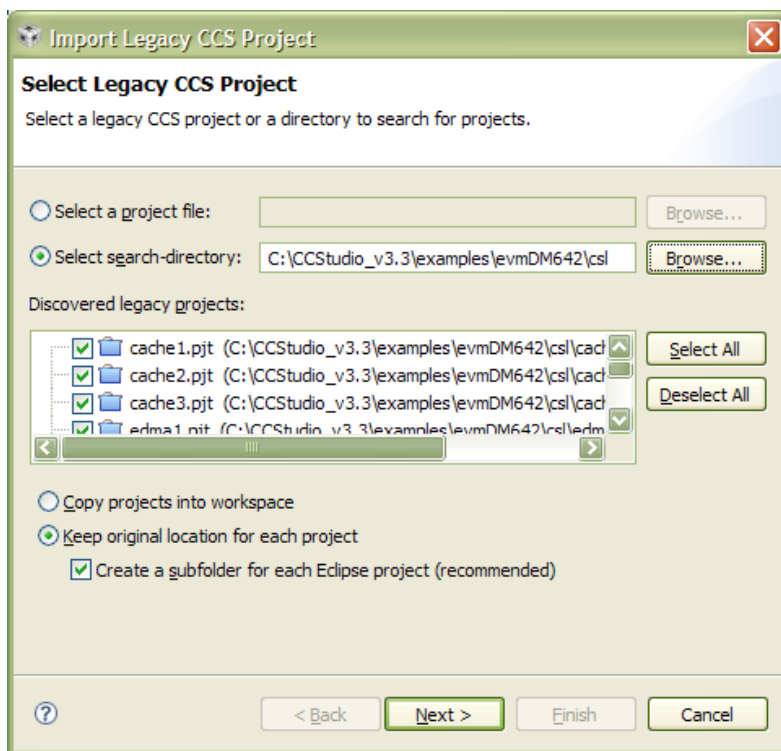


图 2.2.3 导入多个项目

③在下一个屏幕中可以选择要使用的代码生成工具版本。多数情况下可以保留 CCSv4 提供的默认版本。单击“Next（下一步）”。

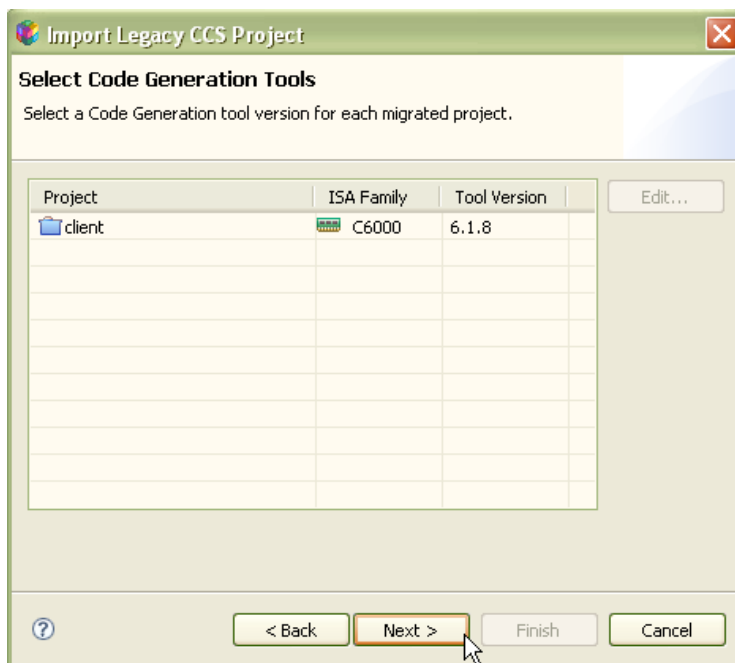


图 2.2.4 选择代码生成工具版本

④指定要使用的 DSP/BIOS 版本。多数情况下可以保留 CCSv4 提供的默认版本。单击“Finish（完成）”。

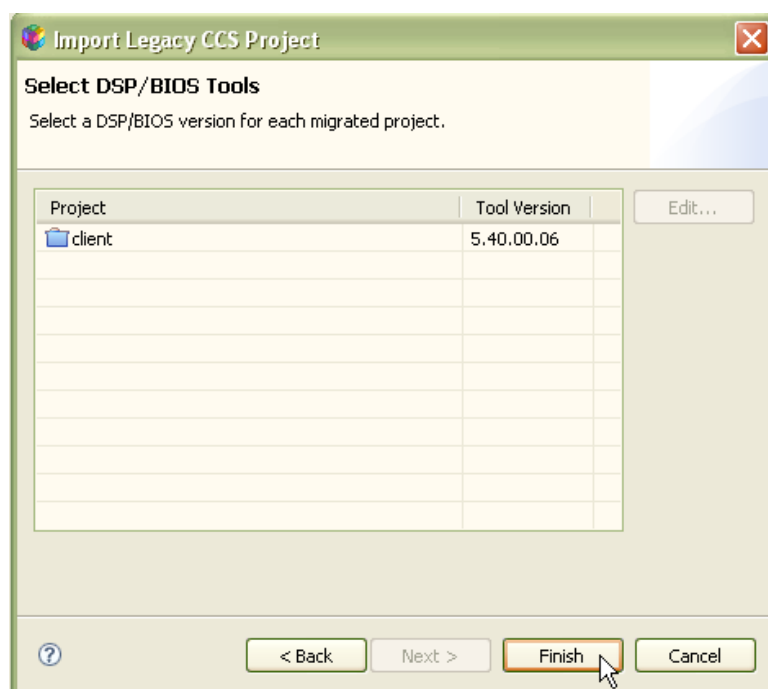


图 2.2.5 选择 DSP/BIOS 版本

⑤向导完成转换过程后，新生成的 CCSv4 项目就会出现在工作区内。

2.2.2 导入 CCE 项目或以前版本的 CCSv4 项目

尽管 CCE 和所有版本的 CCSv4 都有着相同的项目格式，Eclipse 仍然要求导入这些项目，以保持与当前安装版本一致的相关性，例如包含目录、工具版本等。

操作步骤如下：

①此过程和第一部分介绍的项目导入过程非常类似。转到菜单“File -> Import Existing CCS/CCE Eclipse Project（文件->导入现有的 CCS/CCE Eclipse 项目）”。

②有两种导入项目的方法。

- 要导入某个现有目录下的一个或多个项目，请选中“Select root directory（选择根目录）”选项。单击“Browse（浏览）”选择包含项目目录的目录。任何有效的的项目都会显示在“Projects:（项目:）”框中：

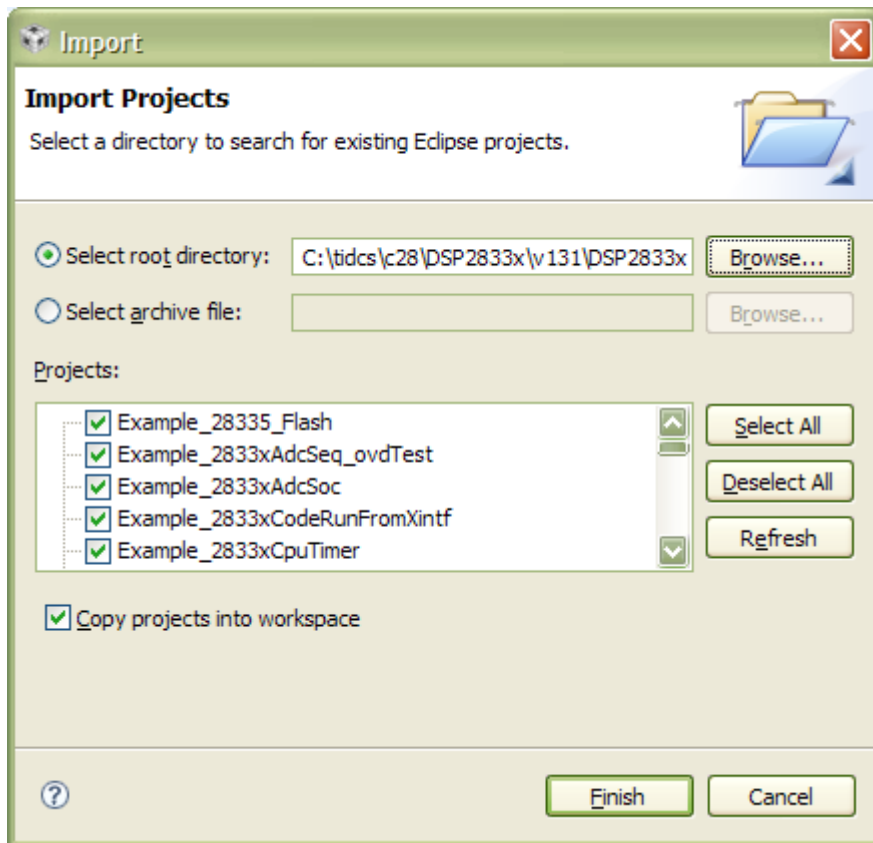


图 2.2.6 选择项目目录

- 要导入某个 zip 文件中的一个或多个项目，请选中“Select archive file: (选择存档文件:)”选项。单击“Browse (浏览)”选择包含项目的 zip 文件。选择了文件之后，任何有效的的项目都会显示在“Projects: (项目:)”框中：

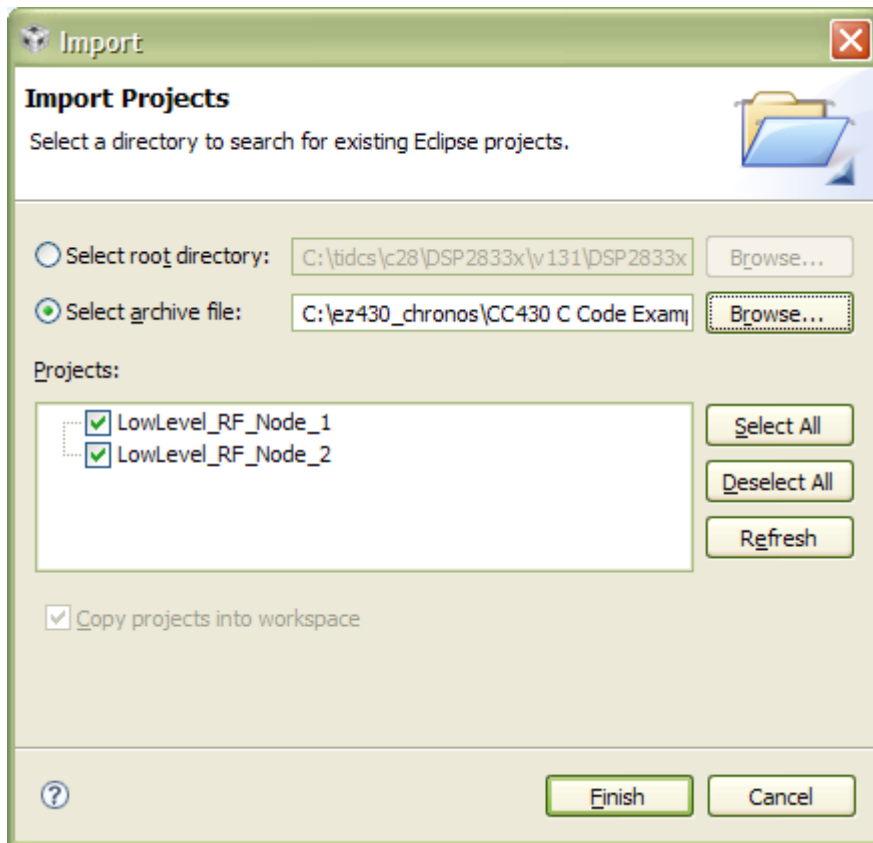


图 2.2.7 选择项目存档文件

2.2.3 创建项目

欢迎屏幕关闭之后，将会显示下面的工作区，此时可以创建新项目。

①转到菜单“File -> New -> CCS Project（文件->新建-> CCS 项目）”。

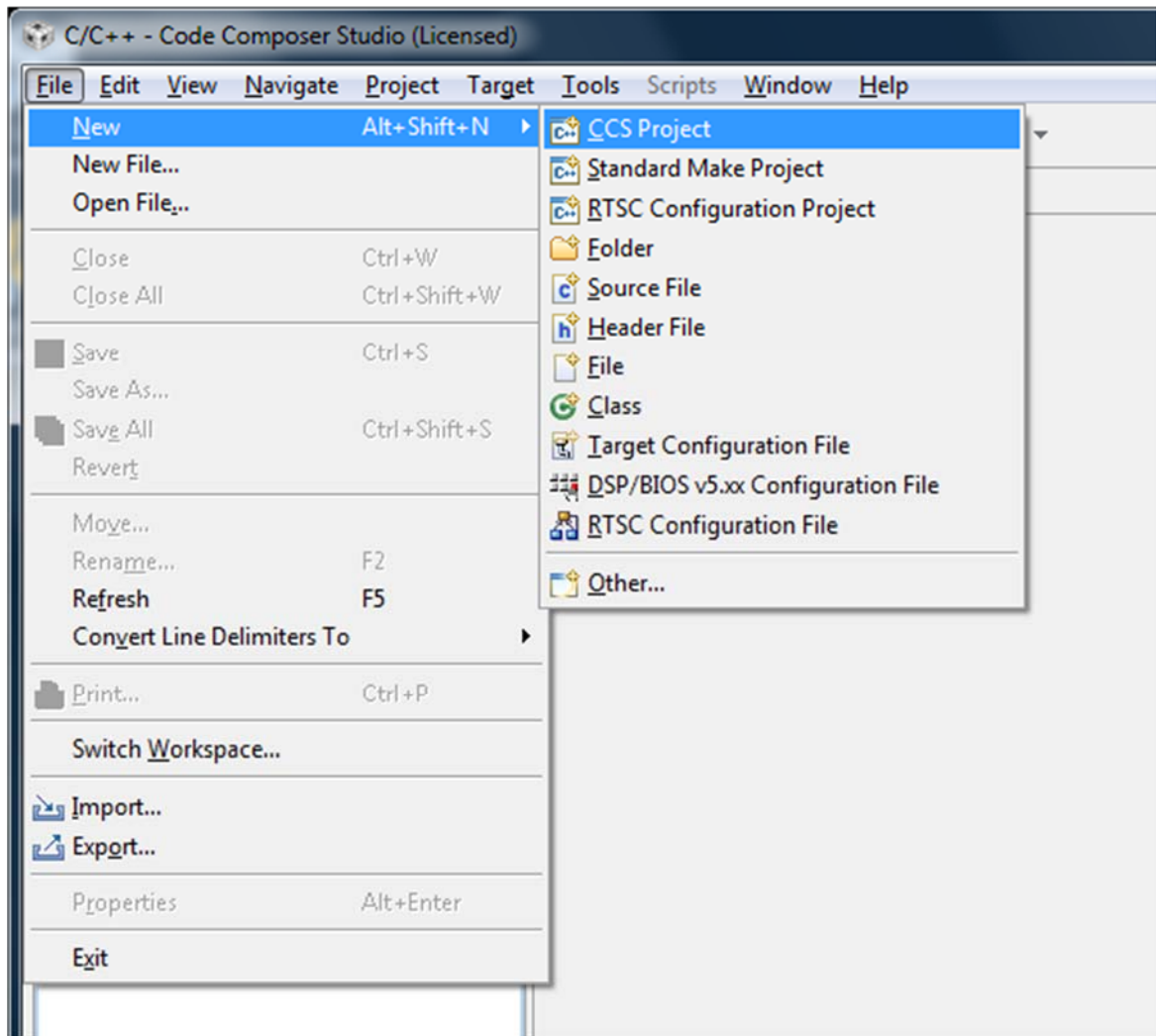


图 2.2.10 创建新项目

②在“Project Name（项目名称）”字段中，键入新项目的名称。若选中“Use default location（使用默认位置）”选项（默认启用），将会在工作区文件夹中创建项目。取消选中该选项可以选择一个新位置（使用“Browse...（浏览...）”按钮）。单击“Next（下一步）”。

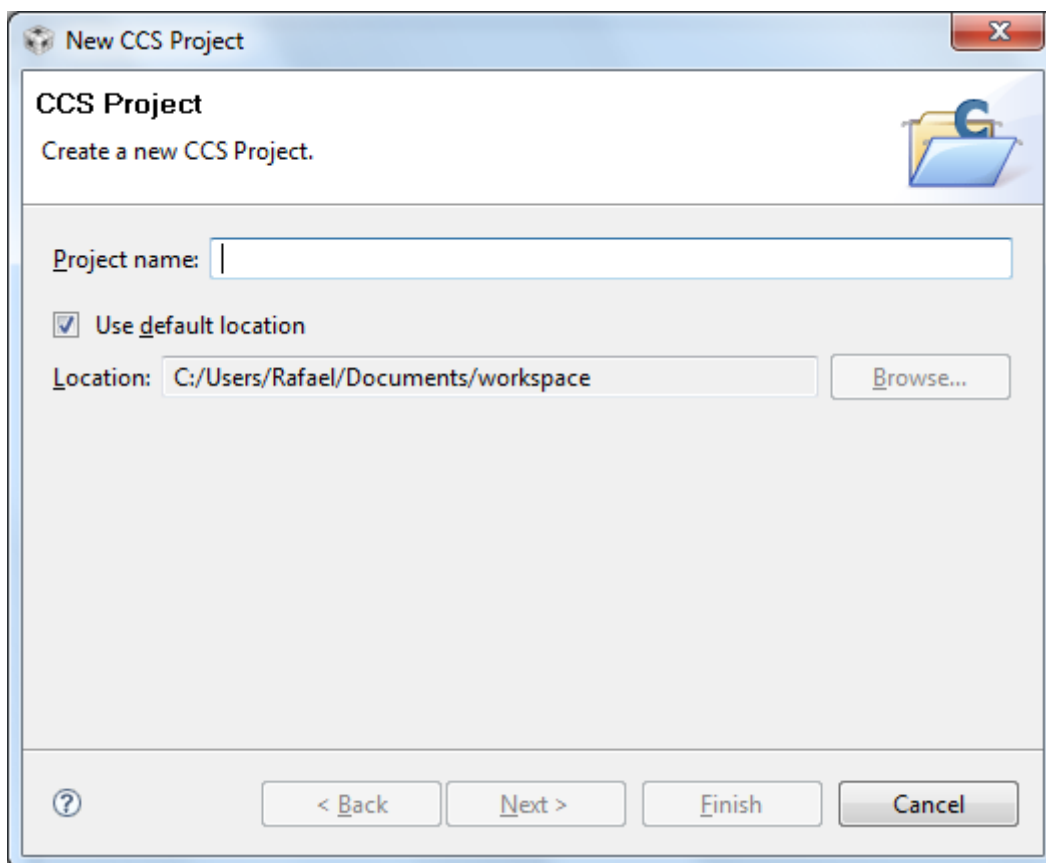


图 2.2.11 命名新项目

将项目命名，然后单击“Next（下一步）”。

③在“Project Type:（项目类型:）”下拉菜单中选择要使用的体系结构。单击“Next（下一步）”。

注意：将在步骤 5 中选择具体设备。

如果项目针对的是 Cortex 设备（Stellaris 或 Hercules），请选择“ARM”。

如果项目针对的是 SoC 设备（DaVinci、OMAP），请根据所使用的芯片核选择“ARM”或“C6000”。

可选：还可以在此屏幕中为项目选择或添加生成配置。默认情况下，“Debug（调试）”和“Release（发布）”处于启用状态。

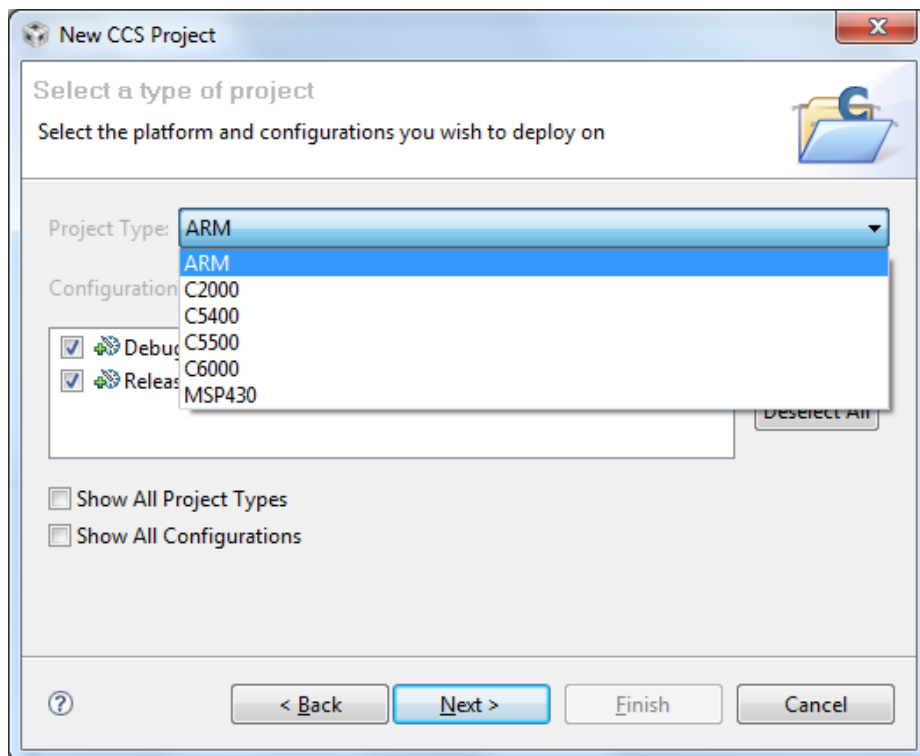


图 2.2.12 体系结构选择

选择“C6000”，然后单击“Next（下一步）”。

④通常可留空，但是如果该项目依赖于需要首先生成的其他项目（例如静态库项目），请在此处选择这些相关项目。单击“Next（下一步）”。

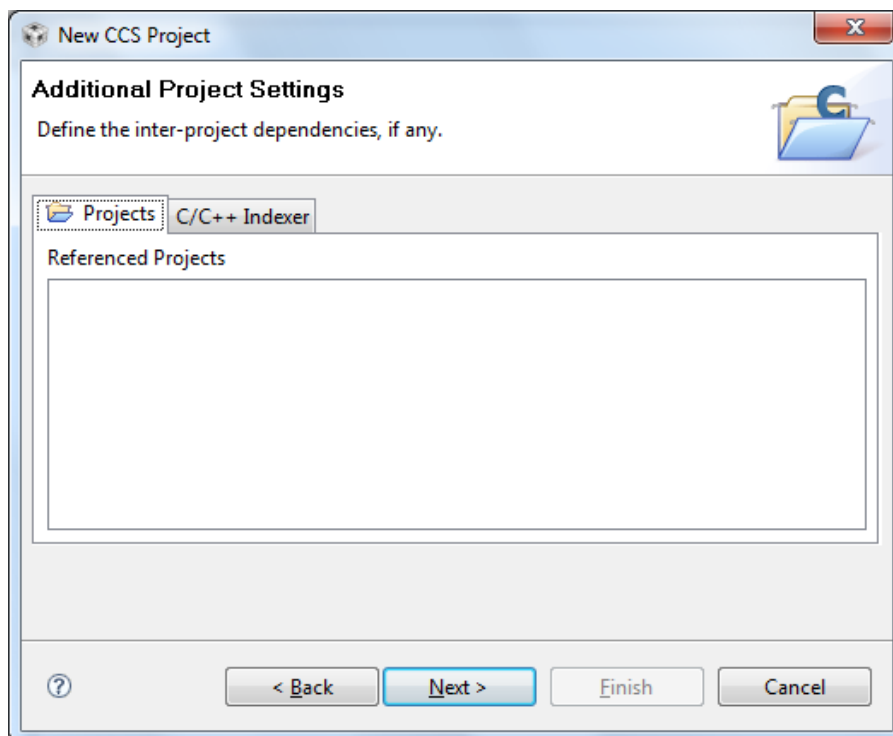


图 2.2.13 定义项目相关性

“C/C++ Indexer (C/C++ 索引器)”选项卡可配置索引器的级别。索引器是 CCSv4 的一项功能，用于创建源代码信息列表，这些信息可支持编辑器中的自动完成和“转到定义”功能。默认选项为“Full C/C++ Indexer (完整 C/C++ 索引器)”，该选项可提供最多的功能。

⑤在接下来的这一屏幕中，大部分选项都可保留为默认值。根据所做的选择，将会显示其他屏幕。

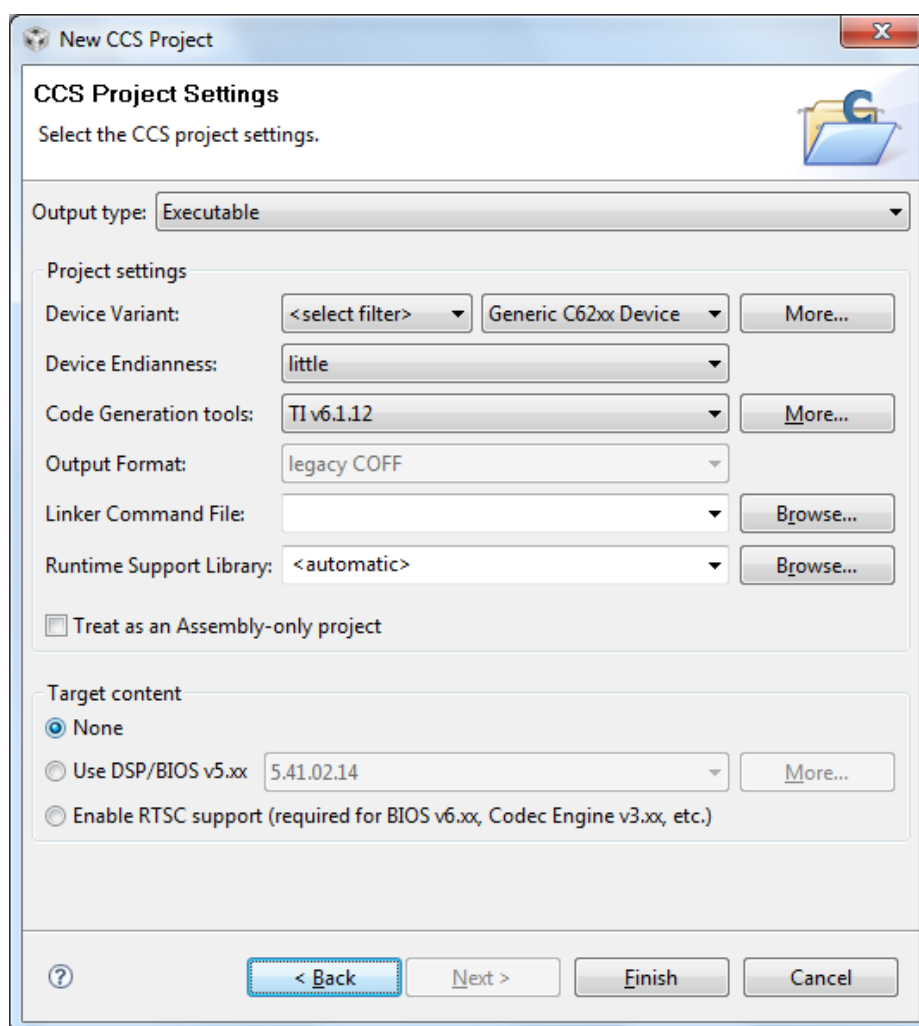


图 2.2.14 项目选项

选择“Generic C674x Device(通用 C674x 设备)”，然后单击“Finish(完成)”。

- “Output type (输出类型)”：将设置保留为“Executable (可执行)”以生成完整的程序。另一个选项为“Static Library (静态库)”，静态库是其他项目要使用的功能的集合。

- “Code Generation tools（代码生成工具）”：保留默认设置（除非安装了其他版本的代码生成工具且要使用某个特定版本）。
- “Output Format（输出格式）”：通常此选项以灰色显示为“legacy COFF（旧 COFF）”。目前只有 ARM 系列还允许选择另外一个选项“eabi (ELF)”。
- “Linker Command File（链接器命令文件）”：可留空，链接器命令文件可在稍后添加。如果存在可用的有效文件，系统将会预填充该字段。如果项目使用 BIOS，则将自动添加链接器命令文件。
- “Runtime Support Library（运行时支持库）”：通常将其保留为“<automatic>（<自动>）”，因为代码生成工具会自动选择正确的运行时库。如果需要，可在此处选择其他运行时支持库。
- “Treat as an Assembly-only project（视为仅汇编项目）”：通常将其取消选中。正如其名称所示，如果项目中没有 C 源代码文件，请选中此复选框。选中此复选框后，系统还将从项目中移除运行时支持库。

⑥单击“Finish（完成）”创建项目。所创建的项目将显示在“C/C++ Projects（C/C++ 项目）”选项卡中，可随时用于创建或添加源文件。

⑦要为项目创建文件，请在“C/C++ Projects（C/C++ 项目）”视图中右键单击项目名称，并选择“New -> Source File（新建 -> 源文件）”。在打开的文本框中，键入包含与源代码类型对应的有效扩展名（.c、.C、.cpp、.c++、.asm、.s64、.s55 等）的文件名称。单击“Finish（完成）”。

- 如果使用 MSP430，请在刚创建的空白源文件中键入字母“b”。然后同时按 <Ctrl 键>-<空格键>，添加“Blink LED”示例代码。
- 如果使用其他设备系列，请在刚创建的空白源文件中键入字母“h”。然后同时按 <Ctrl 键>-<空格键>，添加“Hello world!”示例代码。

代码模板是可使用编辑器“内容辅助”功能引用的代码的模板。也可以创建自定义代码模板。这是快速入门 Code Composer Studio IDE 的一种方式。

⑧要向项目添加现有源文件，请在“C/C++ Projects（C/C++ 项目）”选项卡中右键单击项目名称，并选择“Add Files to Project（将文件添加到项目）”，将源文件复制到项目目录。：也可以选择“Link Files to Project（将文件链接到项目）”来

创建文件引用，这样可以将文件保留在其原始目录中。如果源代码将文件包含在非常特定的目录结构中，则这是十分必要的。

添加位于以下目录的源代码 <sinewave_int.c> 和链接器命令文件 <C6748.cmd>: :C:\Program Files\Texas Instruments\ccsv4\C6000\examples

2.2.4 编译项目

在创建了项目并且添加或创建了所有文件之后，需要编译项目。

①只需转到菜单“Project -> Build Active Project (项目 -> 编译活动项目)”。
“Rebuild Active Project (重新编译活动项目)”选项可重新生成所有源文件和引用的项目。不过如果项目较大，这可能是一个漫长的过程。

注意：如果遇到编译错误，而且没有创建可执行文件，屏幕底部的控制台窗口将会显示一条错误或警告消息，并且不会启动调试会话。

2.2.5 配置编译设置

要配置生成设置，请在“C/C++ Projects(C/C++ 项目)”视图中右键单击项目，并选择“Build Properties... (生成属性...)”。有多个适用于编译器、汇编器和链接器的选项。

2.2.6 项目调试

1) 启动调试器之前

在启动调试器之前，需要选择并配置代码将要执行的目标位置。目标可以是软件模拟器或与开发板相连的仿真器。仿真器是用于直接对硬件进行调试的硬件设备，可以内置到开发板（DSK、eZdsp、EVM 等），也可以采取独立形式（XDS100v2、XDS510 USB、XDS560 等）。

以下介绍如何创建目标配置文件：

CCSV4 提供了一个十分简单易用的图形目标配置编辑器，它提供多个预配置的设备 and 开发板，而且还可以在自定义硬件中使用。

每个项目可以拥有一个或多个目标配置，但只能有一个处于活动状态。

CCSv4 还允许创建一个系统范围的目标配置，以便可以在各个项目之间进行共享。

①右键单击项目名称，并选择“New -> Target Configuration File（新建 -> 目标配置文件）”。

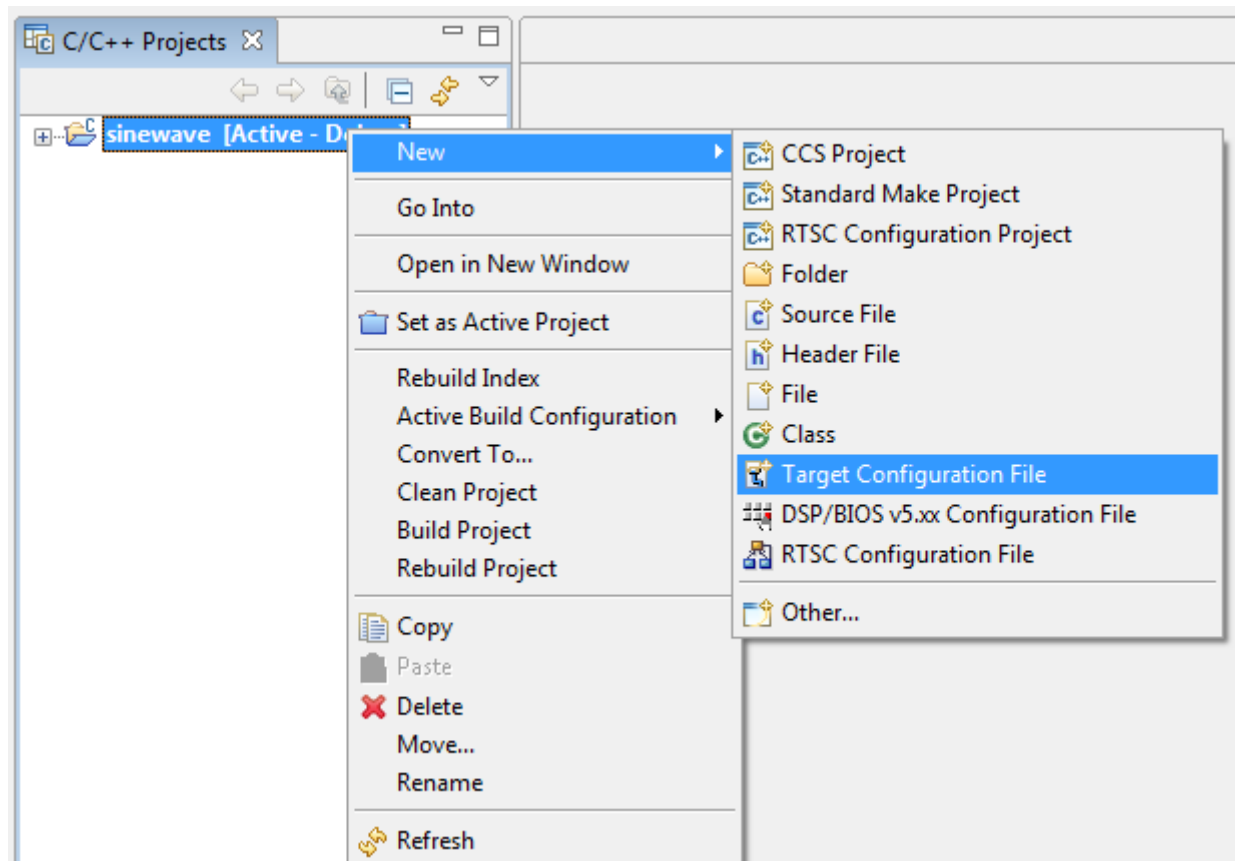


图 2.2.15 创建新目标

②为配置文件命名-将会添加扩展名.ccxml。建议根据所使用的目标和仿真器指定一个有意义的名称，例如，如果使用的是 F28335 设备和 XDS510USB 仿真器，就可以命名为 F28335_XDS510USB。

如果选中“Use shared location（使用共享位置）”选项，新的目标配置将在所有项目之间共享，并存储在默认的 CCSv4 目录下。

③单击“Finish（完成）”。此时将打开目标配置编辑器。

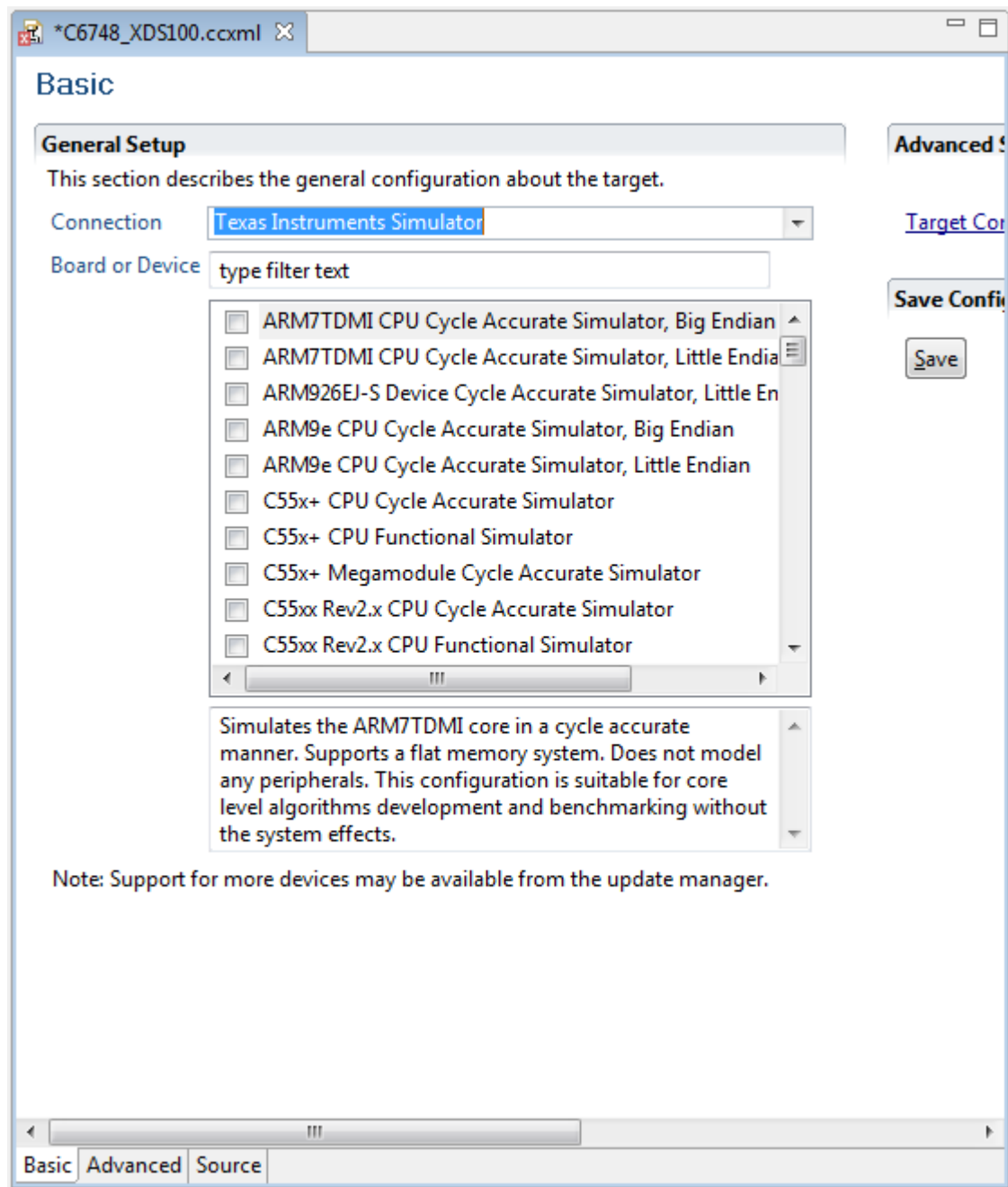


图 2.2.16 目标配置编辑器

④目标中有两项元素是必须配置的：

- 通过“Connection（连接）”下拉菜单可以选择是使用一个软件模拟器，还是使用多种内置或独立仿真器。
- “Device（设备）”部分包含与所选连接兼容的所有设备。上部的框是筛选器，可以帮助在下部框中的浏览表中选择正确设备。

⑤选择设备后，单击“Save（保存）”按钮。该配置将自动设置为“Active（活动）”。

每个项目可以拥有多个目标配置，但只能有一个处于活动状态，该配置将会自动启动。要查看系统现有的所有目标配置，只需转到菜单“View -> Target Configurations（查看 -> 目标配置）”。

2)启动调试器

创建配置之后，可通过转到菜单“Target -> Debug Active Project（目标 -> 调试活动项目）”启动调试器。将会打开“Debug Perspective（调试透视）”，专为调试定制的一组专用窗口和菜单。

注意：如果对源代码或生成选项进行了修改，启动调试器可能会导致 CCSv4 生成活动项目。

（1）加载代码

调试器完成目标初始化之后，项目的输出文件.OUT 将自动加载到活动目标，并且默认情况下代码将在 main()函数处停止。

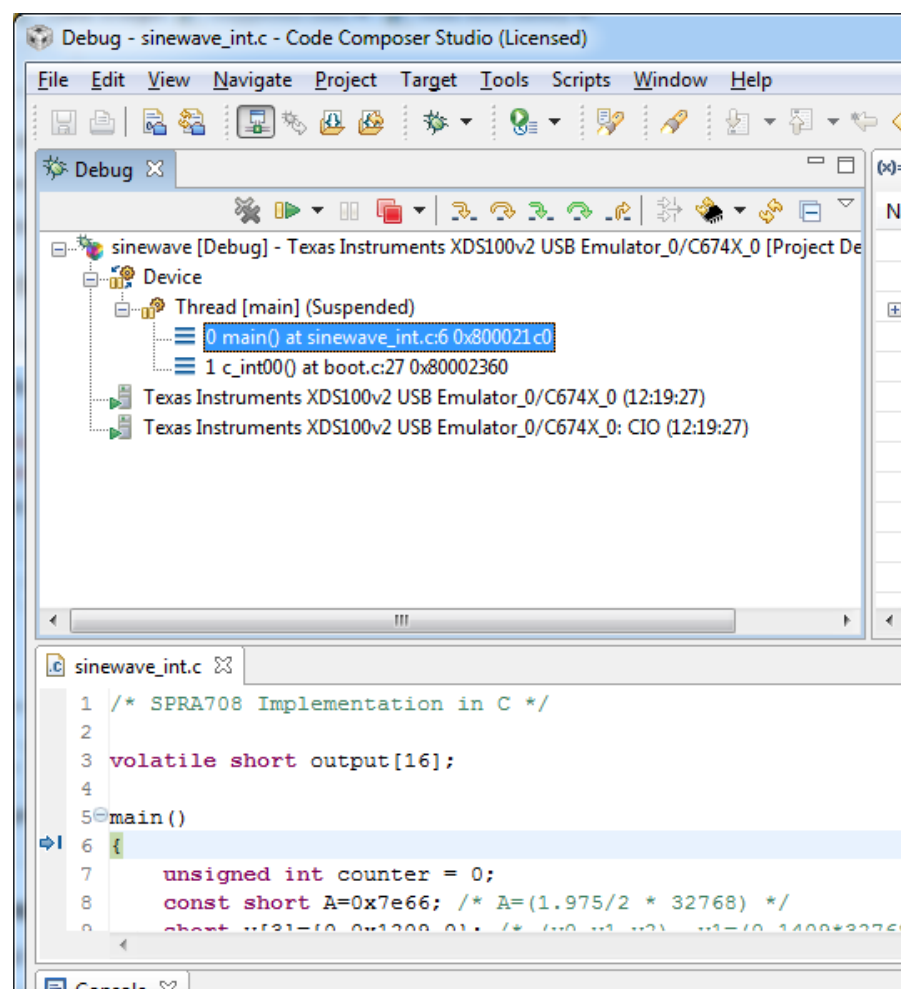


图 2.2.17 设备连接、调用堆栈和源代码

- “Debug（调试）”视图包含每个芯片核的目标配置和调用堆栈。
- 源代码视图显示了在 `main()` 处停止的程序。
- 基本调试功能（运行、停止、步入/步出、复位）位于“Debug（调试）”视图的顶部栏中。“Target（目标）”菜单还有其他几种调试功能。

通过转到菜单“Target -> Debug Active Project（目标 -> 调试活动项目）”启动调试器。如果目标配置需要先运行脚本再加载代码，将打开“Console（控制台）”视图。这些脚本采用 GEL（通用扩展语言）编写而成，在对包含复杂外部内存时序和电源配置的设备进行配置时尤其需要此类脚本。

```

sinewave [Project Debug Session] Texas Instruments XDS100v2 USB Emulator_0/C674X_0 (12:19:27)
C674X_0: Output:      mDDR initialization is in progress....
C674X_0: Output:      PLL1 init done for DDR:132MHz
C674X_0: Output:      mDDR init for 132 MHz is done
C674X_0: Output:      -----
  
```

图 2.2.18 GEL 输出

（2）监视变量和寄存器

在程序加载时还会打开“Local（本地）”和“Watch（监视）”视图，并显示本地和全局变量：

Name	Value	Address	Type
(x) A	-5543	0x80002000	short
(x) counter	3886454757	0x80001FFC	unsigned
(x) y	0x80002008	0x80002008	short[3]

图 2.2.19 查看变量

默认情况下不会打开寄存器视图，但是可通过转到菜单“View-> Registers（查看->寄存器）”进行查看。

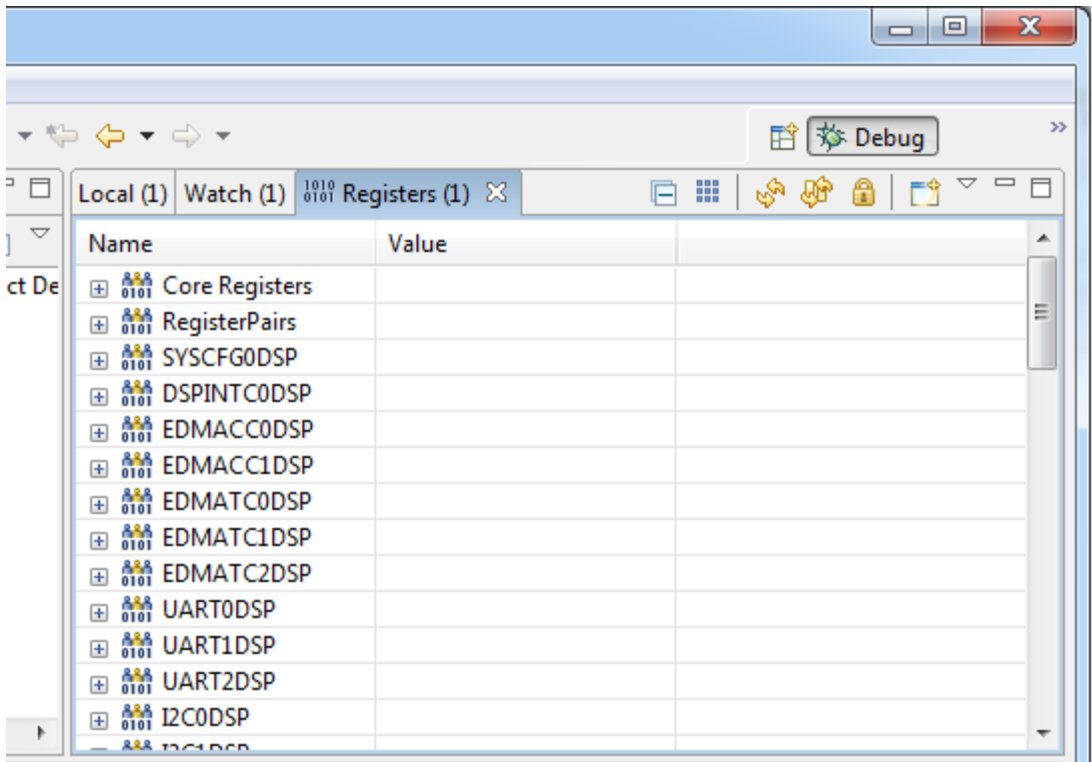


图 2.2.20 查看寄存器

（3）反汇编以及源代码与汇编代码混合模式

默认情况下不会打开反汇编视图，但是可通过转到菜单“View ->Disassembly（查看->反汇编）”查看。

反汇编窗口中一个极其有用的功能是源代码与汇编代码混合模式查看器，如上面的屏幕截图所示。要使用此功能，只需在“Disassembly（反汇编）”视图中右键单击并选择“View Source（查看源代码）”。

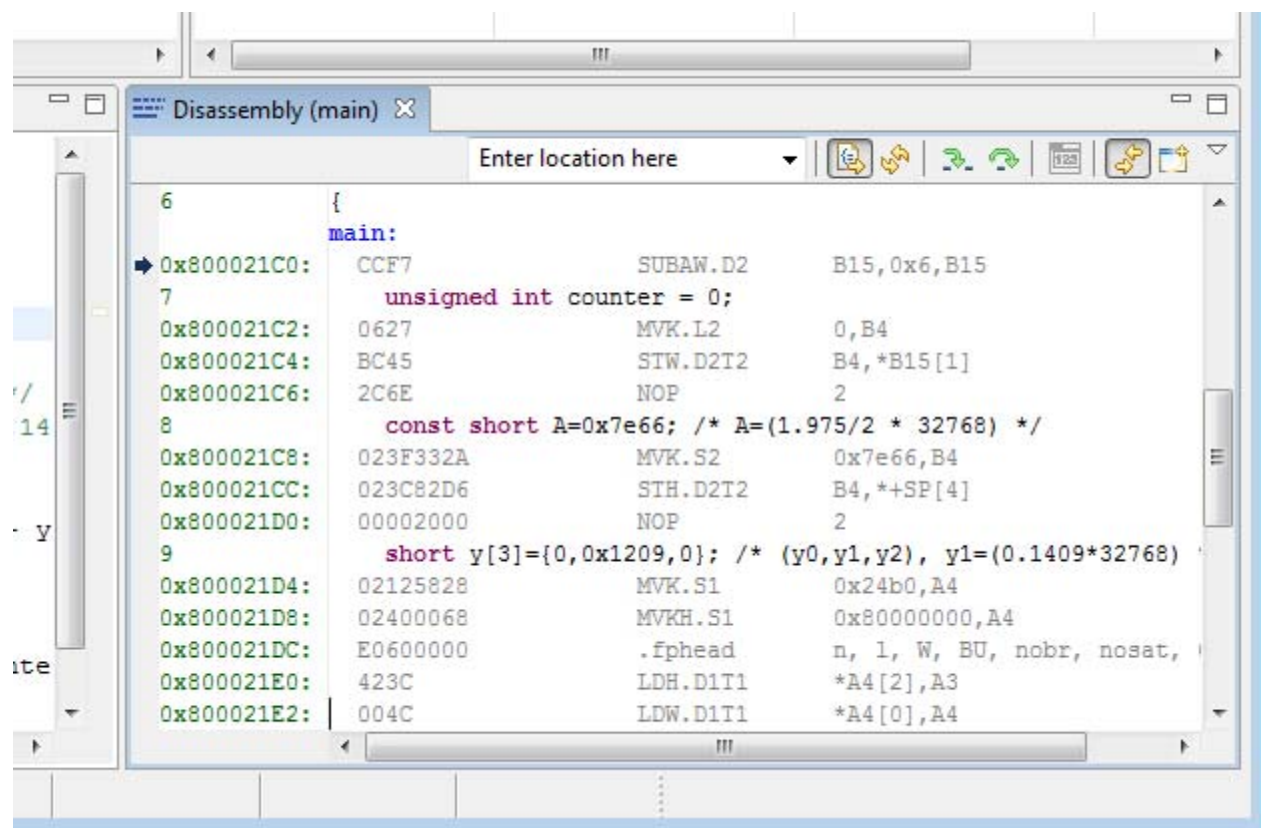


图 2.2.21 源代码/汇编代码混合视图

(4) 内存查看器

默认情况下不会打开内存视图，但是可通过转到菜单“View -> Memory（查看 -> 内存）”查看。

通过此屏幕可访问一些有用的功能：内存可通过多种格式进行查看，可填充任意值，也可保存至 PC 主机中的二进制文件或从中加载，此外还可以查看所有变量和函数，而且每个内存位置都有上下文相关的信息框。

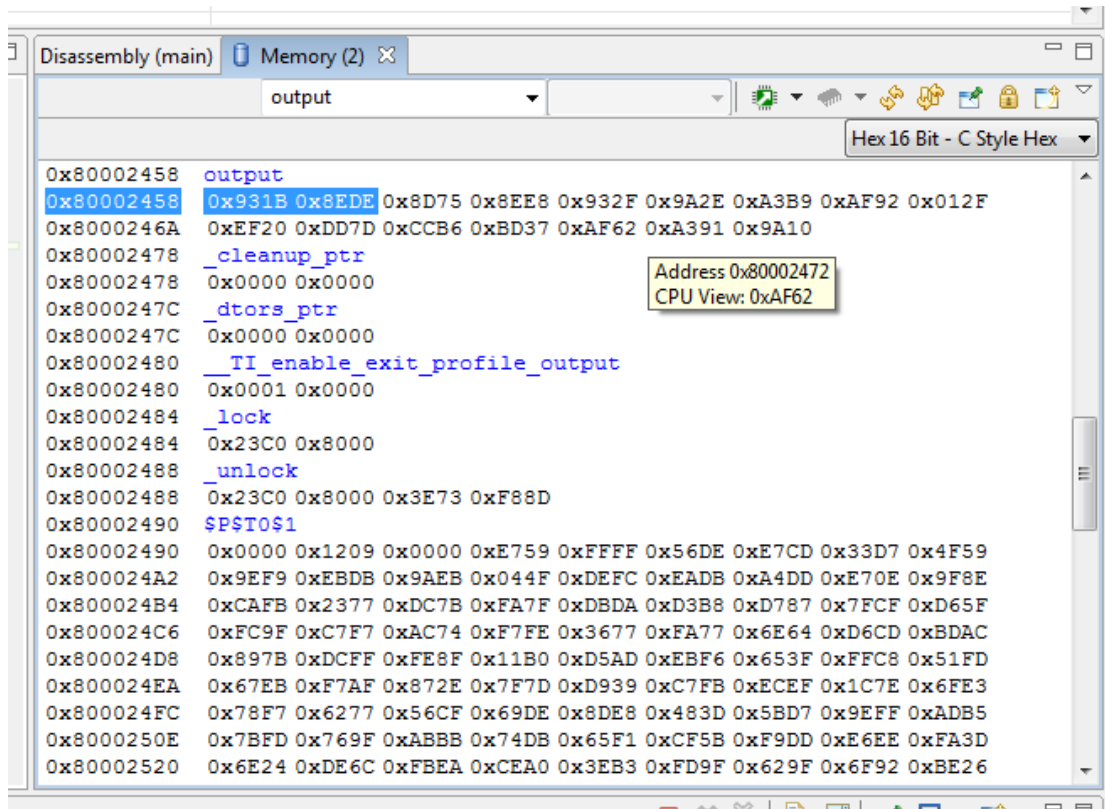


图 2.2.22 查看内存

(5) 管理断点

作为任何调试器都会拥有的最基本功能，CCSv4 中的断点添加了一系列选项，帮助增加调试进程的灵活性。

- 硬件断点可从 IDE 直接进行设置；
- 软件断点仅受到设备可用内存的限制；
- 软件断点可设置为无条件或有条件停止；
- 除了停止目标之外，软件断点还可执行其他功能：文件 I/O 传输、屏幕更新等。

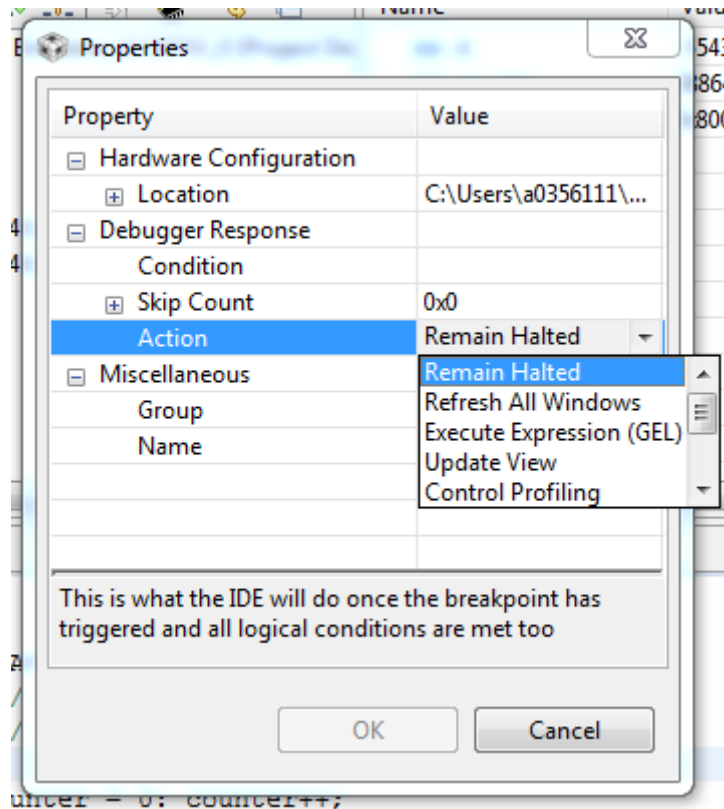




图 2.2.23 断点选项

要设置断点，只需在源代码或反汇编视图中双击代码行即可。硬件  或软件  断点的图标会指示其状态和放置位置。

所有断点（软件、硬件、已启用、已禁用）都可在断点查看器中看到。

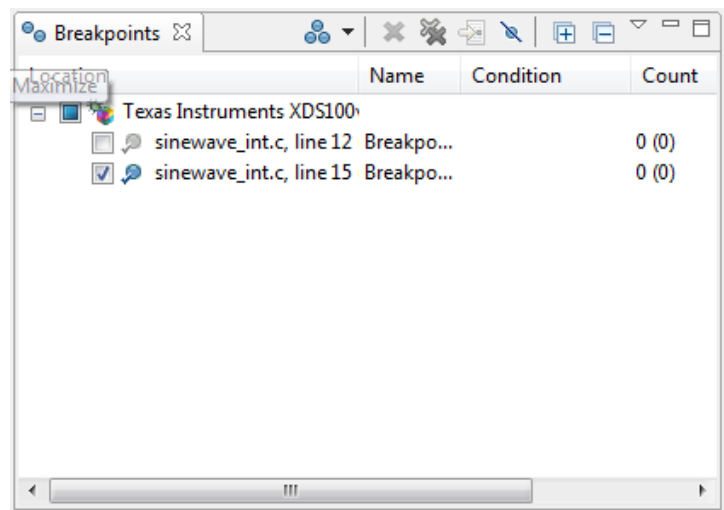


图 2.2.24 断点查看器

要配置断点，只需右键单击蓝点，或者在断点视图中右键单击并选择“Breakpoint Properties...（断点属性...）”。

- 使用“Action（操作）”可以设置断点的行为，例如保持停止、更新一个或所有调试器视图、从文件中读取数据或将数据写入其中、激活或停用断点组等。
- 使用“Skip Count（跳过计数）”可以设置执行断点操作之前通过的数目。
- 使用“Group（分组）”可以对断点进行分组以进行高级控制。

（6）图形显示工具

CCSv4 中提供了一个高级图形和图像可视化工具。它可通过图形形式显示数组，并且可采用多种格式。

要添加图形，只需转到菜单“Tools -> Graph（工具 -> 图形）”，然后从各种显示选项中选择一個。

- 基于时间的图形：“Single Time（单曲线图）”和“Dual Time（双曲线图）”
- 基于频率的图形：所有 FFT 选项

图形窗口中的顶部工具栏可控制多种功能，例如更新速率（冻结、连续、目标停止时或手动）、缩放、配置属性等。



图 2.2.25 图形工具栏


默认情况下，图形窗口会在目标停止时立即更新、使用自动缩放并以样本数显示 X 轴，以整数值显示 Y 轴。所有这些选项都可进行设置。请记住，图形更新时所传输的数据量可能会影响目标硬件的实时操作。

下面的过程显示了包含正弦波发生器输出内容的图形。

- 在源代码窗口中，右键单击断点蓝点（已在上一部分设置）并选择“Breakpoint Properties...（断点属性...）”。
- 在“Action（操作）”属性中，单击该属性值并选择“Refresh All Windows（刷新所有窗口）”。这样将刷新所有窗口，而不是将程序完全停止在该点。

- 变量 `output[]` 包含 16 个正弦波发生器输出样本，因此整个缓冲区必须立即显示在图形窗口中。单击“Tools -> Graph -> Single Time（工具 -> 图形 -> 单曲线图）”，然后将选项配置如下：

属性	值
采集缓冲区大小	16
Dsp 数据类型	16 位带符号整数
Q_value	15
开始地址	output

- 屏幕底部应该出现一个图形窗口。如果需要，可通过单击  按钮更改图形属性。
- 单击“Target -> Run（目标 -> 运行）”。该图形应该以 16 个样本为一组分批更新。
- 要查看 `output` 数组的实际值，请单击“Watch（监视）”选项卡（应当在屏幕右上角部分），然后单击“New（新建）”。键入 `output` 并展开此数组以显示其中的所有值。这些值以 16 位带符号整数输出，因此可通过调整 Q 值使其标准化：在“Watch（监视）”窗口中选择所有值，右键单击并选择“Q-values -> Q-value(15)（Q 值 -> Q 值(15)）”。

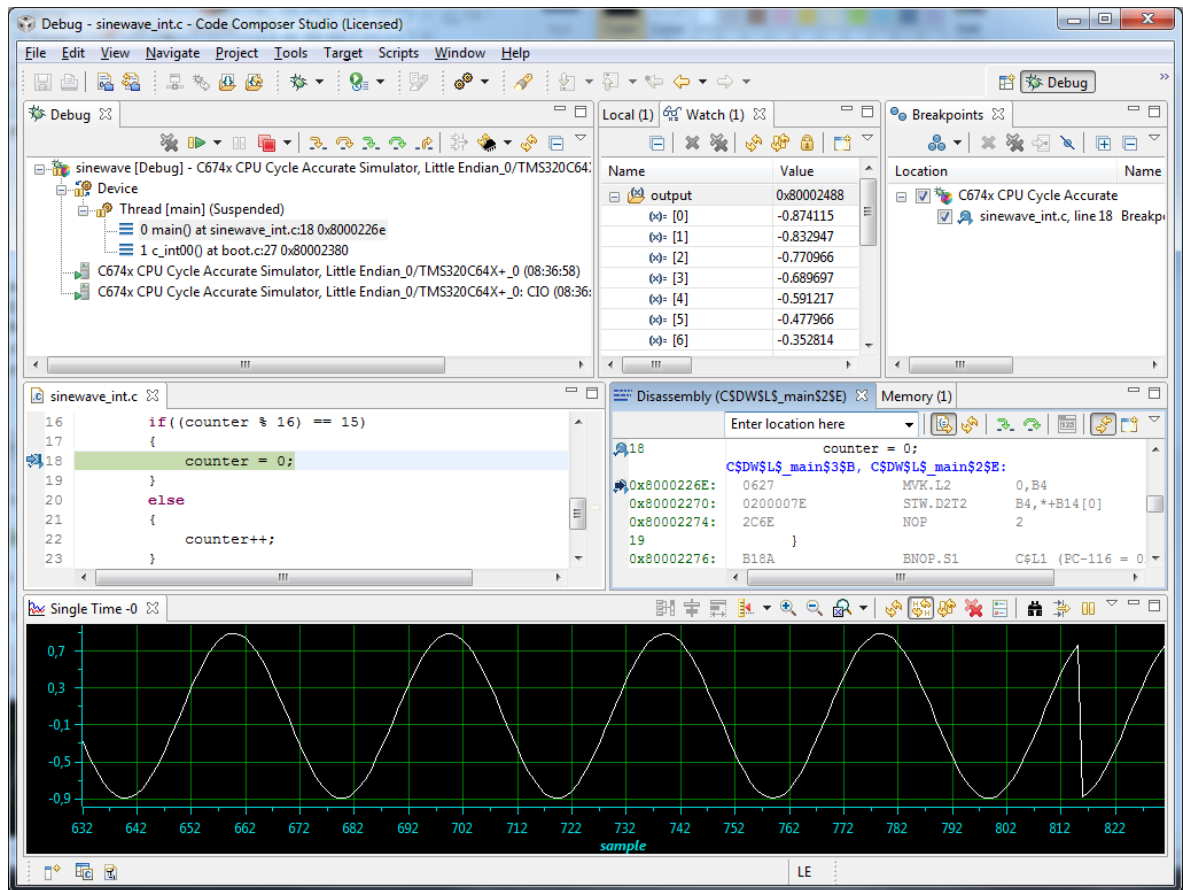


图 2.2.26 正弦波示例

(7) 图像显示工具

要显示图像，只需转到菜单“Tools -> Image（工具 -> 图像）”。


屏幕底部将打开两个视图：“Image（图像）”和“Properties（属性）”。

CCSv4 显示的信息既可以是来自 PC 主机中的文件，也可以是目标开发板中加载的图像。在属性页面中，只需将“Image source（图像源）”选项设置为“File（文件）”或“Connected Device（连接的设备）”即可。

与图形查看器类似，需要设置其他所有属性才能使显示内容有意义。彩色障板、线条尺寸和数据宽度等几种选项会影响图像的正确显示。

要显示加载至目标的图像，请执行以下操作：

- 转到菜单“View -> Memory（查看 -> 内存）”打开内存视图。
- 在地址框中键入有效的目标地址：0xC0000000

- 将图像文件 <sample_24bpp.dat> 加载至 0xC0000000：单击内存操作图标  旁边的三角形，然后单击“Load（加载）”。浏览至下面的目录，然后单击“Next（下一步）”。 C:\Program Files\Texas Instruments\ccsv4\c6000\examples
- 键入与内存窗口中相同的起始地址，并将“Type-size（类型大小）”设置为 32 位。
- 按下图所示设置属性：

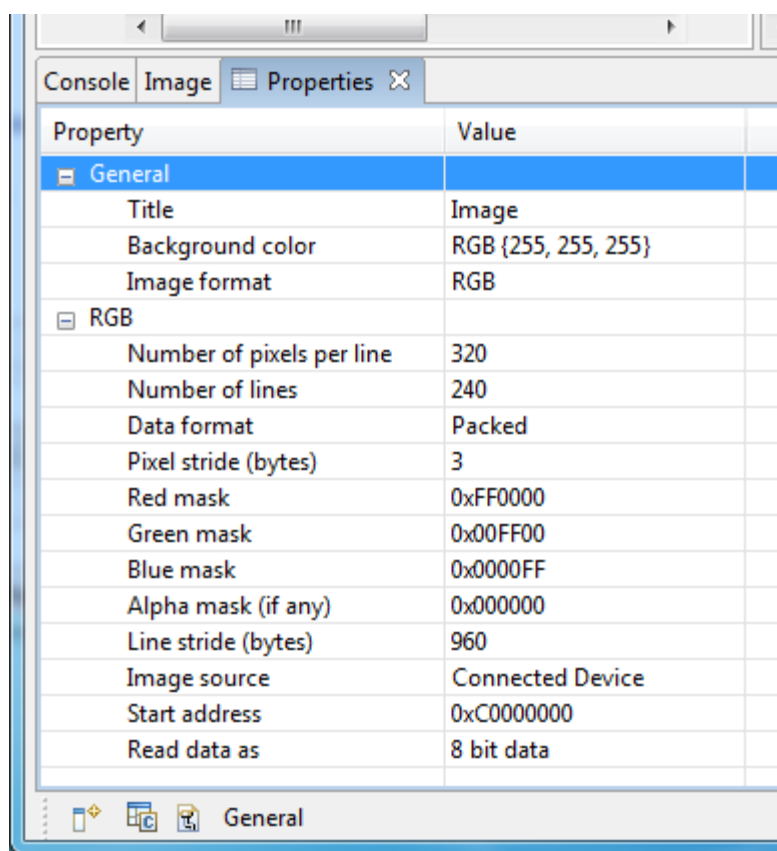


图 2.2.27 图像属性

- 选择“Image（图像）”选项卡，然后右键单击并选择“Refresh（刷新）”。应该会显示下面的图像。

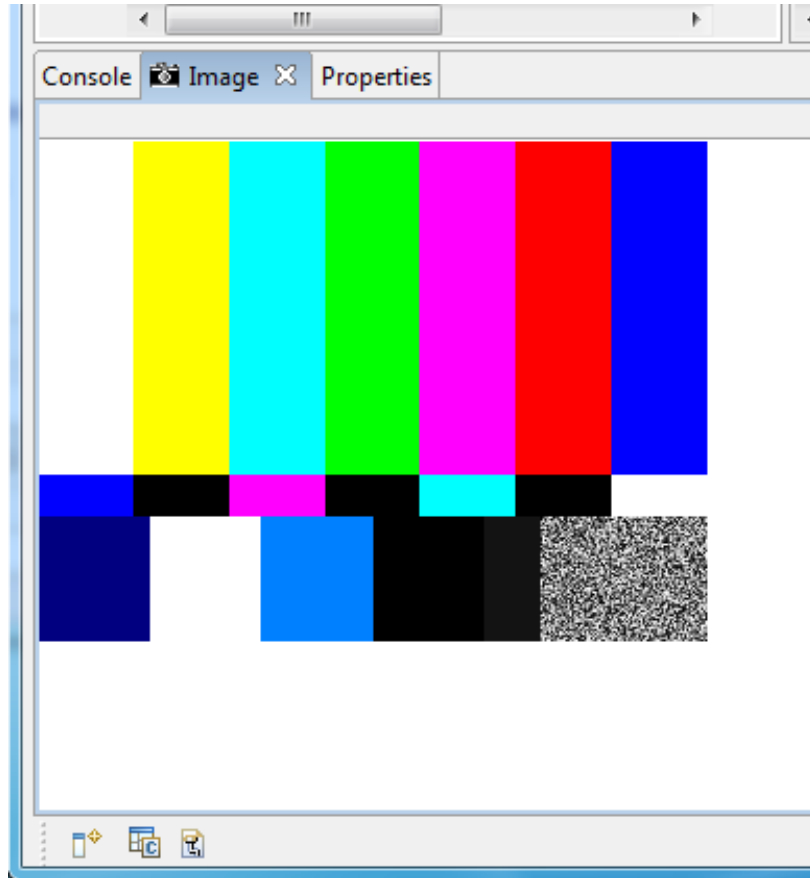


图 2.2.28 图像显示

3. OMAP L138 EVM 使用说明

3.1 简介

OMAP L138 EVM 开发工具是以 OMAP L138 为主控制器的开发套件,OMAP L138 是一款双核(包括一个 DSP 与一个 ARM),我们使用其中的 DSP,即 C6748 进行 DSP 的学习



图 3.1.1 OMAP L138 EVM 开发工具

开发板工具包括：底板、LCD 屏、UI 界面、主控制器、SD 卡、电源、网线、串口线以及 USB SD 卡读卡器，其中主要使用底板与主控制器。

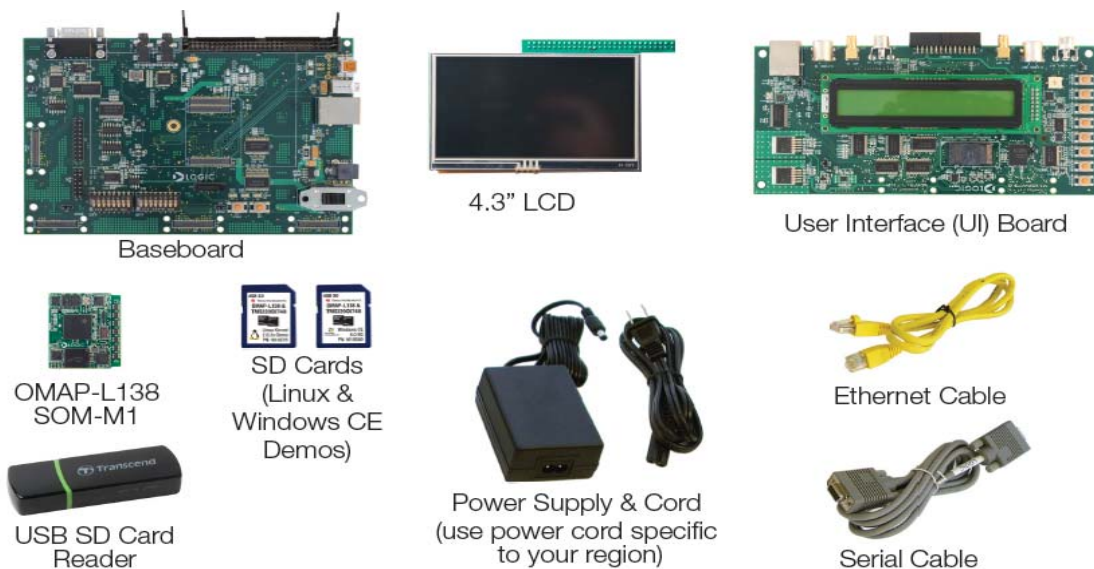


图 3.1.2 OMAP L138 EVM 开发工具

Omap-L138 主控制器包括主 CPU、1Gbit 的 mDDR 内存，64Mbit 的 Flash 以及一些其他接口。内存与 flash 相对较为重要，后面实验中会用到。

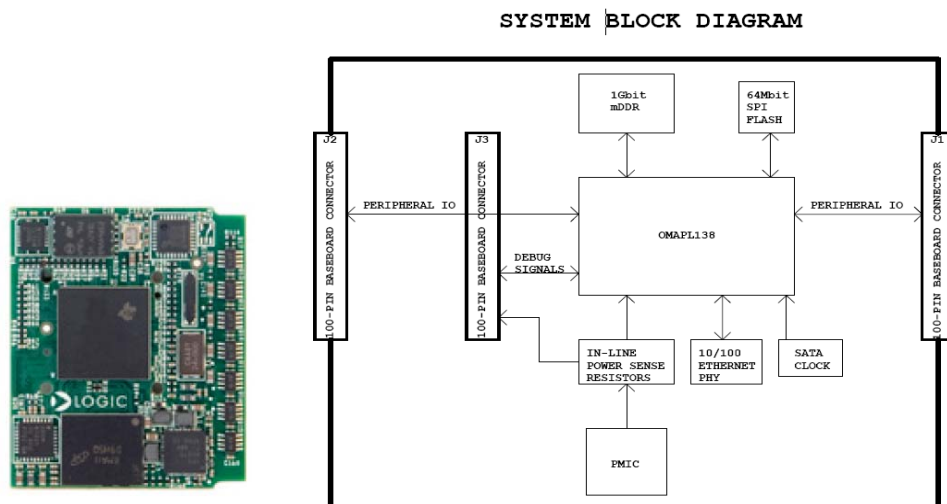


图 3.1.3 主控制板及框图

下面是开发板的各部分简介。

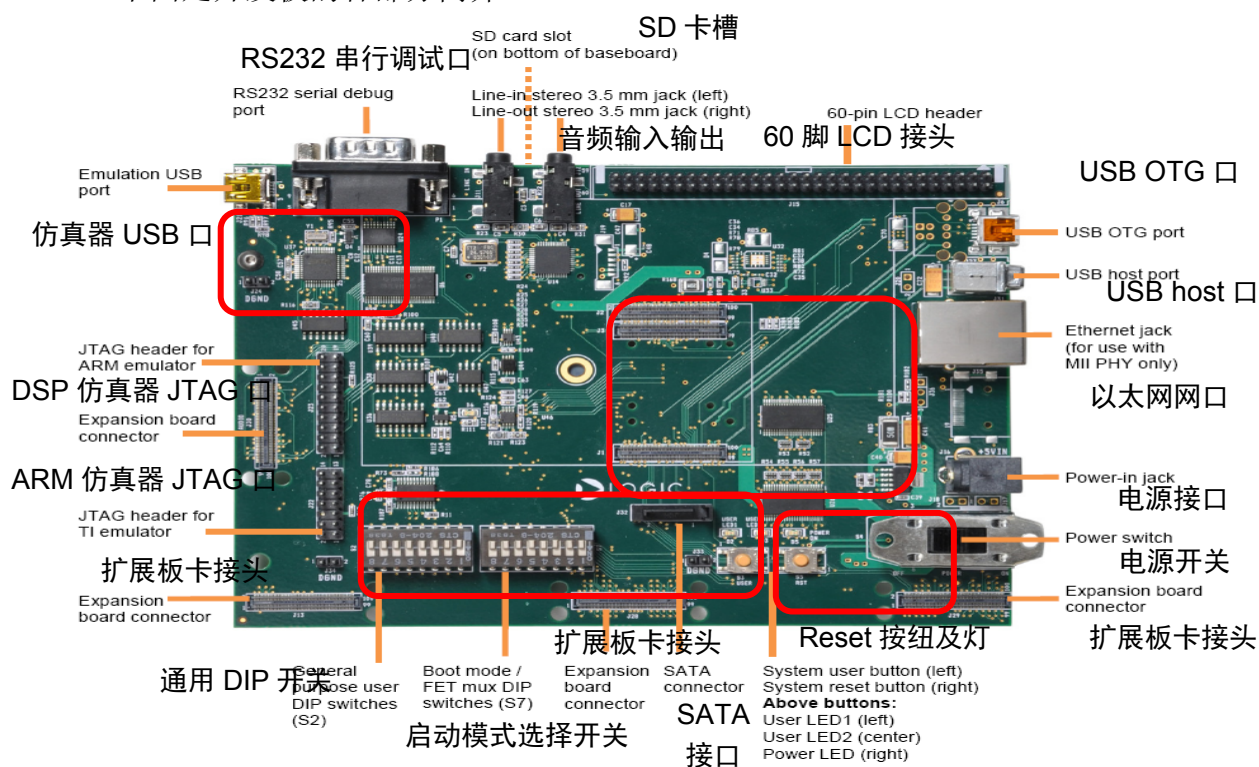


图 3.1.4 各部分介绍

开发板外设主要有：

- 仿真器接口：使用 CCS 对开发板进行开发调试的接口
- 音频输入输出：与 DSP 中的 McASP 相通，可以接耳机与麦克风
- 电源接口：5V 电源，为开发套件提供电源
- 电源开关：5V 电源的开关
- 通用 DIP 开关：与通用 IO 口连接，可作为 DSP 的数字输入

3.2 开发板使用步骤

1. 连接控制器板与底板

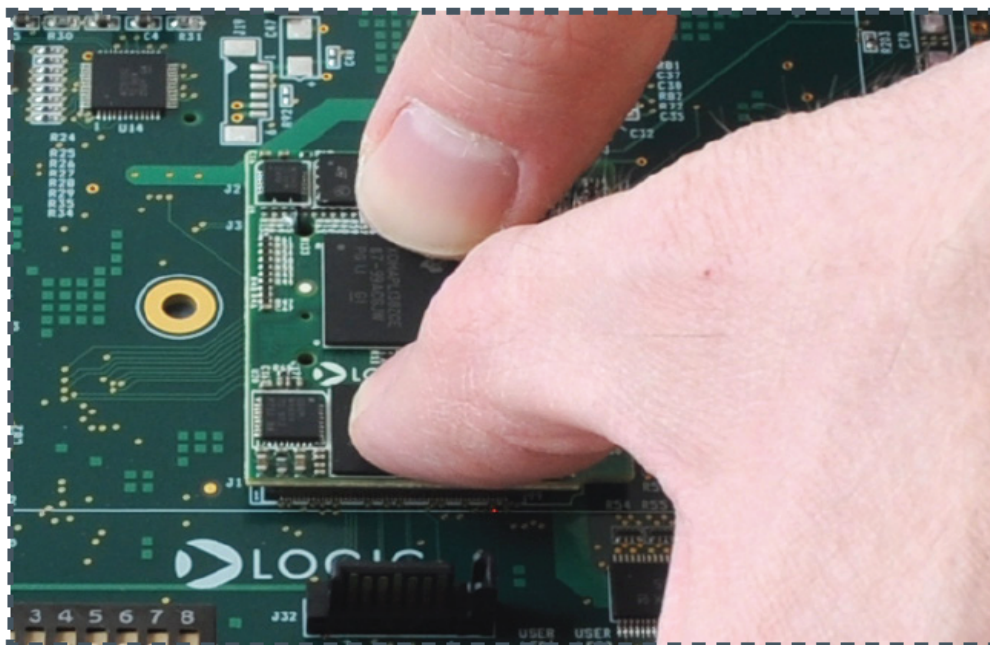


图 3.2.1 安装主控板

2. 检查所有的拨码开关，确保全部在 OFF 位置

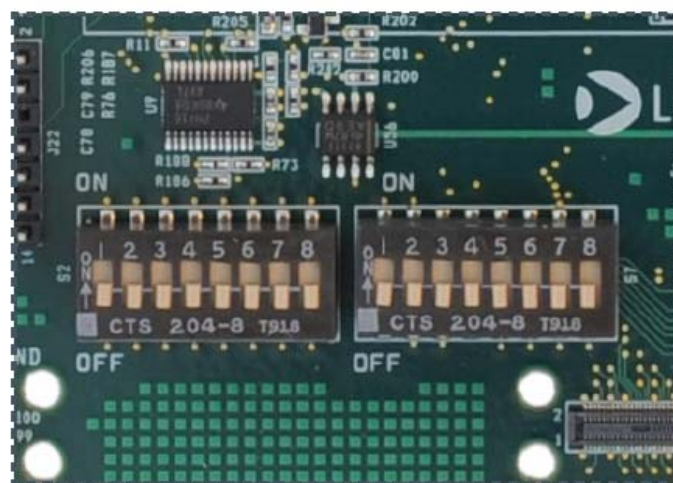


图 3.2.2 检查拨码开关

3. 连接电源，合上开关即可

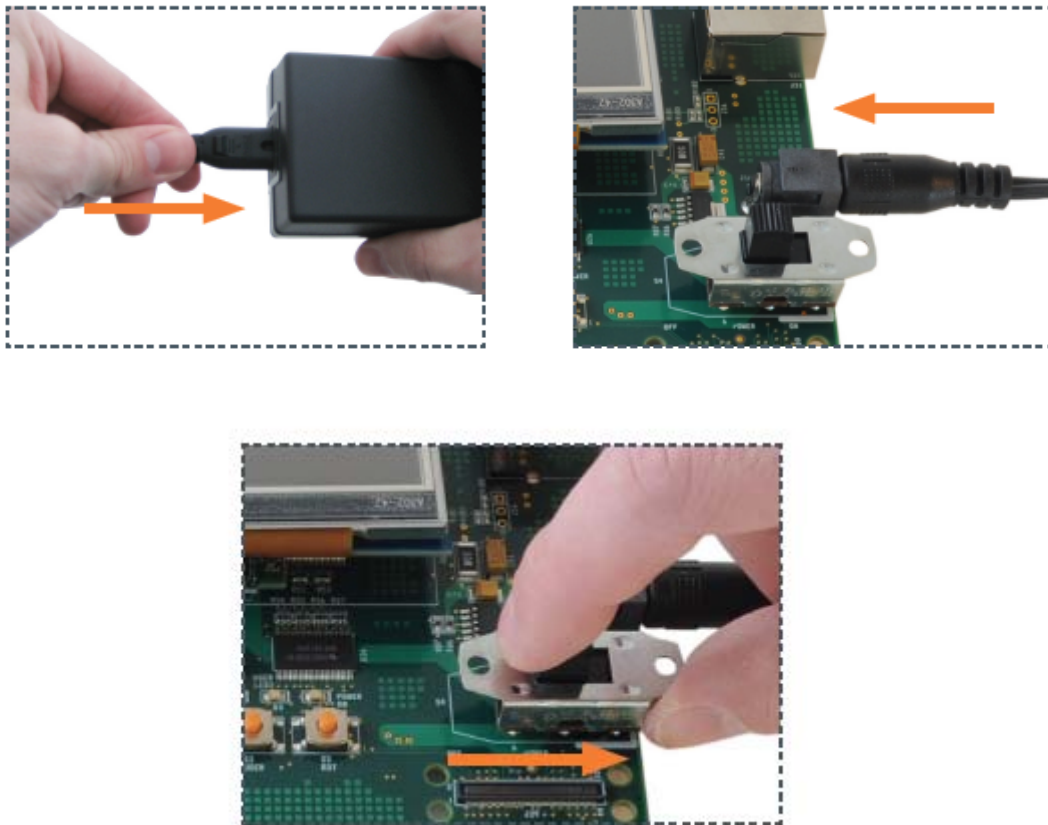


图 3.2.3 连接电源

4. 安装 CCS

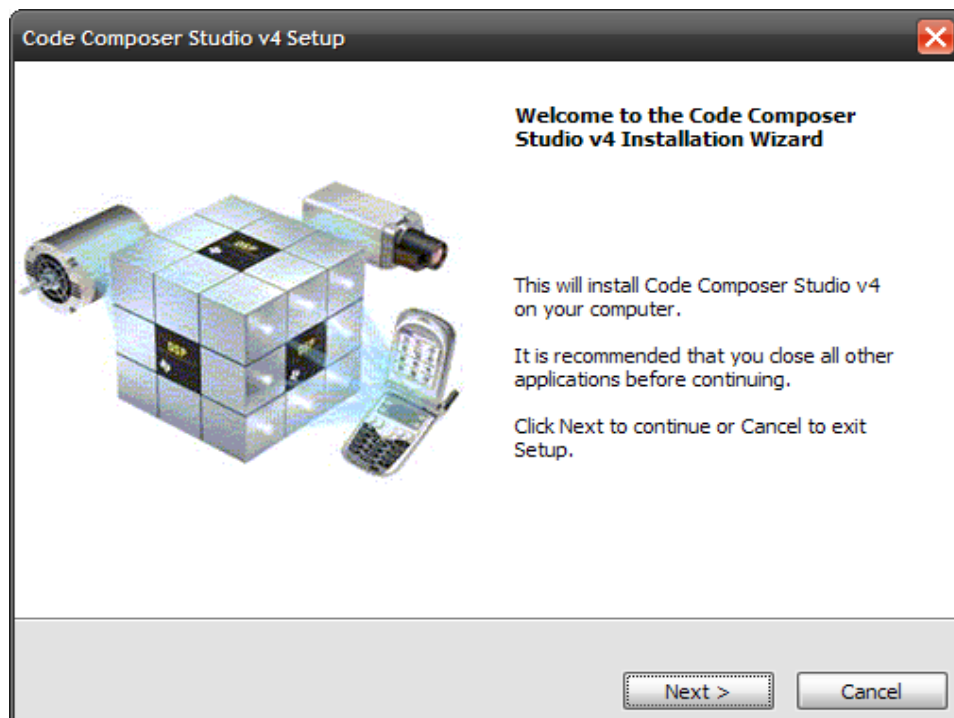


图 3.2.4 安装 CCS

5. 连接 PC 与开发板（注：不要搞错 usb 接口）

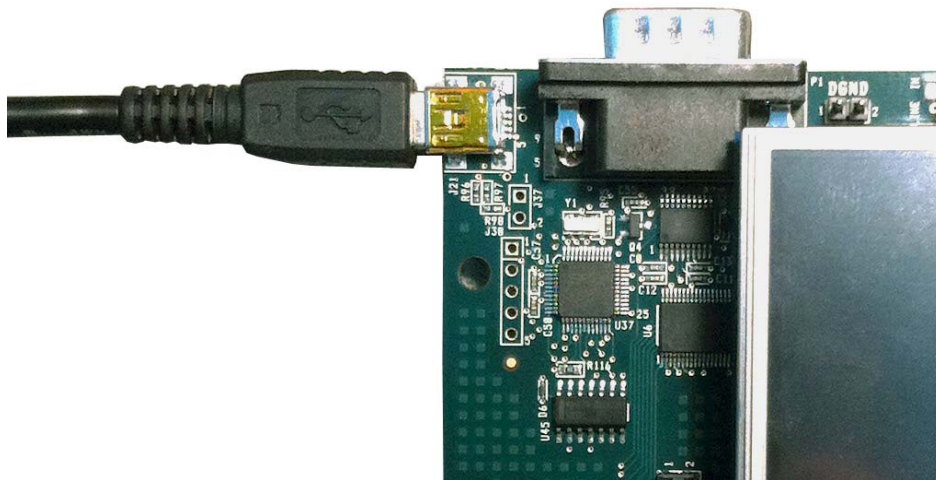


图 3.2.5 连接仿真器

6. 启动 CCS 后导入已有的程序

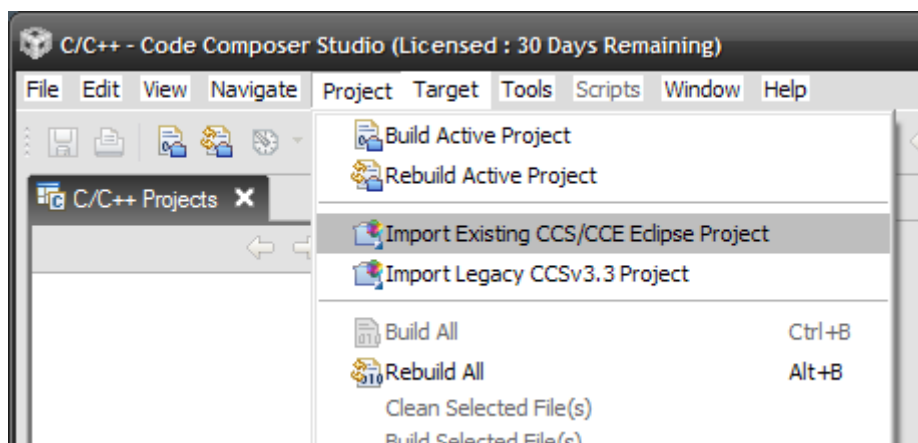


图 3.2.6 导入示例程序

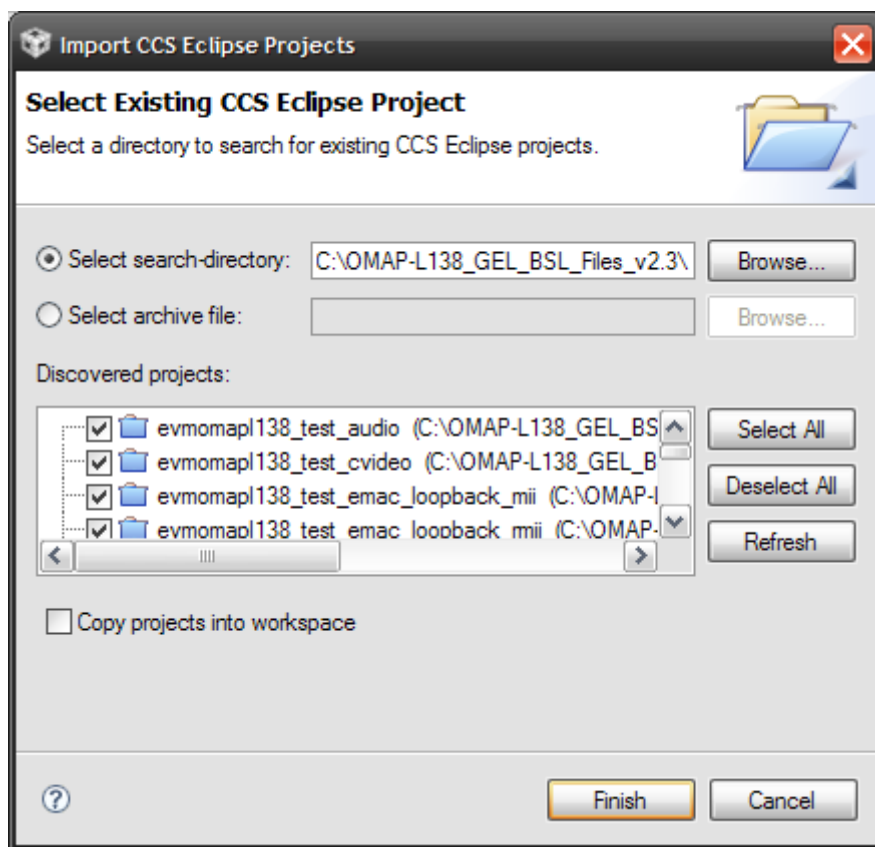


图 3.2.7 选择导入的程序

7. 将目标程序设为有效，并编译

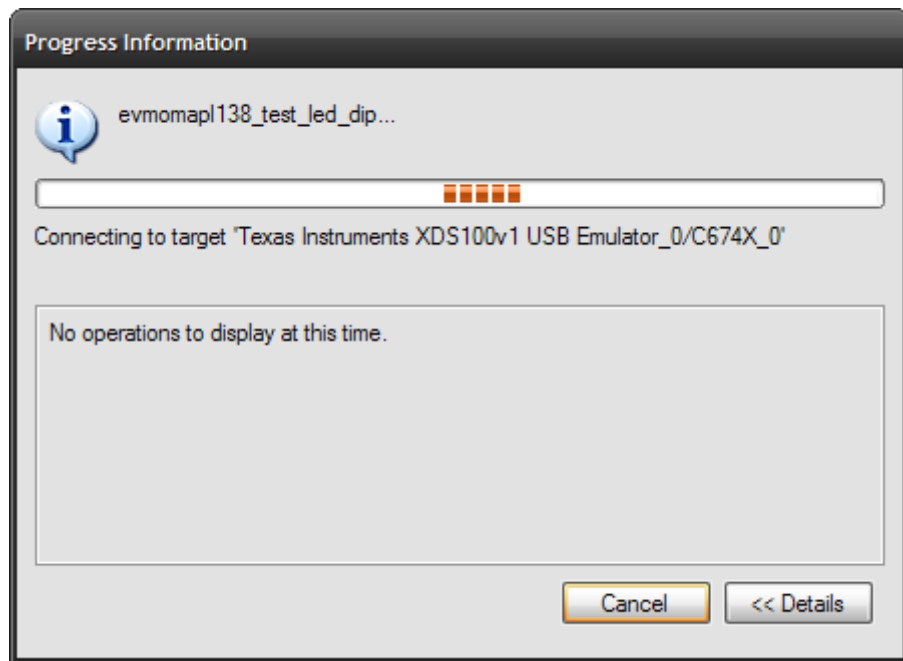


图 3.2.8 调试目标程序

9. 点击运行，并在窗口中观测输出

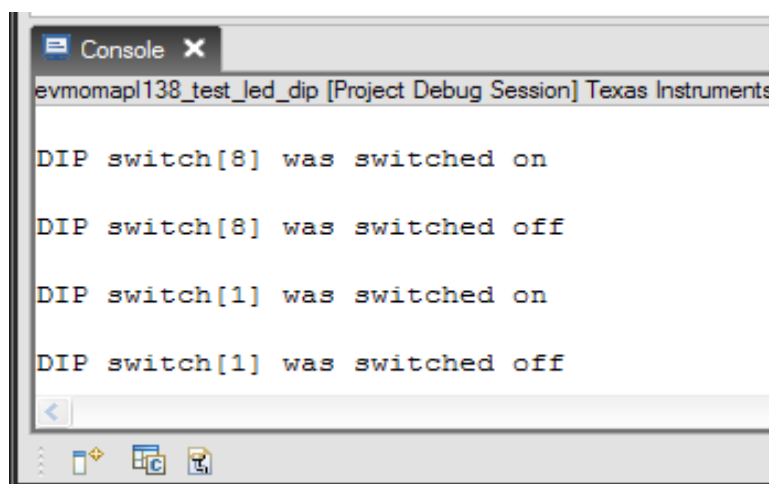


图 3.2.8 运行并观测输出

4. 实验具体要求

4.1 实验一

根据拨码开关的状态，来选择一定的地址对外部 RAM 进行读写，了解 IO 扩展的方式以及 RAM 地址的用法。具体的实验要求如下：

1. 输出提示，请输入

2. 拨动拨码开关，同时 DSP 记录拨码开关数值，并写入 RAM
3. 输出提示，并输出 RAM 中相应位的数值



图 4.1.1 原理

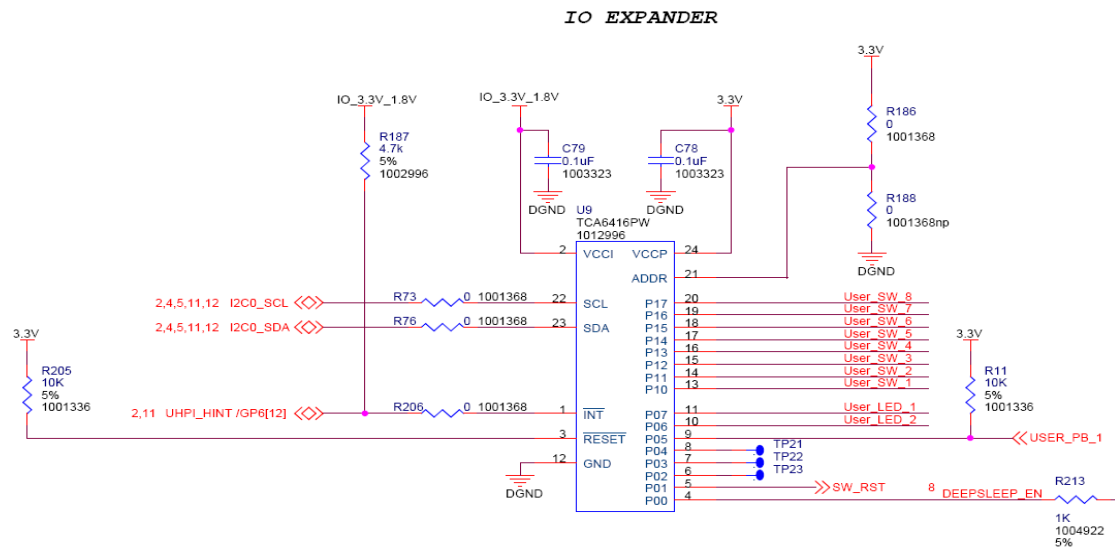


图 4.1.2 拨码开关部分扩展原理图

以下是 RAM 分配的示例：

```

-stack          0x00000800
-heap           0x00000800
MEMORY
{
    dsp_l2_ram:    ORIGIN = 0x11800000   LENGTH = 0x00040000
    shared_ram:    ORIGIN = 0x80000000   LENGTH = 0x00020000
    external_ram:  ORIGIN = 0xC0000000   LENGTH = 0x08000000
    arm_local_ram: ORIGIN = 0xFFFF0000   LENGTH = 0x00002000
}
SECTIONS
{
    .text          > shared_ram
    .const         > shared_ram
    .bss           > shared_ram
    .far           > shared_ram
  
```



```

.switch    > shared_ram
.stack     > shared_ram
.data      > shared_ram
.cinit     > shared_ram
.systemem  > shared_ram
.cio       > shared_ram
}

```

4.2 实验二

根据示例程序,通过音频口发出 1234567 的声音,能实现简易乐谱效果更好。
具体要求如下:

1. 根据提示, 输出相应频率的音符
2. 可以根据发音的长短需要进行控制
3. 尝试不同的发音组合, 以实现简易乐谱
4. 探索音频输入的配置, 实现输入与输出



图 4.2.1 原理

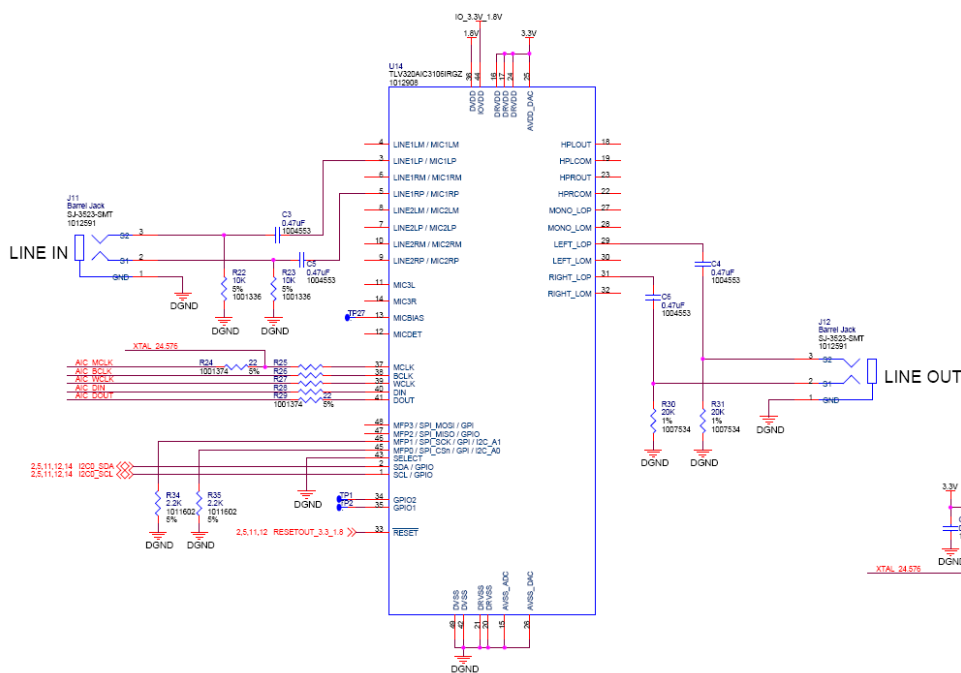


图 4.2.2 拨码开关部分扩展原理图

4.3 实验三

对一段离散数据进行 FFT 分析，并将其输出结果呈现在输出窗口中，以增加对算法的认识。其具体要求如下：

1. 无示例程序
2. 熟悉并编写 FFT 算法
3. 自行生成至少 128 点的数据，并进行 FFT 分析
4. 输出结果显示在图像窗口