# TIPS & TRICKS
# 技巧与窍门

# Tips and Tricks技巧与窍门

- 工作区 **Workspaces**
- 窗口和视图（基本）**Windows and Views (basics)**
- 项目 **Projects**
  - 共享项目 Sharing Projects
  - 源代码管理 Source Control
  - 编辑器技巧 Editor Tips
- 调试 **Debugging**
  - 调试配置 Debug Configurations
  - 没有项目的调试 Debugging Without a Project
  - 图表 Graphs
  - 模块视图 Modules View
- 脚本控制台 **Scripting Console**

# 工作区
# WORKSPACES

# Eclipse概念：工作区（Workspace）

- **CCS的主要工作文件夹**
- **包含被其定义的所有项目的管理信息**
  - 创建新项目时所使用的默认地点
- **用户的喜好(User preferences)，自定义的观点(custom perspectives)，插件的缓存数据(plug-in cache)，等等，都存储在工作区**
- **工作区概念不可与CCSv3工作区文件（*. WKS）混淆**
- **可维持多个工作区**
  - 每个CCS实例仅可以同时打开一个工作区
  - 相同的工作区不能共享多个CCS的运行实例
  - 不建议用户之间共享工作区

# 使用多个工作区

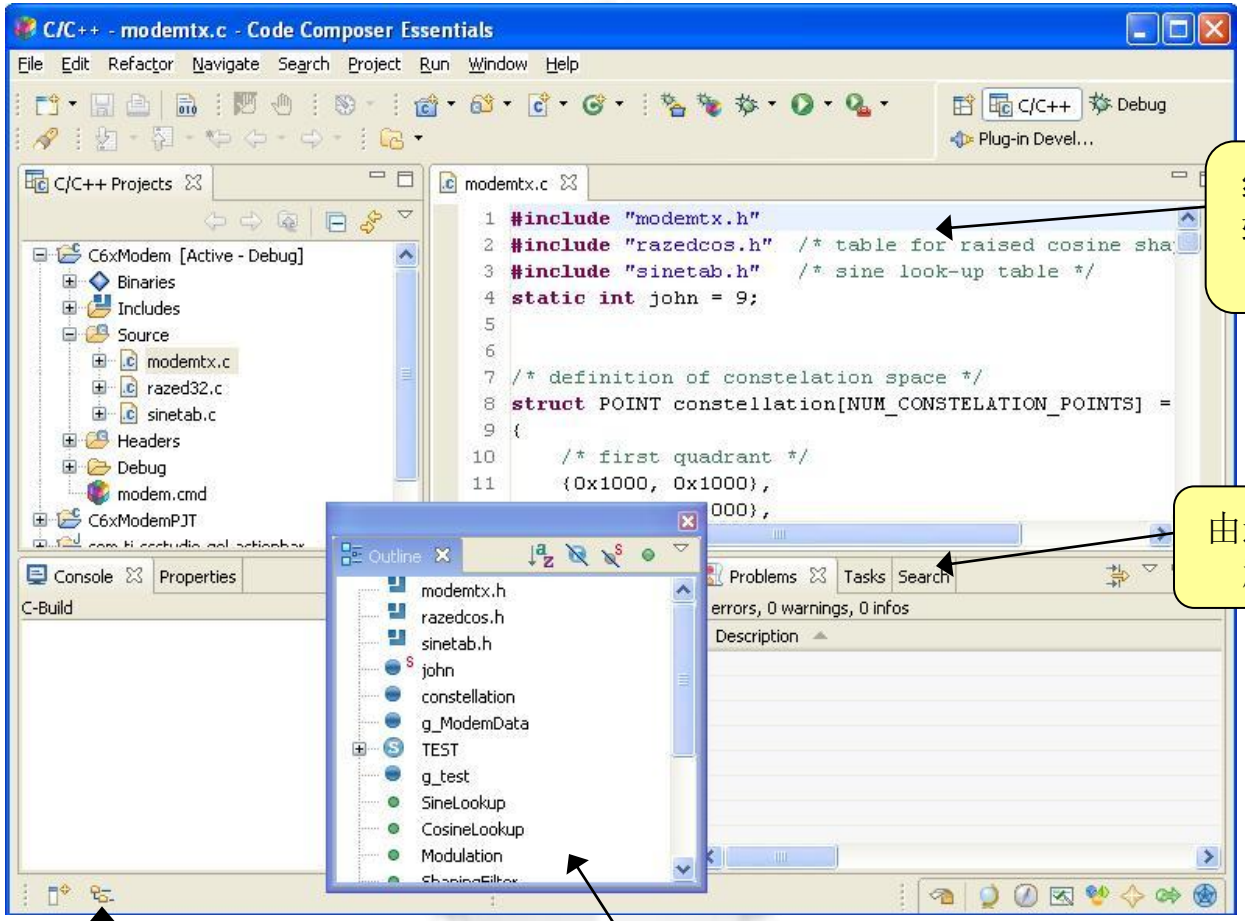
- 多用户：请每个用户在共享的计算机上建立的各自的工作区
  - 自定义首选项，布局等将保持在每个用户的工作区内
  - 每个用户都可以在他们的工作区内，做只适用于的自己环境的具体项目
- 项目组织管理：为更好的维护您的CCS项目，您可以把所有独立项目分解成单独的工作区
  - 每个发行版对应一个工作区
  - 每个模块/功能的工作区
  - 等等
- 性能：工作区的内容越大（打开的项目数），对CCS的性能的影响就越大

# 清理您的工作区

- 随着时间的推移，工作区文件夹可能会遭到破坏
- 为了获取最好的CCS的性能和稳定性，请定期清理您的工作区
- 可以通过以下途径清理工作区：
  - 在工作区（workspeace）文件夹删除“.metadata”文件夹
  - 使用新的工作区文件夹
- 请在清理前保存当前工作区的设置，然后导入到新的工作区
  - 保存设置：“文件 - >导出...->一般>”首选项“ - >到首选项文件”
  - 'File->Export...->General->Preferences->To preference file
  - 导入设置：“文件>导入...->一般 - >首选项 - >从首选项文件”
  - 'File->Import...->General->Preferences->From preference file

# 窗口与视图
# WINDOWS AND VIEWS

# 窗口类型




编辑器：只有编辑窗口是这个组别的一部分

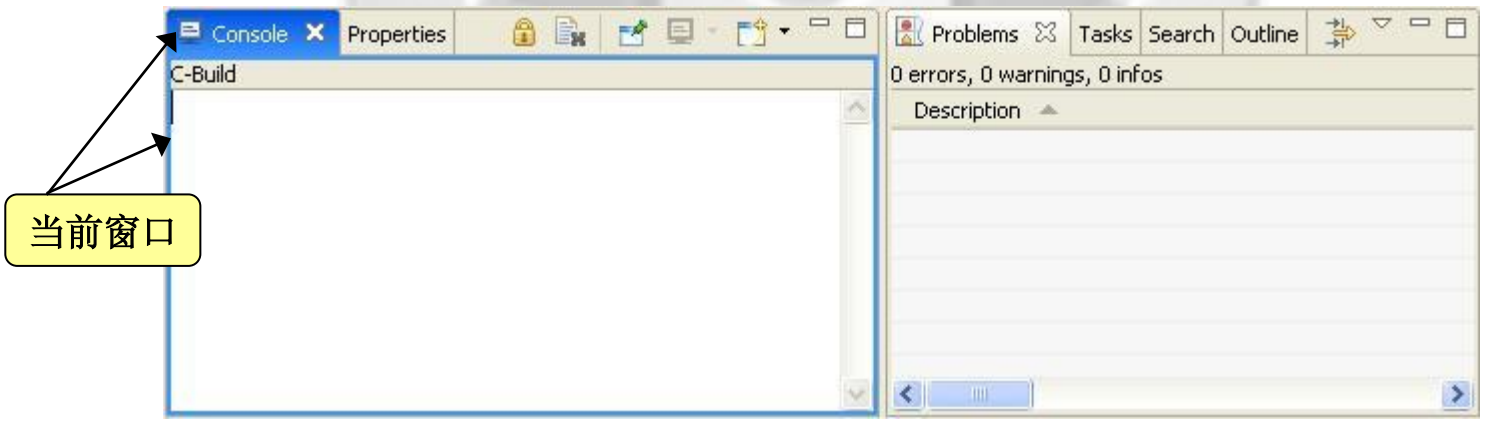
由选项卡组合在一起的几个窗口

快速视图(**Fast view**)：隐藏窗口，通过图标按钮将它们还原。点击窗口外恢复隐藏状态。

分离窗口：就像在CCSv3主窗口中浮动的窗口

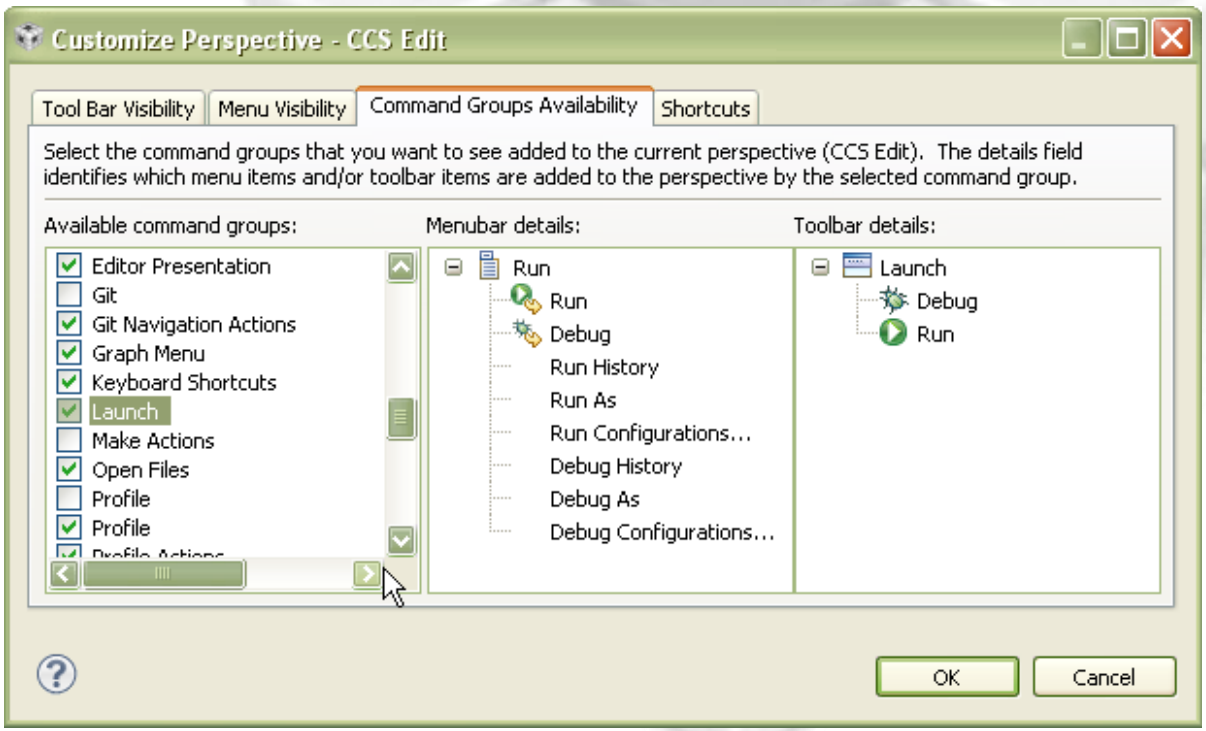
# 窗口技巧

- 双击一个窗口的标题栏将窗口最大化
- 双再次点击将它恢复到原来的大小
- 快速视图对那些你不频繁使用，但是需要时却会占用很大桌面的空间窗口非常适用
- 焦点（focus）中的窗口会显示一个蓝色的标题和边框



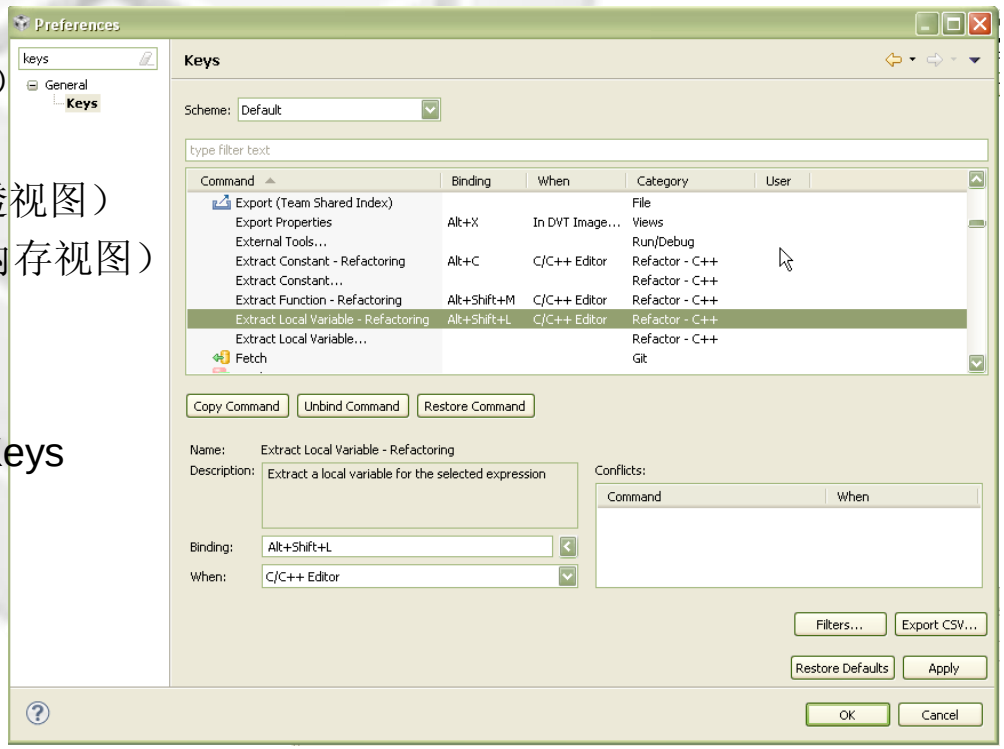
# 自定义透视图

- 您可以自定义透视图中的菜单项和工具栏
- 右键单击工具栏
  - 选择“自定透视图”（ customize perspective ）



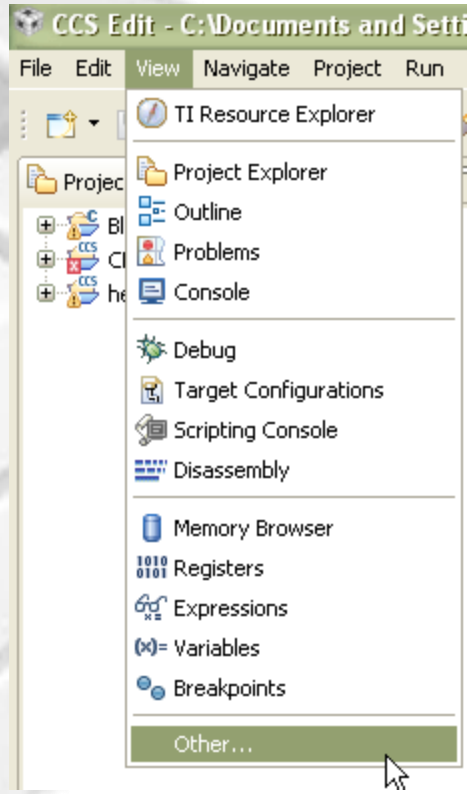
# 使用键盘快捷键

- 对于喜欢使用键盘快捷键的用户，这里是一些有用的默认快捷键：
  - CTRL+ SHIFT + P：构建激活的项目（所有的透视图内）
  - CTRL + ALT + P：重建激活的项目（所有的透视图内）
  - Ctrl + F：查找和替换（在编辑器窗口）
  - CTRL+ H：在文件系统中查找文件
  - F8：Free 运行（在CCS调试透视图）
  - F5，F6：C 单步步入、步出
  - Alt + C：连接到目标（在CCS调试透视图）
  - CTRL+ G：转到地址（在反汇编和内存视图）
- 可在此查看和修改所有键的绑定：
  - “窗口 - >首选项 - >常规 - >键”
  - Window->Preferences->General->Keys



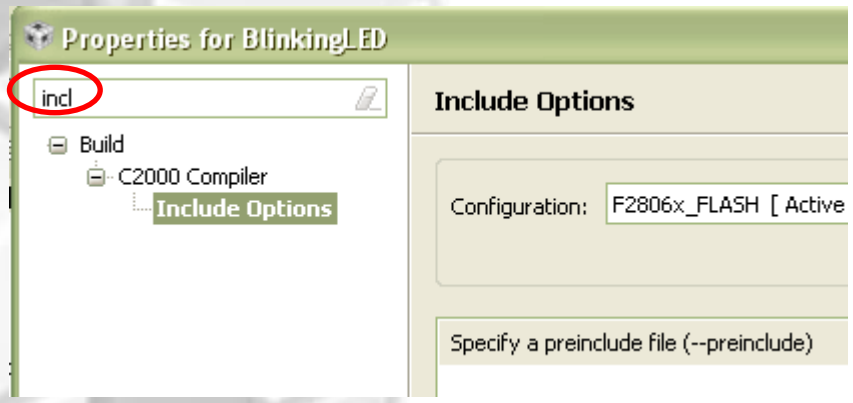
# 访问视图 (Accessing Views)

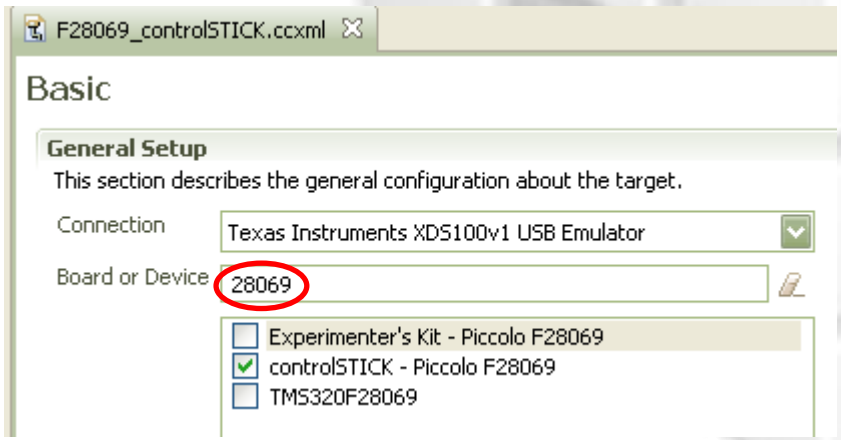
- 打开一个新的视图：Windows - > Show View
  - 当前透视图的常用视图
  - 最近打开过的视图
- 要访问未列出的视图，选择 Others…
- ”视图(View)”主菜列出最流行的视图

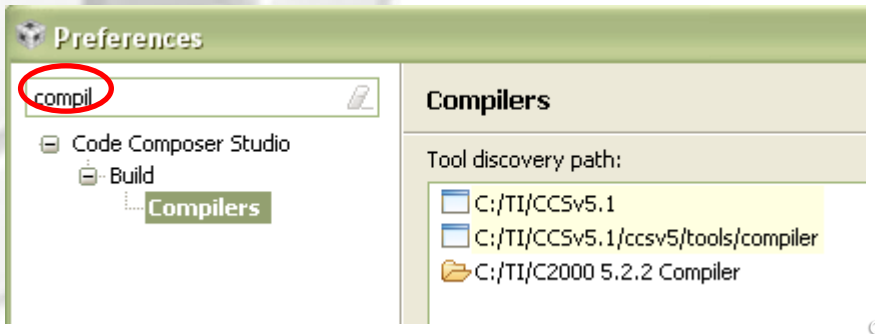


# 过滤字段(Filter Field)

- 更快地找到选项/属性
  - 根据在字段中输入的字符收窄选项列表
- 出现在包含许多选择的对话框中
  - 项目属性(Project Properties)
  - 窗口属性(Window Properties)
  - 目标配置视图
    (Target Configuration View)



# 项目
# PROJECTS

# 共享项目

- 共享“简单”的项目（所有的源/头文件都在项目文件夹中）
- 共享包含“链接文件”(linked file)的项目和所有的源（项目使用链接文件）
- 只共享“链接文件”项目（没有源）
  - 只共享项目文件夹（项目的接收人已经有所有源）
  - 使项目“可移植”(portable)
    - 消除项目中的绝对路径

# 共享项目 - 简单的项目

- **使用案例：希望分享一个项目文件夹和需要构建该项目的所有源文件。所有的源文件都在项目文件夹内。**
- 没有链接文件的项目共享起来很容易：
  - 可以原封不动的发布整个项目的文件夹给各个对象
  - 收到项目的用户可以导入该项目到他们的工作区：使用“项目 - >导入现有的CCE / CCS项目”(Project -> Import Existing CCE/CCS Project)，并选择收到的项目文件夹
  - 适用于简单的项目，所有引用的文件都在项目文件夹内

# 共享项目 - 使用链接文件的项目（1）

- **使用案例：希望分享一个项目文件夹和需要构建项目的所有源文件。一些（或全部）项目的源文件是通过链接所指定的**
- 使用CCS导出项目(Export Project)，以创建一个归档文件（zip）。该文件包含项目的文件夹以及所有项目使用的链接源文件
  - 注意，如果项目中使用链接路径变量(Linked Path variables)的话，这种方法将失败！
- 导出您的项目：如何在源计算机上归档您的项目
  - 文件 - >导出 (File -> Export)
  - 展开“常规(General)”，然后选择“归档文件(Archive File)”。点击“下一步(Next)”
  - 弹出的对话框将允许您选择需要被导出到归档的项目。
  - 当您选择一个项目时，右边图框中会显示的所有会被出口的组件。点上项目旁边的复选框来指示希望出口该项目
  - 指定归档的名称，并选择ZIP或TAR，然后点击“完成”
  - CCS会在您的计算机上生成一个包含该项目的zip/tar文件。它包含了项目目录中的文件和链接/引用的所有资源

# 共享项目 - 使用链接文件的项目（2）

- 在另一台机器上导入项目：如何从归档导入项目-将项目导入到工作区
- 项目 - >导入现有的CCS/ CCE的Eclipse项目
  - Project -> Import Existing CCS/CCE Eclipse Project
  - 更改在顶部的单选按钮选择“选择归档文件(Select archive file)”
  - 浏览并选择您的归档文件
  - 它会列出所有在归档中发现的项目。默认情况下是全选；选择您想要导入的项目
  - 点击“完成”
  - 项目会被导入您目前在的工作区
- 链接/引用的资源将会从归档中被拷贝到相对于原来的计算机上同样的路径

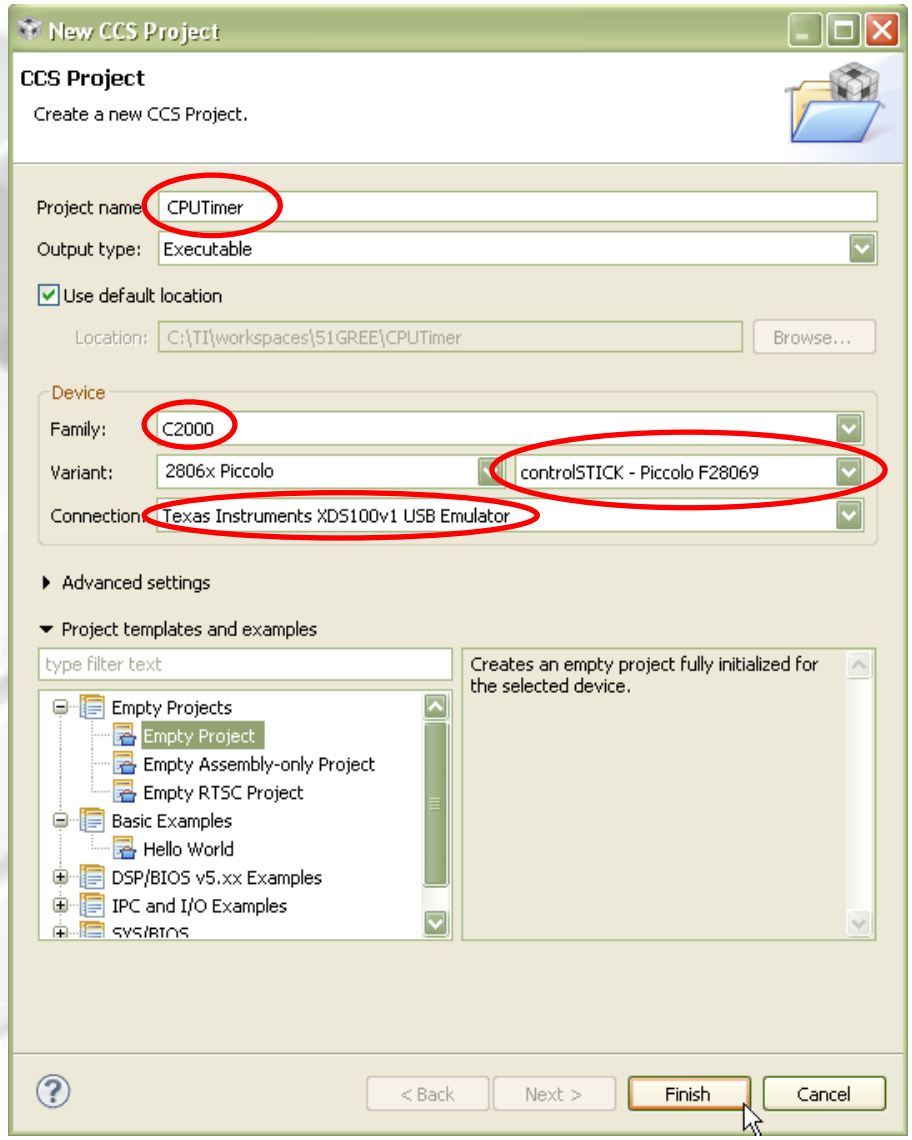
# 共享项目 - 使用链接文件的项目（3）

- 使用个例：
  - 共享一个项目文件夹。项目的接收人已经有了所有源文件的副本
  - 需要把项目文件夹/文件等提交到源代码控制工具
- 大多数个例只是需要共享项目，而不是相关的所有源文件包
  - 用户有源文件的本地副本
- 需要使项目可移植化，以确保该项目能够被轻松共享
- 可移植项目必须避免任何绝对路径
- 理想的可移植项目必须能在不修改任何项目文件的情况下也能使用
  - 这对由在源代码控制工具上维护的项目是理想选择

# 范例：创建一个可移植的链接文件项目
# Example:
# Creating a Portable Linked File Project

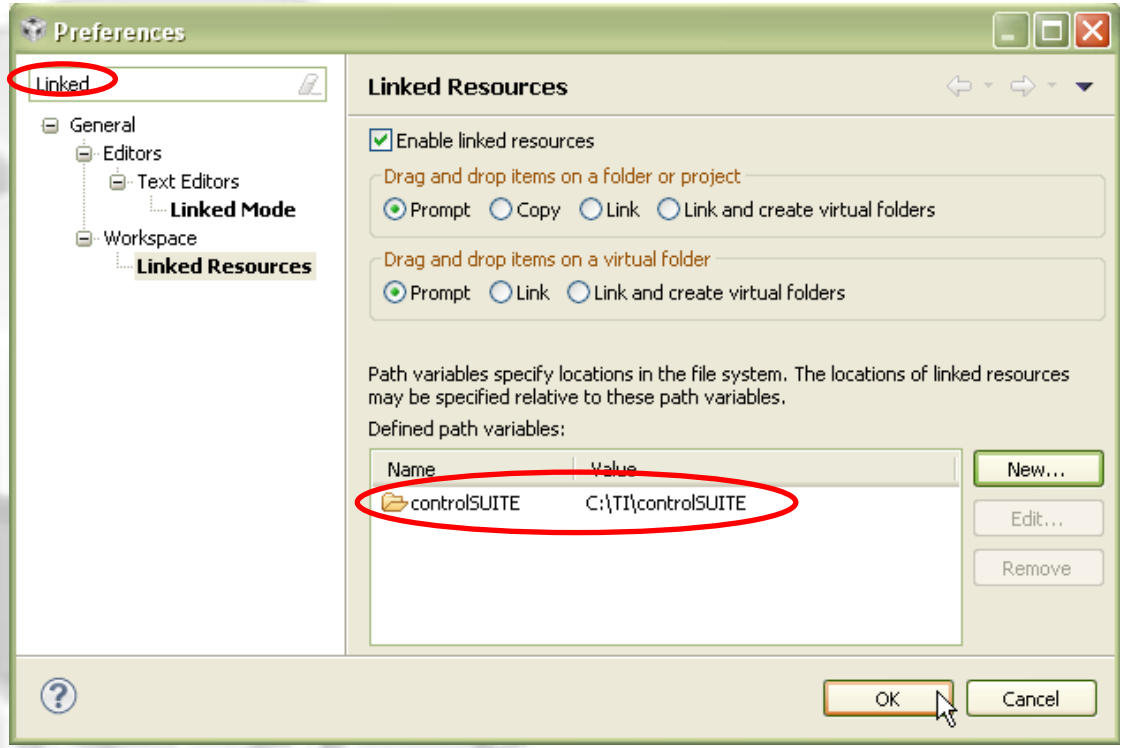
# CPU定时器：创建一个新项目

- 使用“新的CCS项目”(New CCS Project)向导
  - 在“欢迎”(Welcome)页面选择“新建项目”(New Project)
- 如右图所示填写
- 完成后选择“完成”
- 生成的项目将出现在项目资源管理器(Project Explorer)视图
- 从项目中删除生成的“main.c”文件



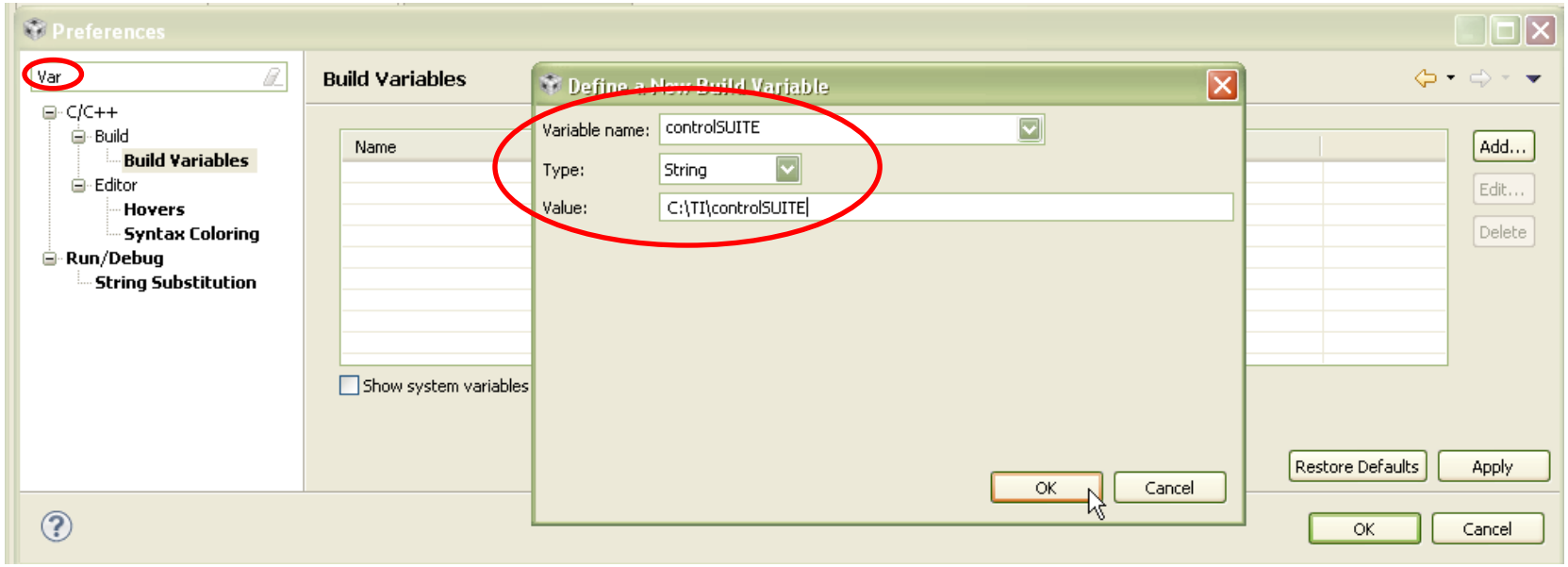
# CPU定时器：创建一个链接资源路径变量

- 打开工作区的首选项(Preferences)
  - 窗口 - >首选项
  - Window -> Preferences
- 打开“链接资源”(Linked Resource)首选项
  - 在过滤器输入”Linked”，以便查找
- 使用“新建”(New)按钮来创建一个“链接资源的变量”(Linked Resource Variable)，指向controlSUITE根目录的位置
- 完成时，点击“OK”



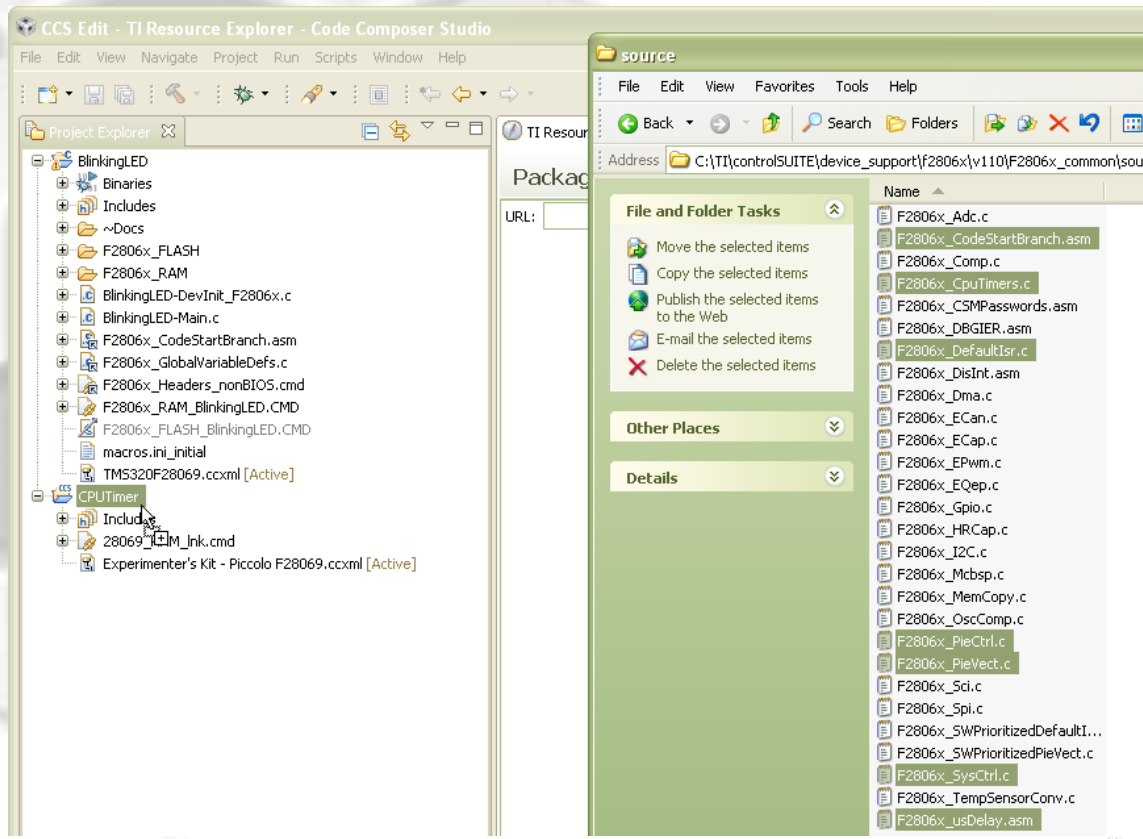
# CPU定时器：创建一个构建变量

- 转到“构建变量”(Build Variables)首选项
  - 在过滤领域输入”Variables”，方便查找
- 构建变量允许您在项目属性中使用变量
  - 链接资源的路径变量仅适用于链接文件
- 使用“添加”按钮来创建一个“构建变量”，指向**controlSUITE**根目录的位置
- 完成时，点击“OK”



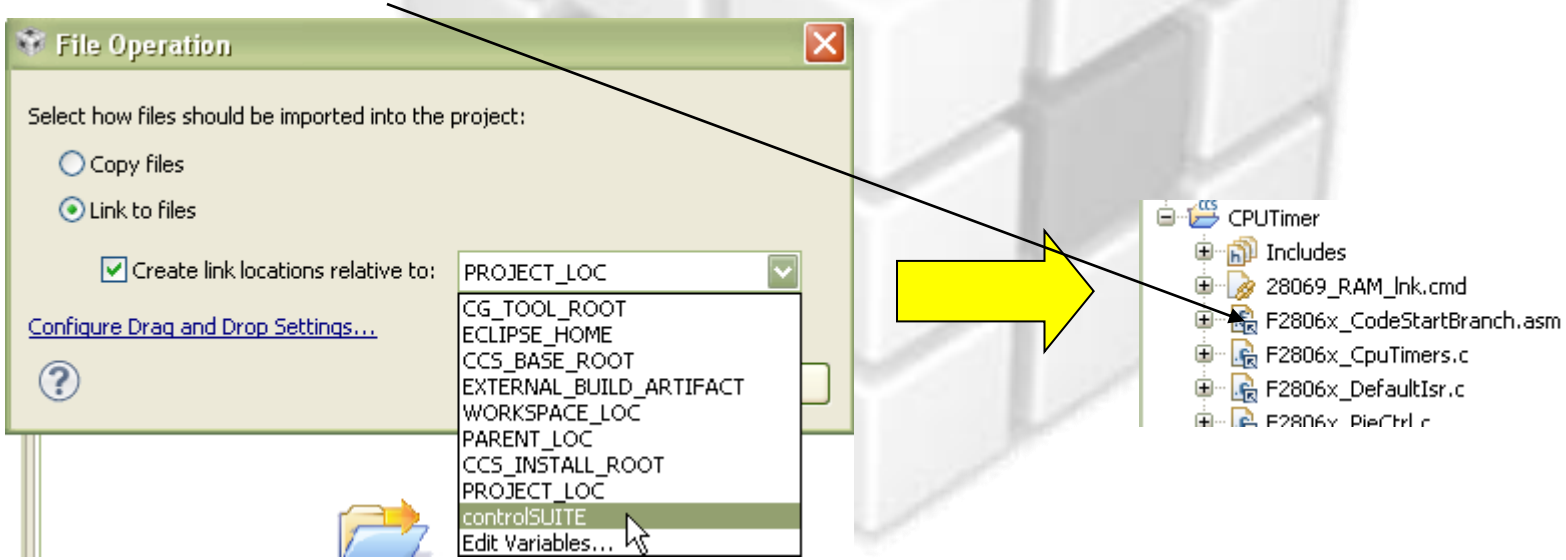
# CPU定时器 :把源文件链接到项目

- 打开Windows Explorer并浏览到：
  - C:\TI\controlSUITE\device_support\f2806x\v110\F2806x_common\source
    - 把以下源文件拖放到CCS的Project Explorer视图中的新项目
      - F2806x_CodeStartBranch.asm
      - F2806x_CpuTimers.c
      - F2806x_DefaultIsr.c
      - F2806x_PieCtrl.c
      - F2806x_PieVect.c
      - F2806x_SysCtrl.c
      - F2806x_usDelay.asm



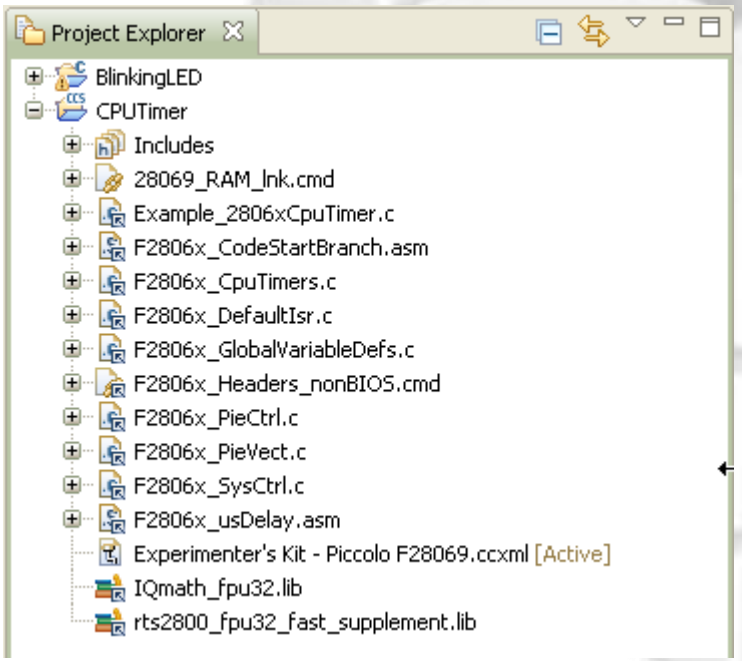
# CPU定时器 :把源文件链接到项目

- 一个对话框将出现，确认您是否想复制或链接文件到项目
  - 选择“链接到文件(Link to files)”
  - 选择“创建链接的相对位置(Create link locations relative to)”
    - 使用我们新创建的链接资源的变量（controlSUITE）
  - 点击“确定”
- 有着“链接”(Link)图标的文件将出现在Project Explorer中



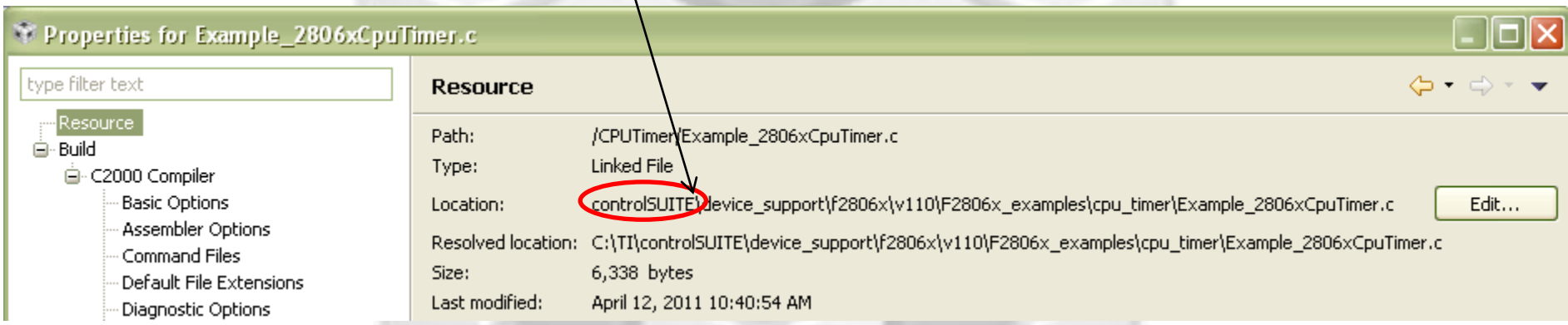
# CPU定时器 :把源文件链接到项目

- 用相同的方式链接其余所需的文件：
  - C:\TI\controlSUITE\device_support\f2806x\v110\F2806x_examples\cpu_timer\Example_2806xCpuTimer.c
  - C:\TI\controlSUITE\device_support\f2806x\v110\F2806x_headers\source\F2806x_GlobalVariableDefs.c
  - C:\TI\controlSUITE\device_support\f2806x\v110\F2806x_headers\cmd\F2806x_Headers_nonBIOS.cmd
  - C:\TI\controlSUITE\libs\math\IQmath\v15c\lib\rts2800_fpu32_fast_supplement.lib
  - C:\TI\controlSUITE\libs\math\FPUfastRTS\V100\lib\IQmath_fpu32.lib



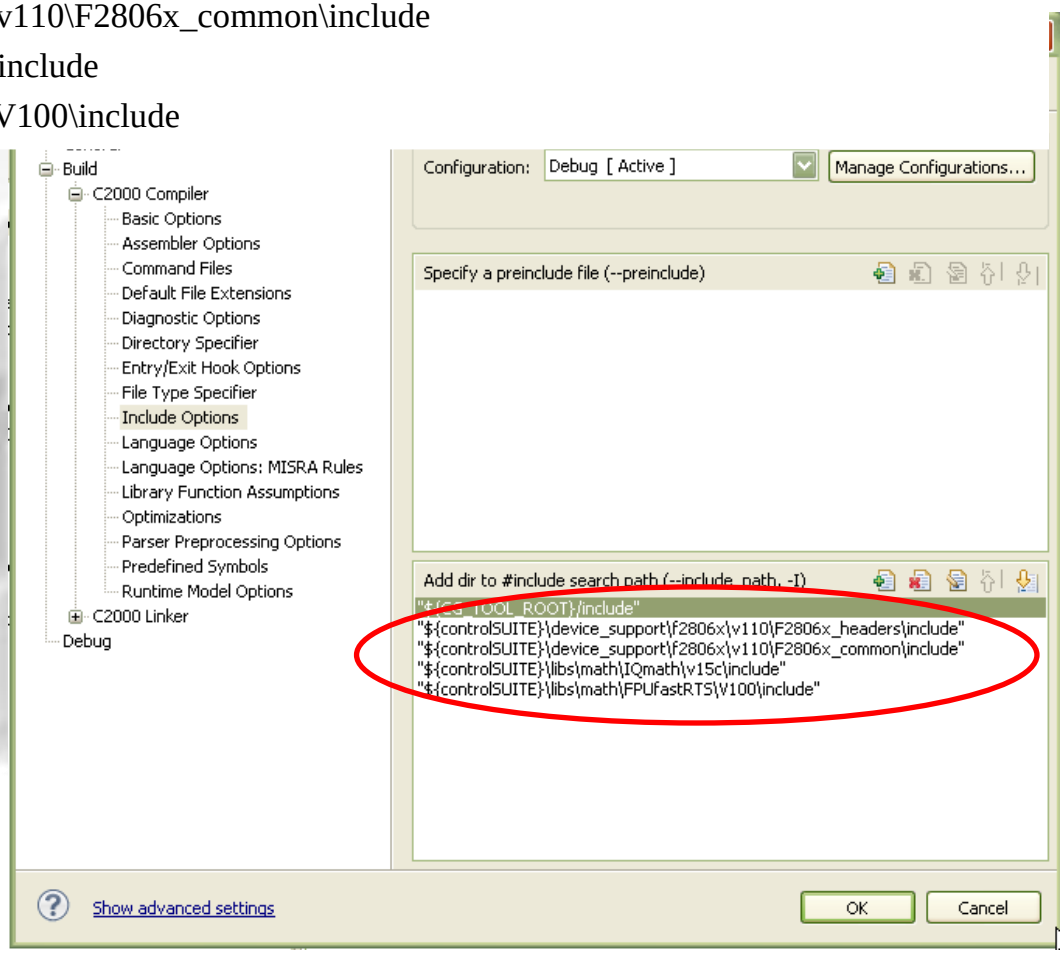
# CPU定时器 :把源文件链接到项目

- 右键单击源文件，并检查“属性”(Properties)
  - 注意 “位置”(Location)参数如何引用链接资源的变量



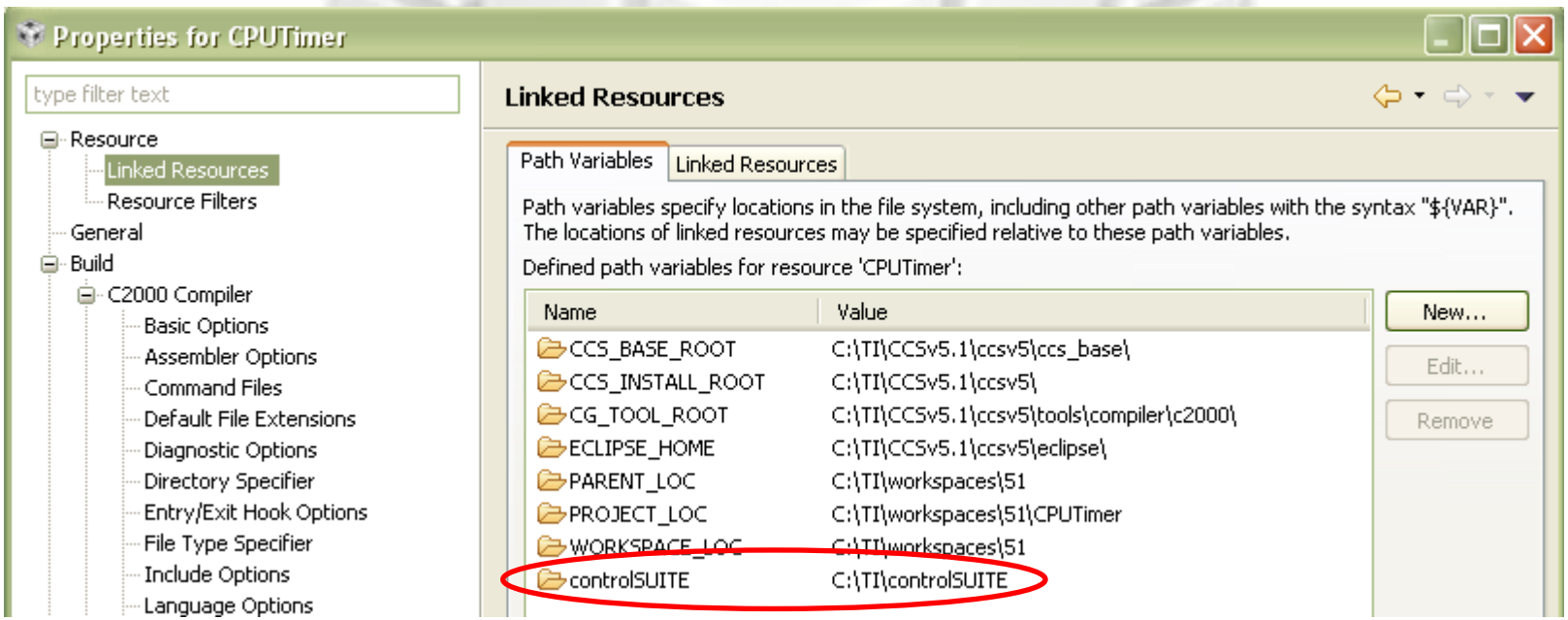
# CPU定时器：修改项目属性

- 右键单击该项目并选择“属性”(Properties)
- 在编译器“包含选项(Include Options)”中，把以下路径添加到搜索列表中：：
  - ${controlSUITE}\device_support\f2806x\v110\F2806x_headers\include
  - ${controlSUITE}\device_support\f2806x\v110\F2806x_common\include
  - ${controlSUITE}\libs\math\IQmath\v15c\include
  - ${controlSUITE}\libs\math\FPUfastRTS\V100\include
- **‘${<BUILD VARIABLE>}’**是在项目属性中使用构建变量的语法



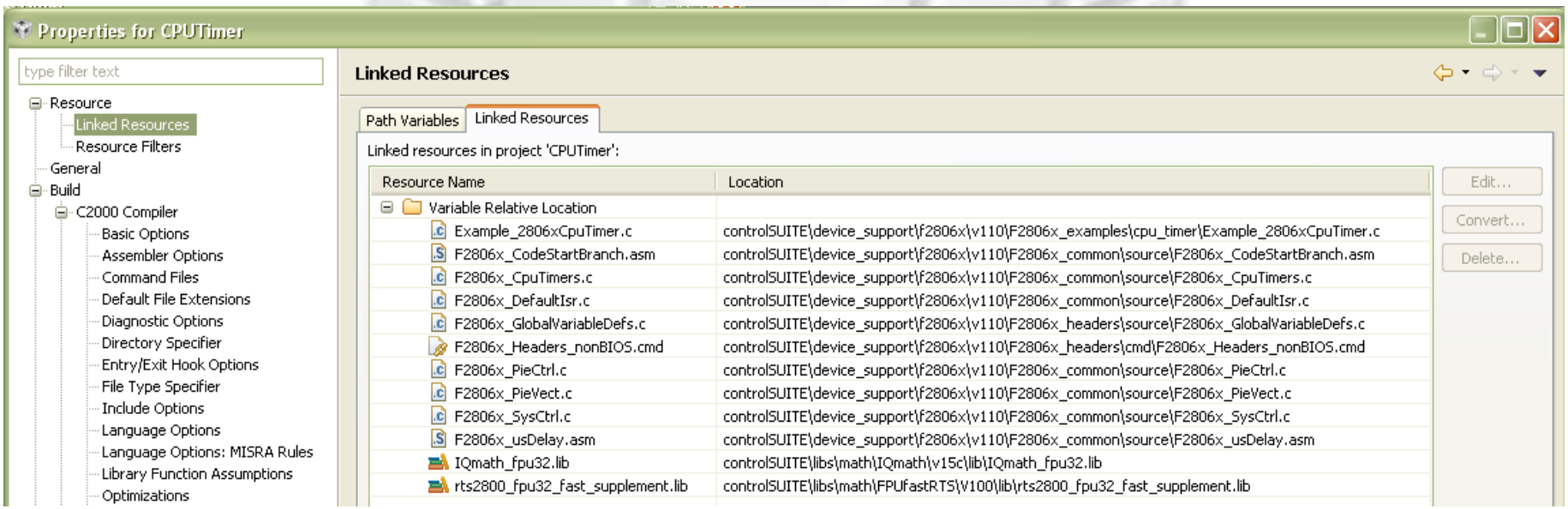
# CPU定时器：项目属性

- 打开“资源 - >链接的资源(Resource -> Linked Resource)”,显示项目所有链接资源的路径变量
  - 这将显示所有在项目层面和工作区层面上所创建的变量
- 工作区层面上的链接资源的路径变量会出现在列表中
- 在这里可以编辑变量，但改变只会发生在项目层面（存储在项目文件中）



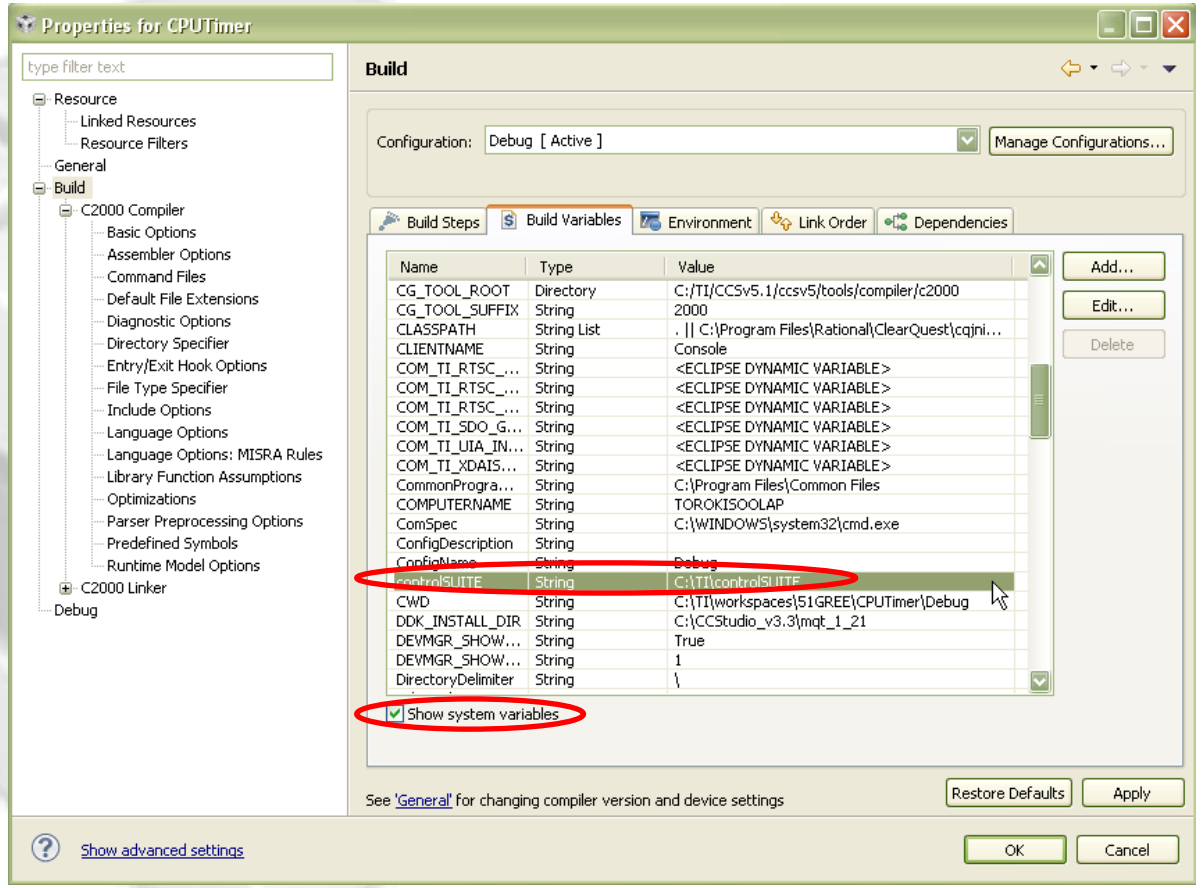
# CPU定时器：项目属性

- “链接的资源”(Linked Resources)标签显示所有已链接到项目的文件
  - 链接文件会被分类排序成使用链接变量的资源和使用绝对路径的资源
- 可以使用“编辑(Edit)...”按钮修改链接
- 链接可以通过“转换(Convert) ...”按钮，转换为使用绝对路径



# CPU定时器：项目属性

- 展开“构建” (Build)来查看项目中所有的构建变量
  - 默认情况下，只有项目级别的变量会被列出
  - 启用“显示系统级变量 (Show system variables)”复选框来查看在工作区和系统级设置的变量
- 现在工作区级创建的构建变量出现在列表

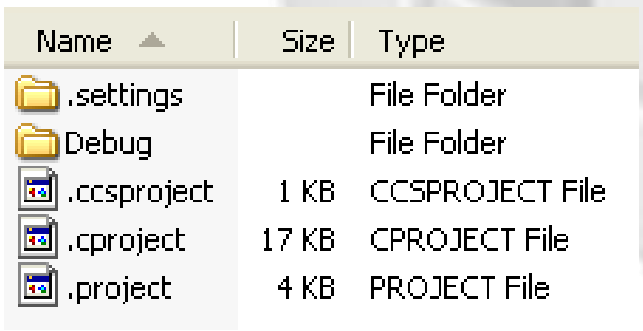


# 项目与工作区级的变量

- 在刚才的几页里，我们展示了如何在项目级别设置“链接资源的路径变量”(Linked Resource Path Variables)和“构建变量”(Build Variables)
- 在工作区级别设置这些变量比在项目级别设置的好处是什么？
  - 所有项目都可以重复使用相同的变量（设置一次）
  - 不需要修改项目！
    - 这对源控制工具管理的项目非常重要

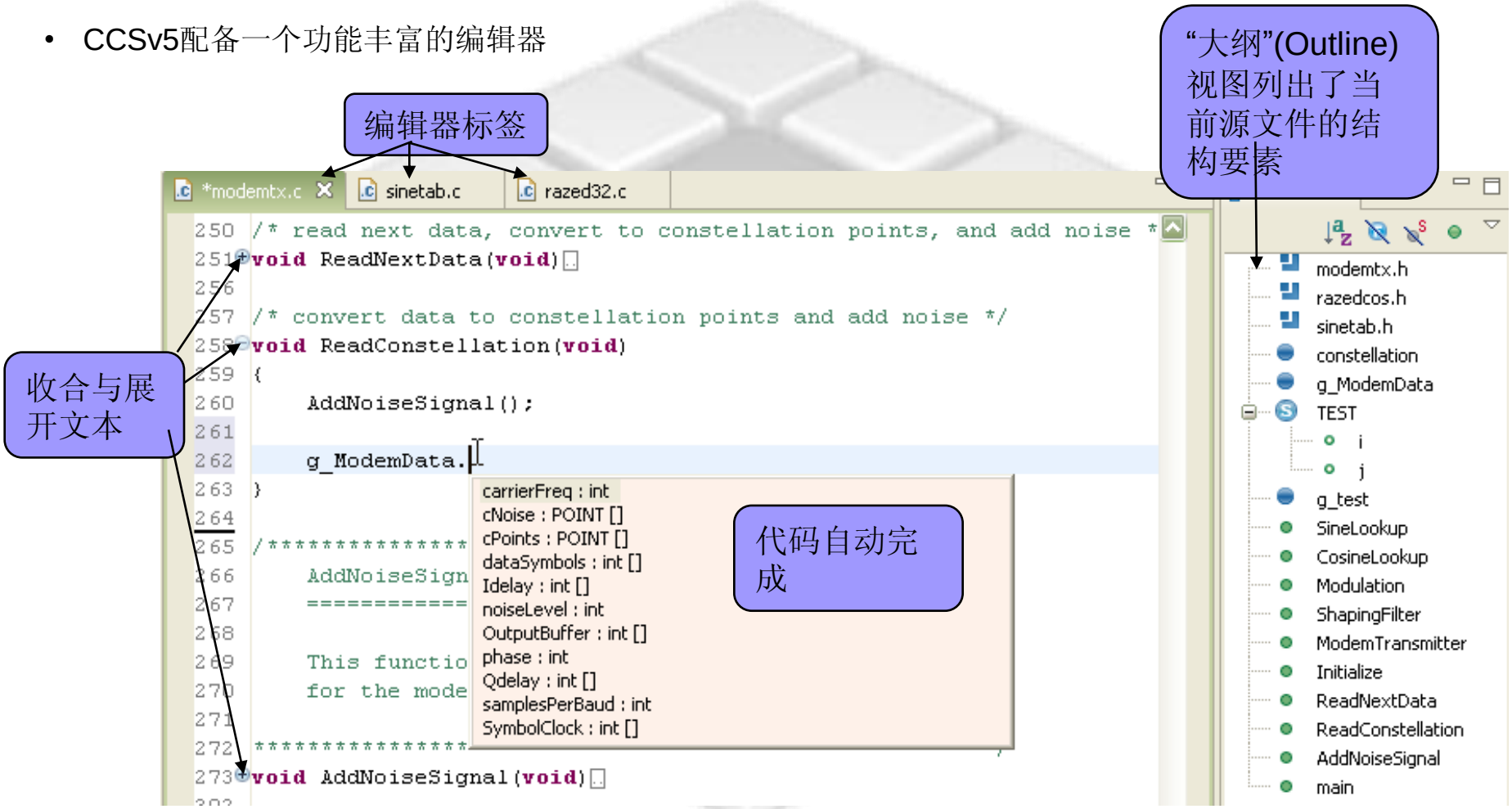
# 版本控制 - 提交(Check In)哪些文件？

- CCS项目文件夹内生成多个文件/文件夹
  - .CPROJECT 和.PROJECT是**Eclipse**项目文件，并应提交
  - .CCSPROJECT是CCS的具体项目文件，应提交。
  - .settings是一个标准的Eclipse文件夹，其中包含适用于该项目的设置。应提交此文件夹
  - .launches是在启动调试会话启动时生成的目录。它不涉及您的项目的构建，没有必要提交
  - 项目配置文件夹（调试/发布） (Debug/Release)不需要进行提交



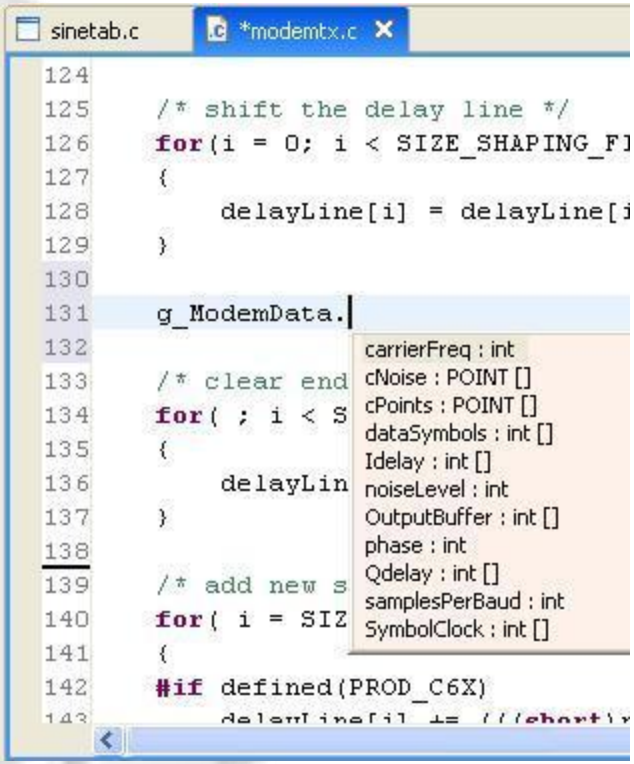
# 源代码编辑器（ Source Code Editor）

- CCSv5配备一个功能丰富的编辑器



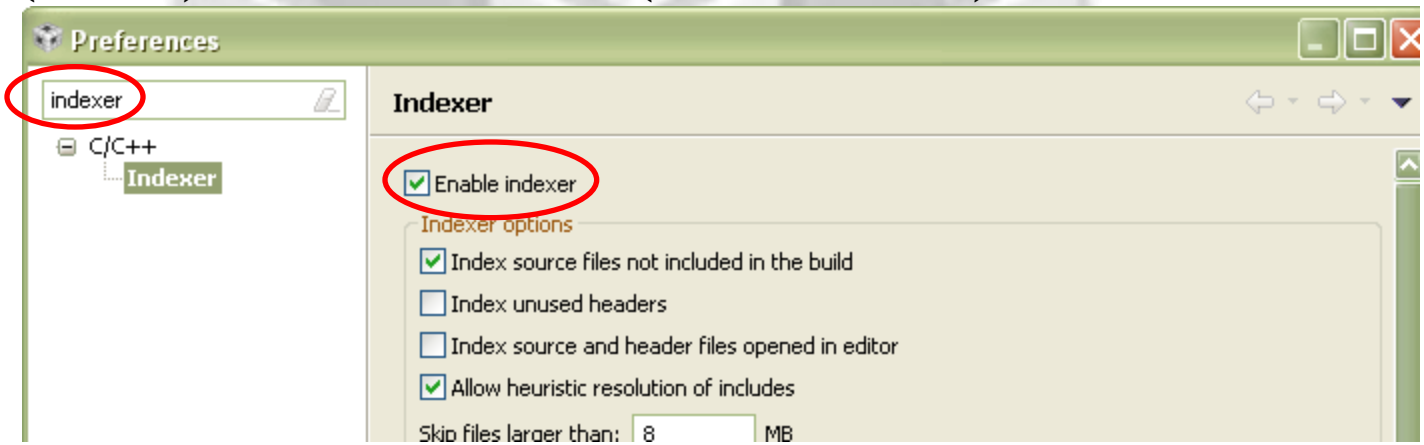
# 高级编辑功能(Advanced Editor Features)

- 代码完成(Code Completion)
  - 完成单词
  - 自动成员(member)信息
  - 自动参数(parameter)信息
  - ... ...
- 导航
  - 后退/前进按钮
  - 返回最后编辑部位的按钮
  - 转到定义(definition)
  - 转到声明(declaration)
- 显示行号
- 代码折叠
  - 收合功能



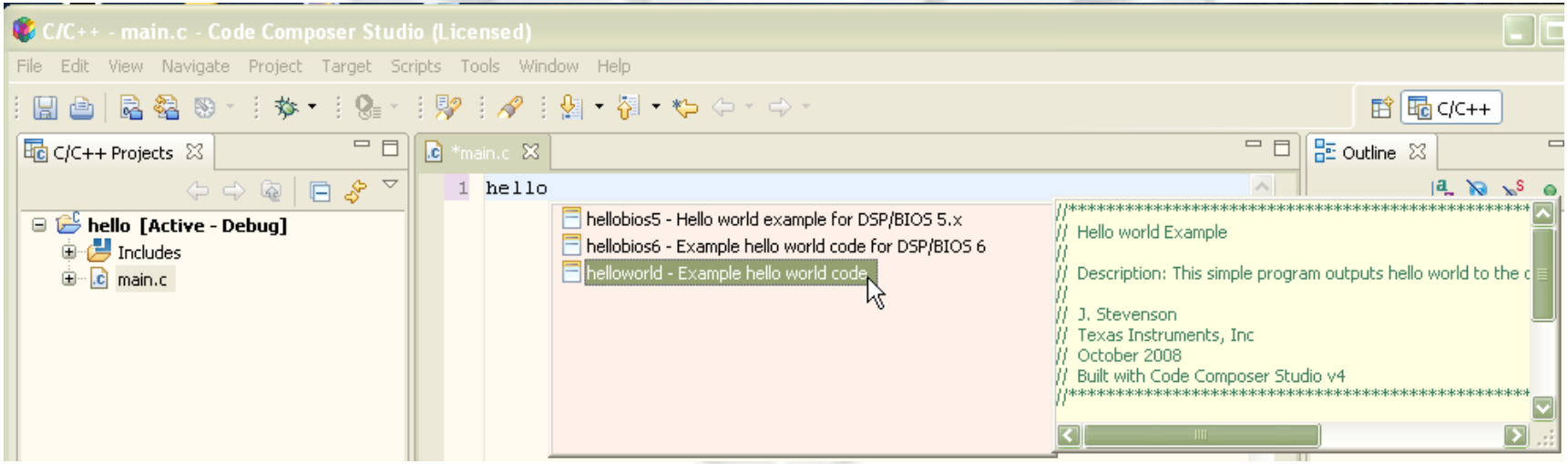

# 性能提示：索引器(Indexer)

- 如果您不需要使用CCS编辑器，或不关心高级编辑功能：
  - 关闭索引器
    - 索引器为了支持一些高级编辑功能，会不断地扫描所有打开的项目
    - 索引器使用可观的系统资源，对CCS操作响应有负面影响
      - 当工作区中有特别大型的项目或许多打开的项目时，这个问题比较明显（或两者）
    - CCS的设置默认是启用索引器
  - 调整索引器设置
    - 打开“窗口” - >“首选项”(Window -> Preferences)，在搜索字段输入“索引器”(Indexer)。禁用“启用索引器”(Enable indexer)复选框来关闭索引器



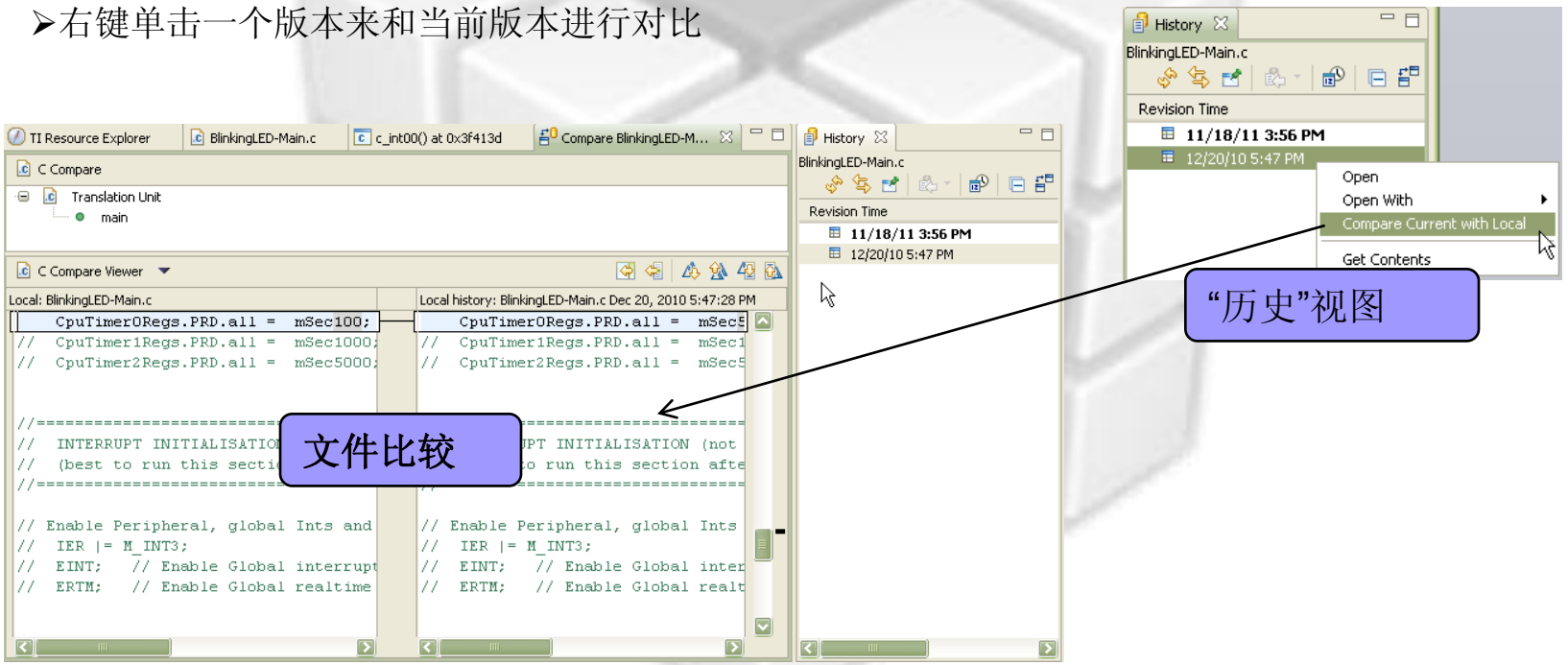
# 源模板(Source Templates)

- CCS提供源模板
  - 例如：“Hello World”
    - 在编辑器中输入‘h’字符, 按“CTRL+空格键”使用 “内容辅助”(Content Assist)功能， （也可以右键单击编辑器，从上下文菜单中选择”内容辅助“ ）
      - 选择”HelloWorld”
    - 创建自定义模板
      - 窗口>首选项... - > C + + - >编辑 - >模板
      - Window->Preferences…->C++->Editor->Templates



# 视图：本地历史（Local History）

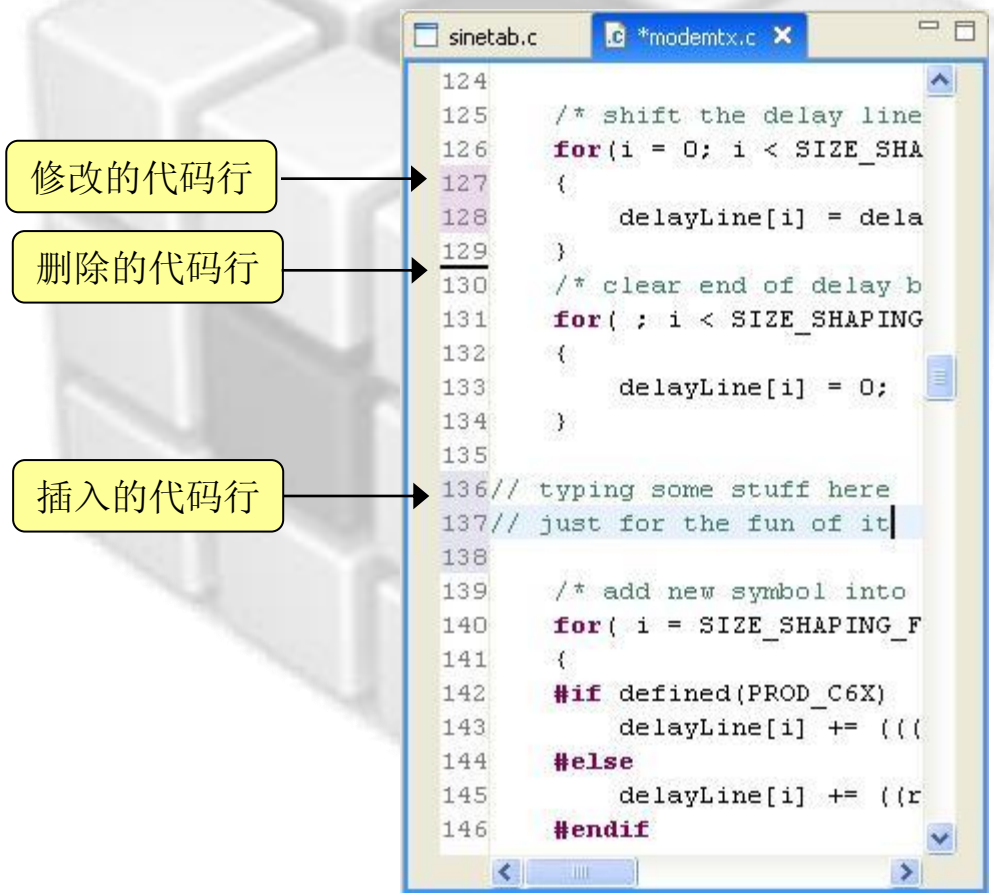
- CCS保留了本地源的变化历史
  - 切换到的CCS编辑(CCS Edit)的透视图
  - 在资源管理器中右键单击一个文件，选择“团队 - >显示本地的历史”(Team -> Show Local History)
- 您可以把当前的源文件和任何以前的版本作比较,或回滚到以前的版本
  - 双击一个版本将其在编辑器中打开
  - 右键单击一个版本来和当前版本进行对比




- CCS还保留项目历史
  - 恢复从项目中删除的文件
  - 右键单击该项目，并在菜单中选择“从本地历史恢复”(Recover from Local History)

# 编辑标记

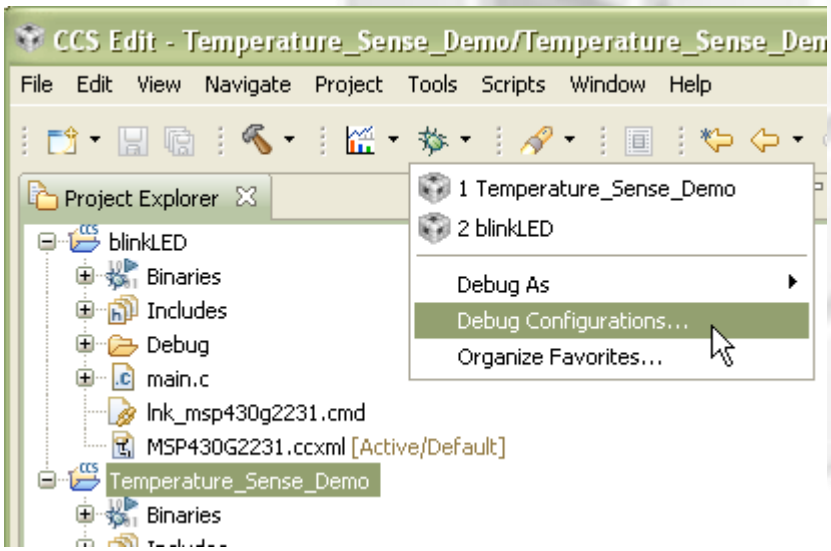
- 如果你打开行号显示，编辑器会在行号列上用不同的颜色标明自从上次保存后源文件中的变化



# 调试
# DEBUGGING

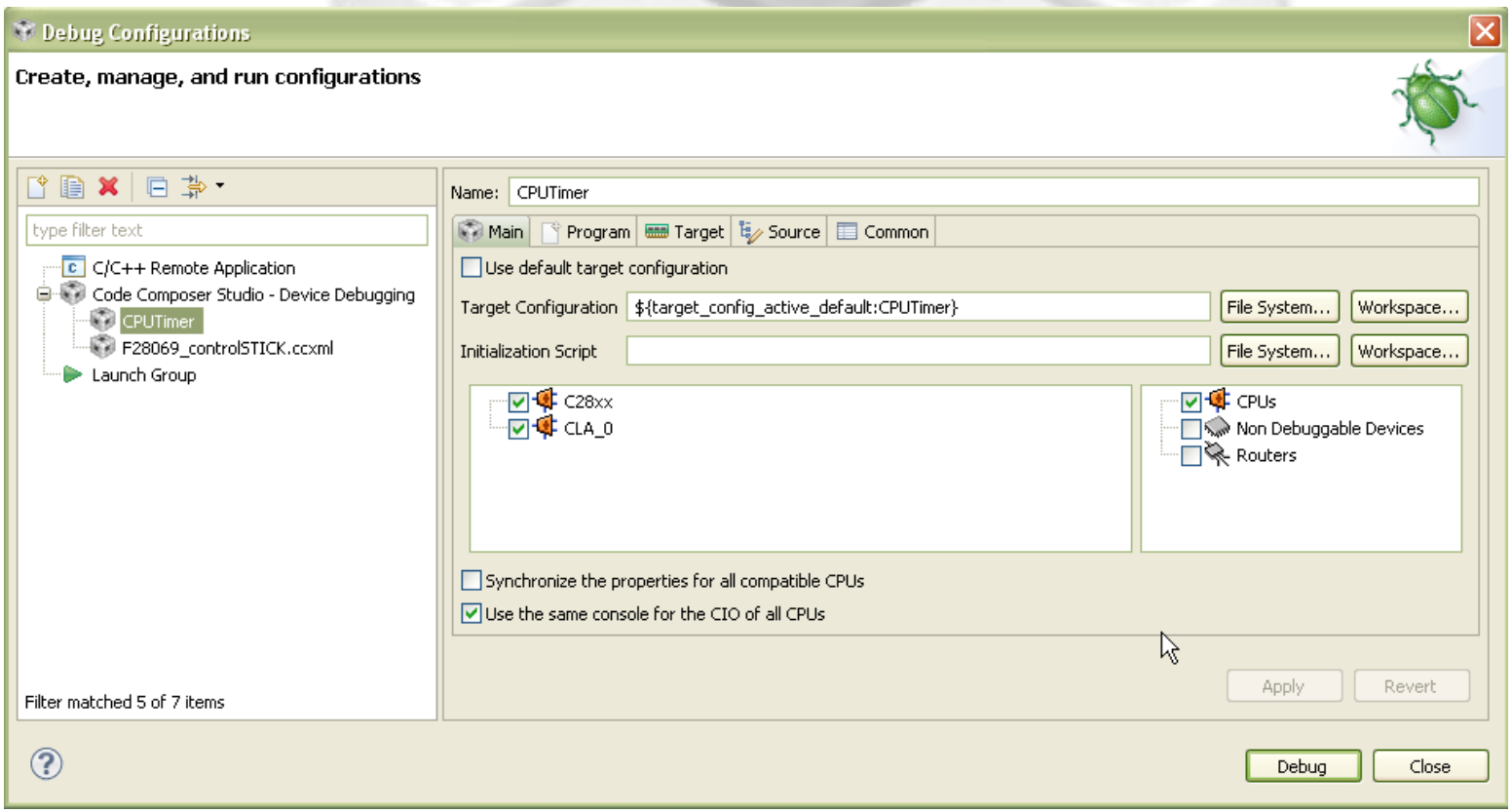
# Eclipse概念：调试配置

- Eclipse概念：调试配置- 当一个项目或目标配置第一次启动调试会话后所创建的缓存信息。
  - 缓存的内容包括因该使用的目标配置，调试设置...
- 从最近使用的配置列表中，选择一个现有的配置，启动调试会话
- 如何配置：使用 “调试”(Debug)按钮旁边的下拉菜单，并选择“调试配置..”(Debug Configurations)



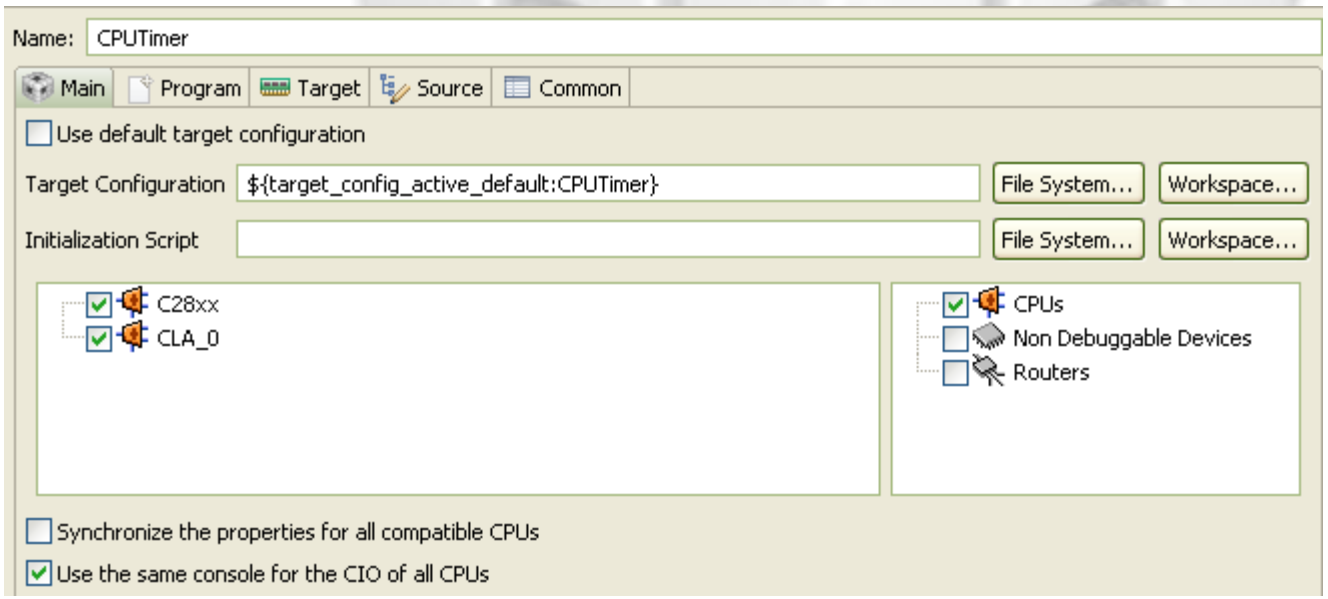
# 调试配置（Debug Configurations）

- 用来管理现有的调试配置，或建立新的调试配置的界面



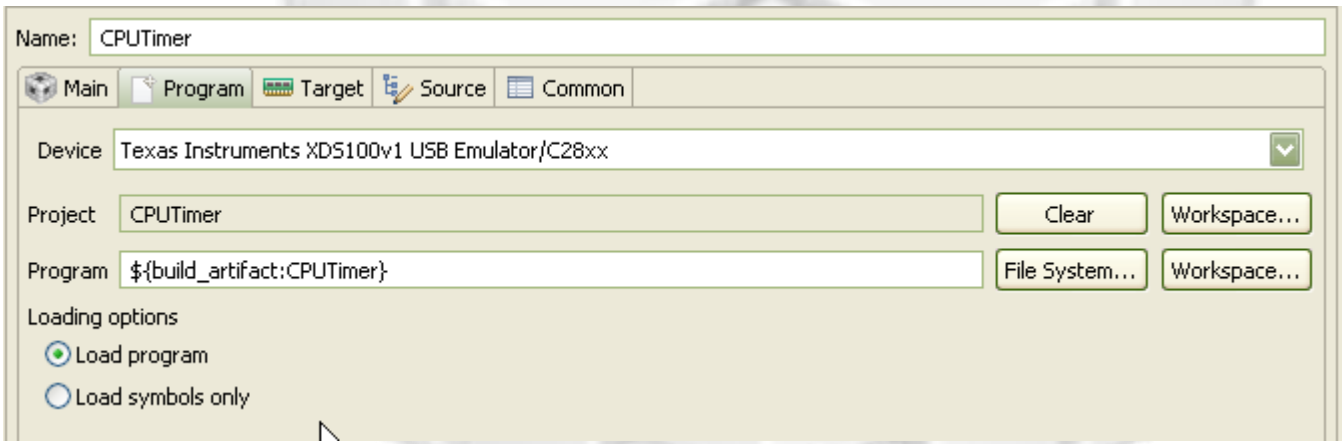
# 调试配置-主选项（Main）

- 使用“主”(Main)标签：
  - 指定使用的目标配置(Target Configuration)
  - 使用“初始化脚本”(Initialization Script)字段，指定一个用于目标初始化的DSS JavaScript
  - 指定将在“调试”(Debug)视图中显示的JTAG扫描链上的设备
  - 指定是否所有的CPU共享相同CI / O控制台（多内核设备）
    - CI / O将在同一控制台交错（由CPU的名称打头）
    - 取消选中“Use the same console for the CIO of all CPUs”选项，将导致每个CPU创建一个单独的CI/ O控制台



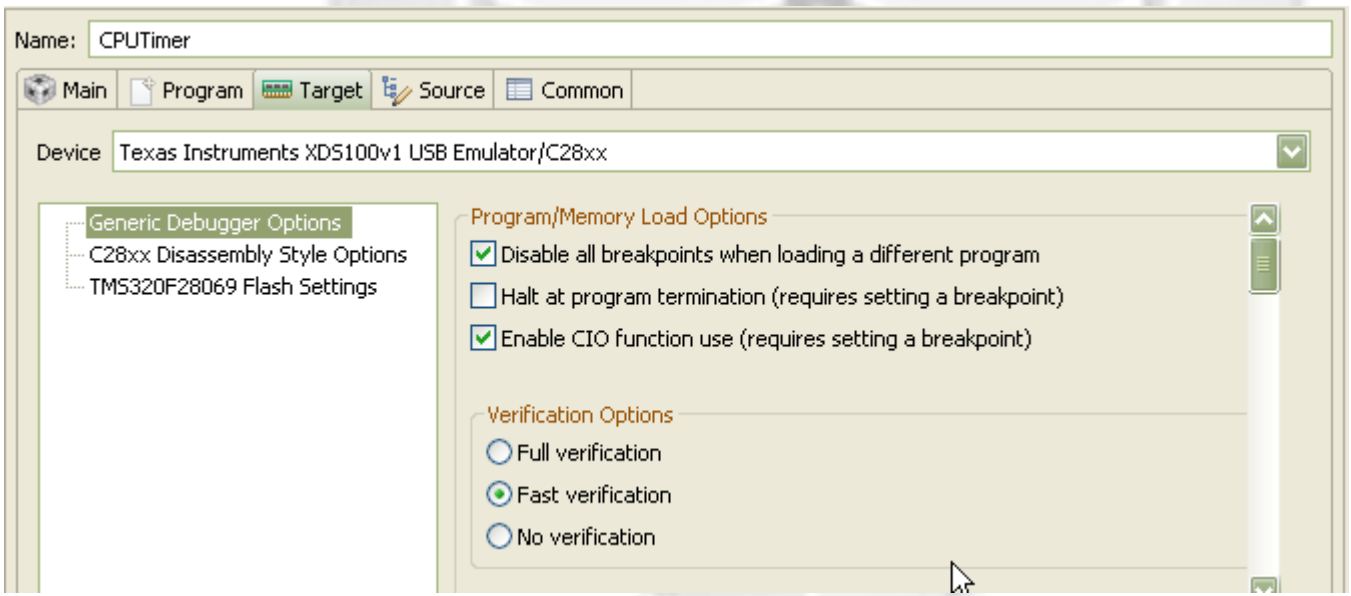
# 调试配置 - 编程选项（Program Options）

- 使用“Program”标签：
  - 可以指定什么CPU加载什么可执行文件（多核设备）
  - 指定加载程序（默认）或只是加载符号（调试闪存上的代码等）



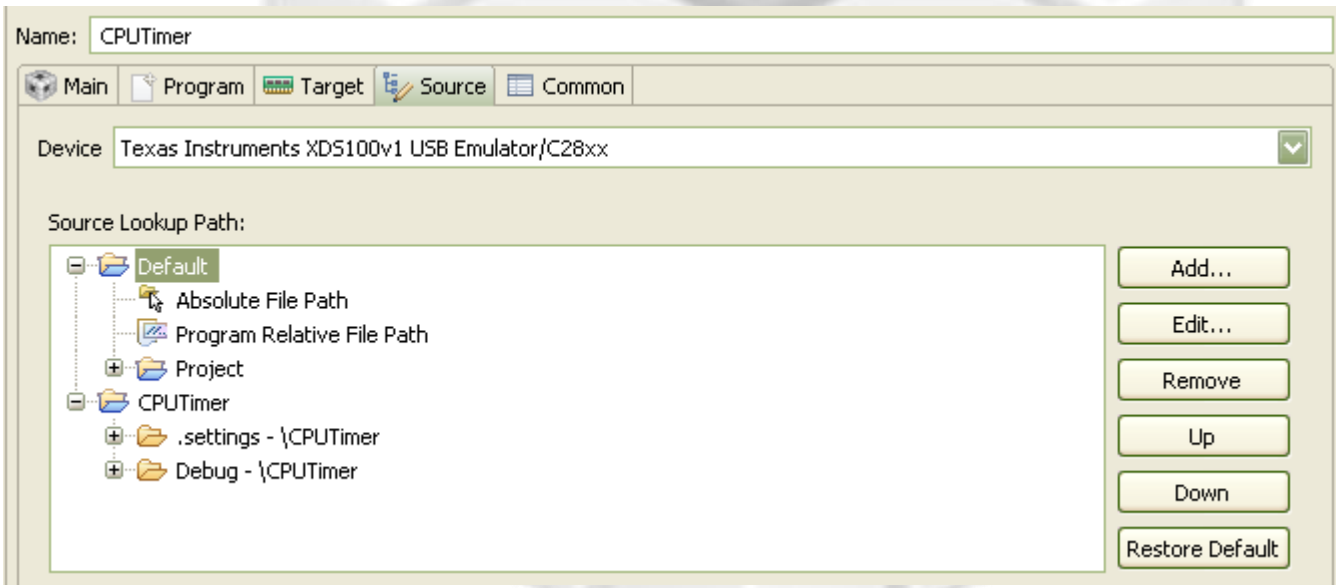
# 调试配置 - 目标选项（Target Options）

- “目标”(Target)选项卡可用于设置各种调试选项，例如自动运行到主函数，自动连接到硬件目标，实时选项，加载时验证程序，等等... ...
- 适用的设备可以选择“闪存编程器”Flash Programmer选项



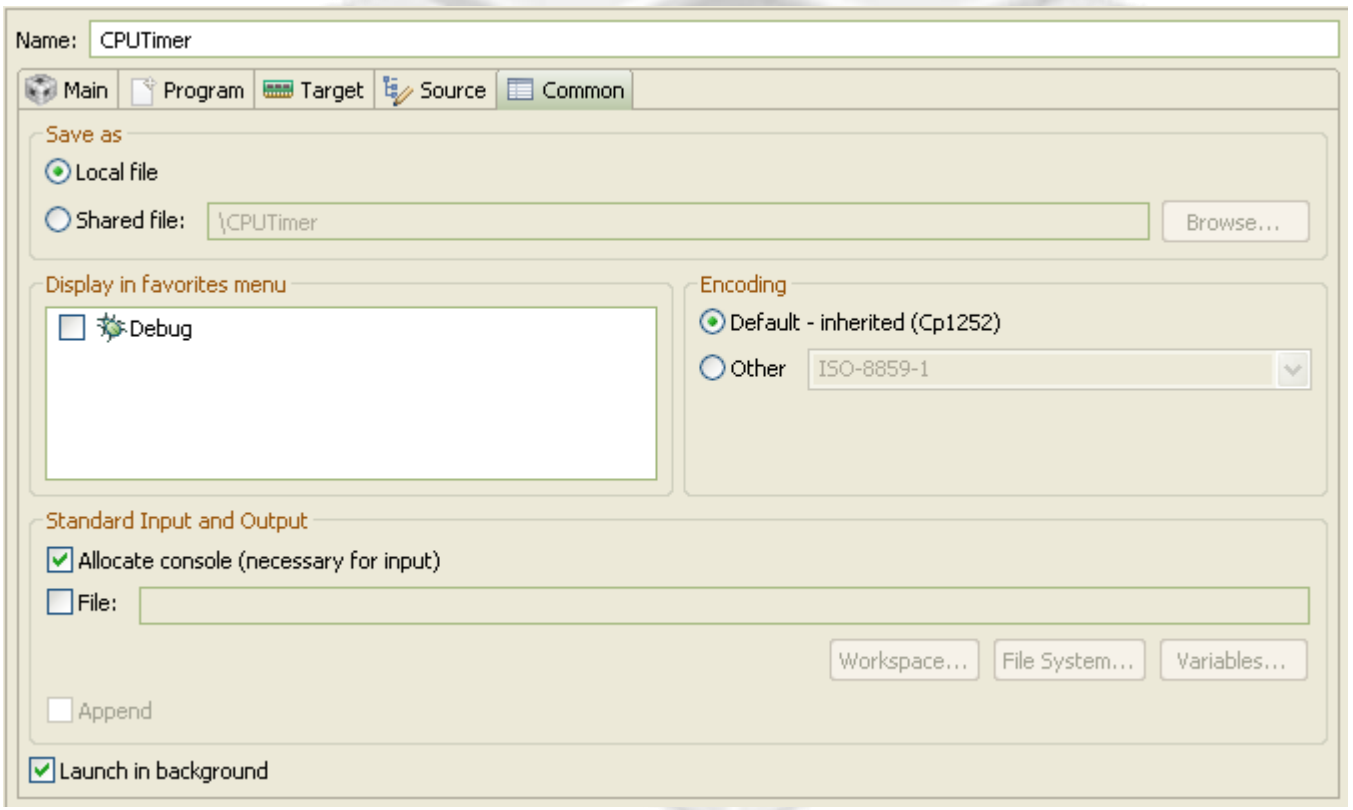
# 调试配置 - 源选项（Source Options）

- “源”（source）选项卡允许您添加其他查找源文件时使用的搜索路径
  - 默认情况下, CCS会自动添加项目中所有源文件的路径



# 调试配置 - 共同选项(Common Options)

- “共同”（Common）标签包含了一些杂选项
  - 例如：可以指定调试器把所有的CIO发送到一个文件，而不是控制台

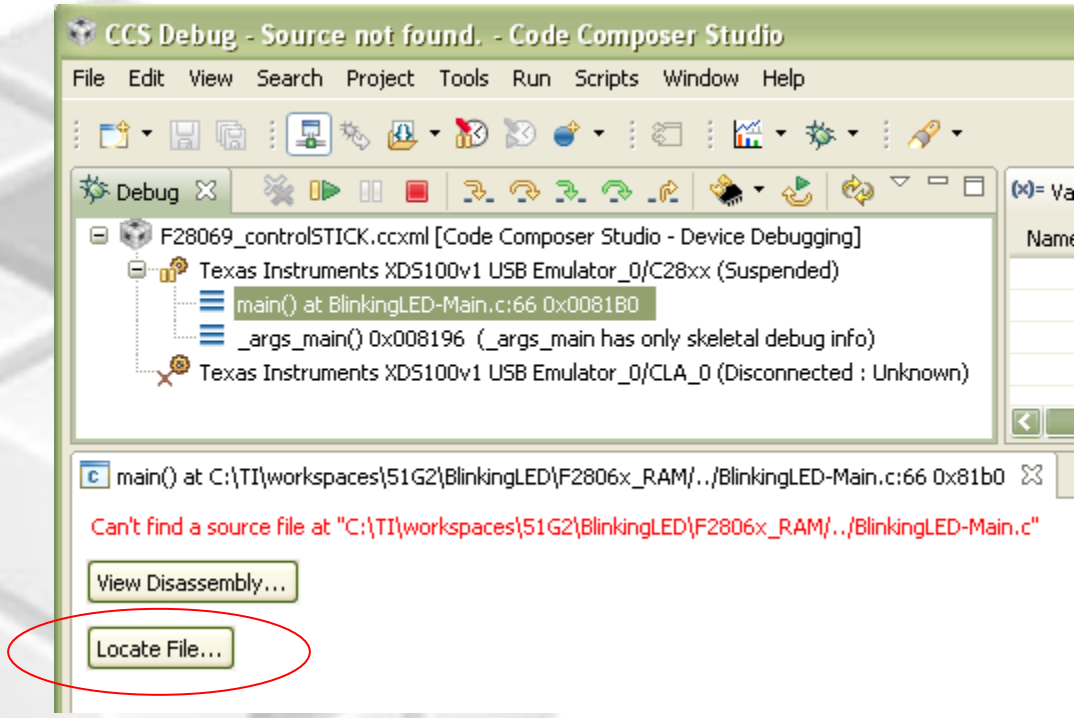


# 没有项目的调试

- 因为没有项目，CCS不知道调试中的程序的源文件的位置
  - 在CCS没有打开项目的情况下，用户仍然可以加载程序*. out
- 用户指示CCS在哪里可以找到源文件的两种方法：
  - 告诉CCS第一个文件是在什么地方，让CCS使用符号的相对路径信息，找到其余的文件（推荐）
  - 设置“源查找路径”(Source Lookup Paths)，让CCS在寻找源文件时扫描该路径：
    - 为当前调试会话设置
    - 为调试配置设置 - 适用于该调试配置启动的每一个调试会话
    - 在全局（工作区）级设置 - 适用于任何在此工作区启动的调试会话

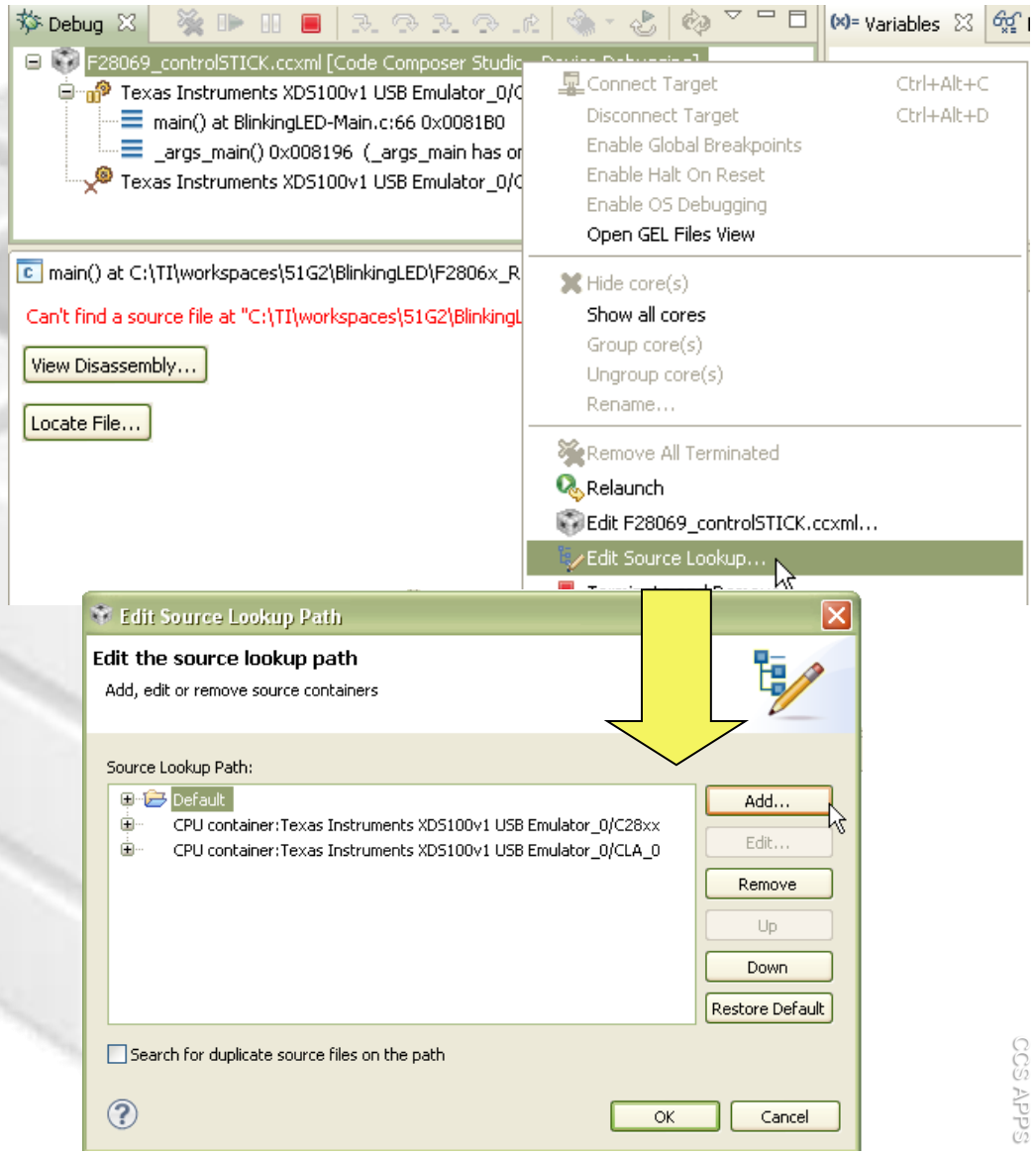
# 源查找路径：方法1（推荐）

- 如果在调试过程中CCS不能发现源文件，编辑器会显示红字提示
- 使用“查找源文件...”(Locate the Source File)按钮, 浏览到源文件的位置
  - 调试器可以找到在同一位置的其他源文件
  - 或使用相对路径信息找到其他文件
  - CCS会记录以上文件的位置，方便未来该程序的调试
- 这是查找源文件最有效的的操作方法



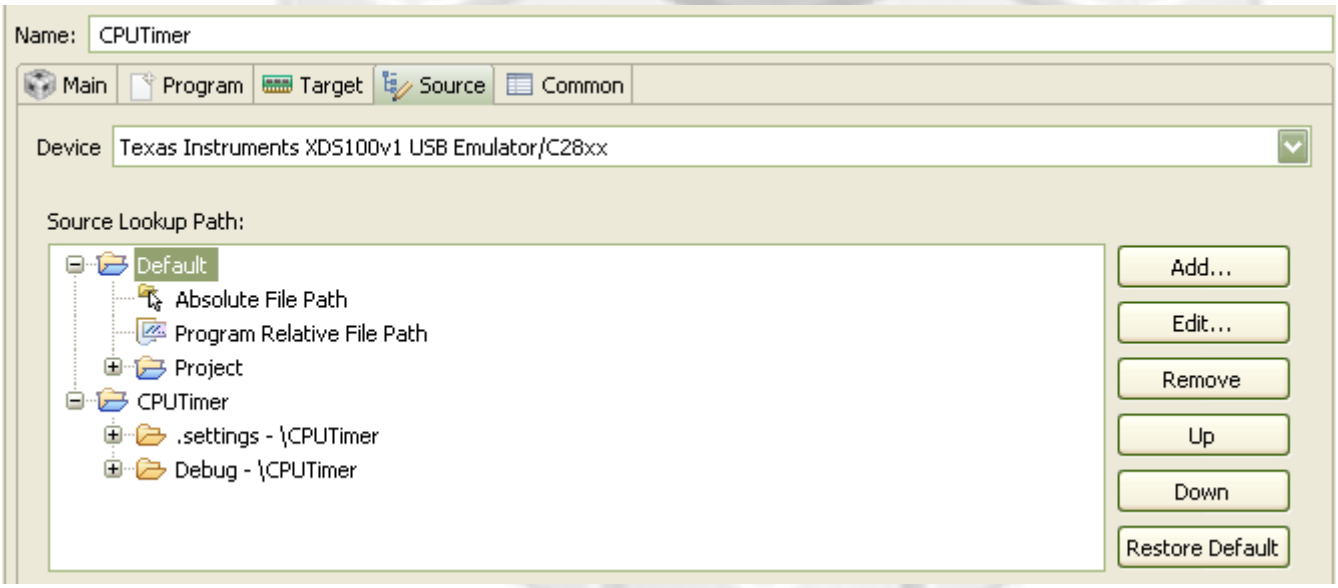
# 源查找路径：方法#2 - 调试上下文

- 用户也可以在每个调试上下文中，明确指定适用的源查找路径
- 右键单击“调试”视图，并在上下文菜单中选择“编辑源查找…”(Edit Source Lookup)
- 如果想要添加一个文件系统路径，选择“文件系统目录”(File System Directory), 浏览到并添加该路径
- 对于多核调试，每个调试上下文可以拥有自己的一套源查找路径
- 警告：CCS将在指定的目录内进行递归式文件搜索。如果目录结构复杂，搜索可能会比较缓慢。这种情况下推荐考虑使用方法#1
  - 此警告适用于方法#3和#4



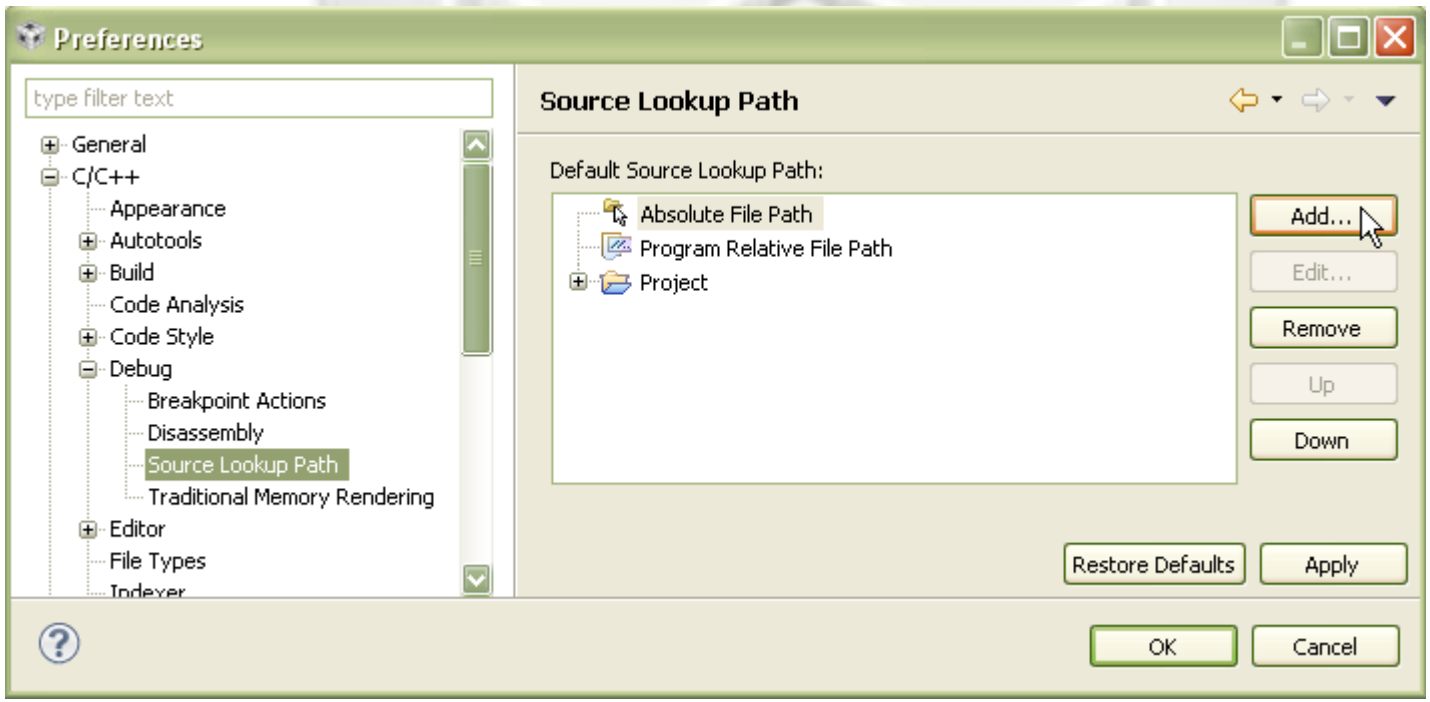
# 源查找路径：方法#3 - 调试配置

- 在调试配置(debug configuration)中的“源”(Source)选项卡允许你添加额外的源查找与搜索路径
  - 默认情况下会自动添加项目中任何源文件的路径



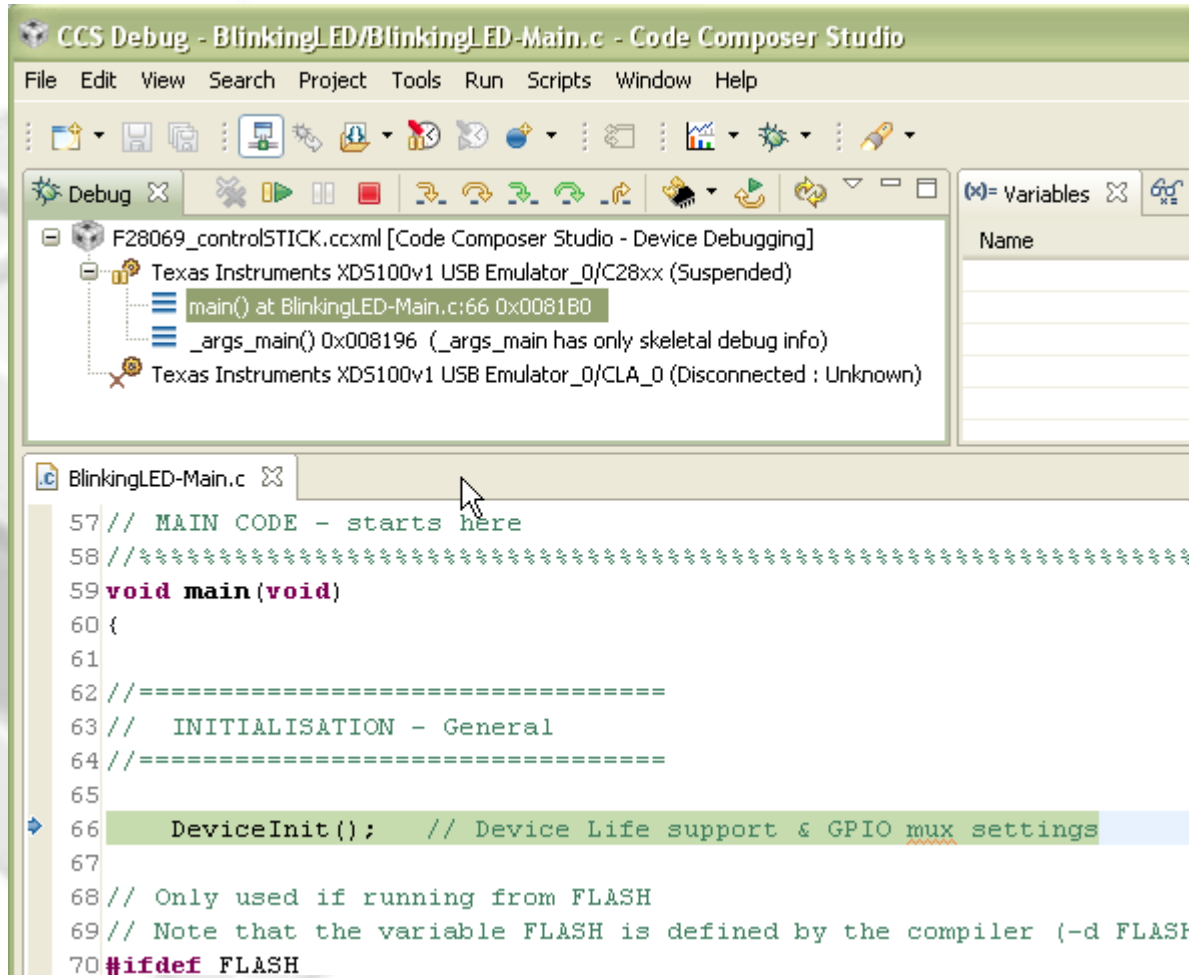
# 源查找路径：方法#4 - 全局（工作区）

- 用户也可以在全局级别设置源查找路径，这些路径适用于所有的调试上下文（多核环境）和调试会话
  - 窗口 - >首选项 - > C/ C+ + - >调试 - >常用的源查找路径
  - Windows -> Preferences -> C/C++ -> Debug -> Common Source Lookup Path



# 更多的调试：源查找路径

- 一旦路径变成调试器已知的路径（任何方法），CCS将在调试时自动打开源文件

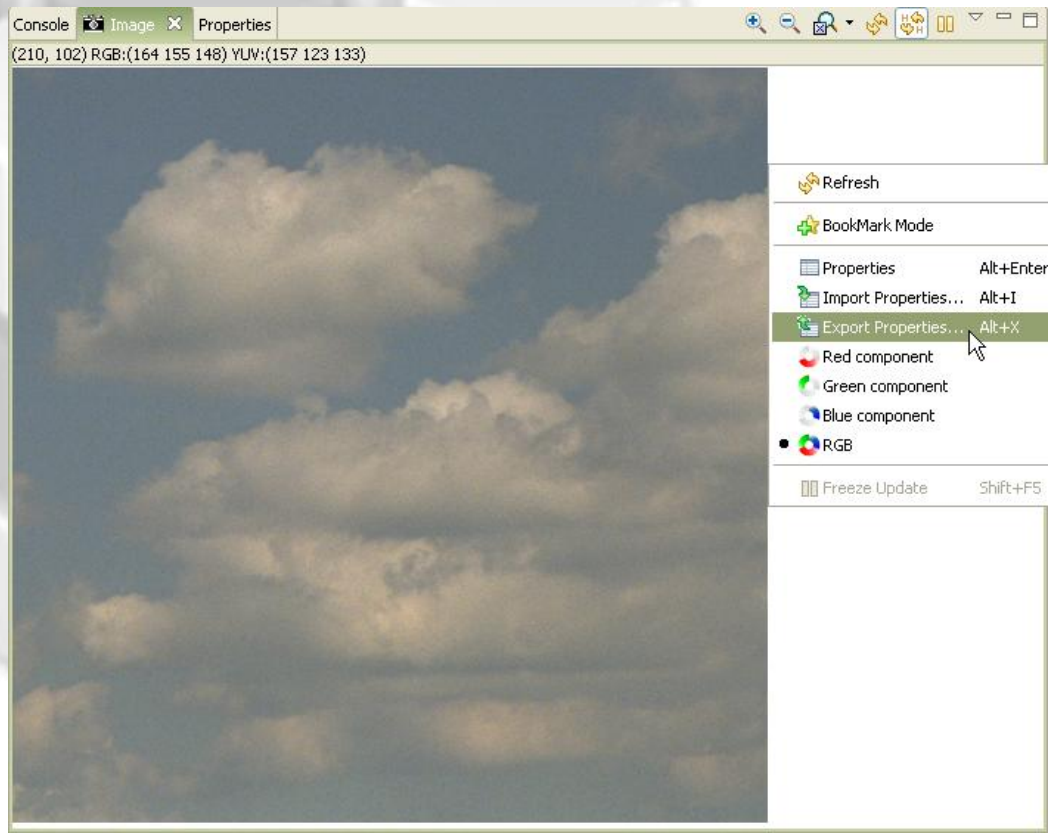


# 图表(Graphs)

- 导出图表属性到一个文本文件，方便设置的导入或分享给别人

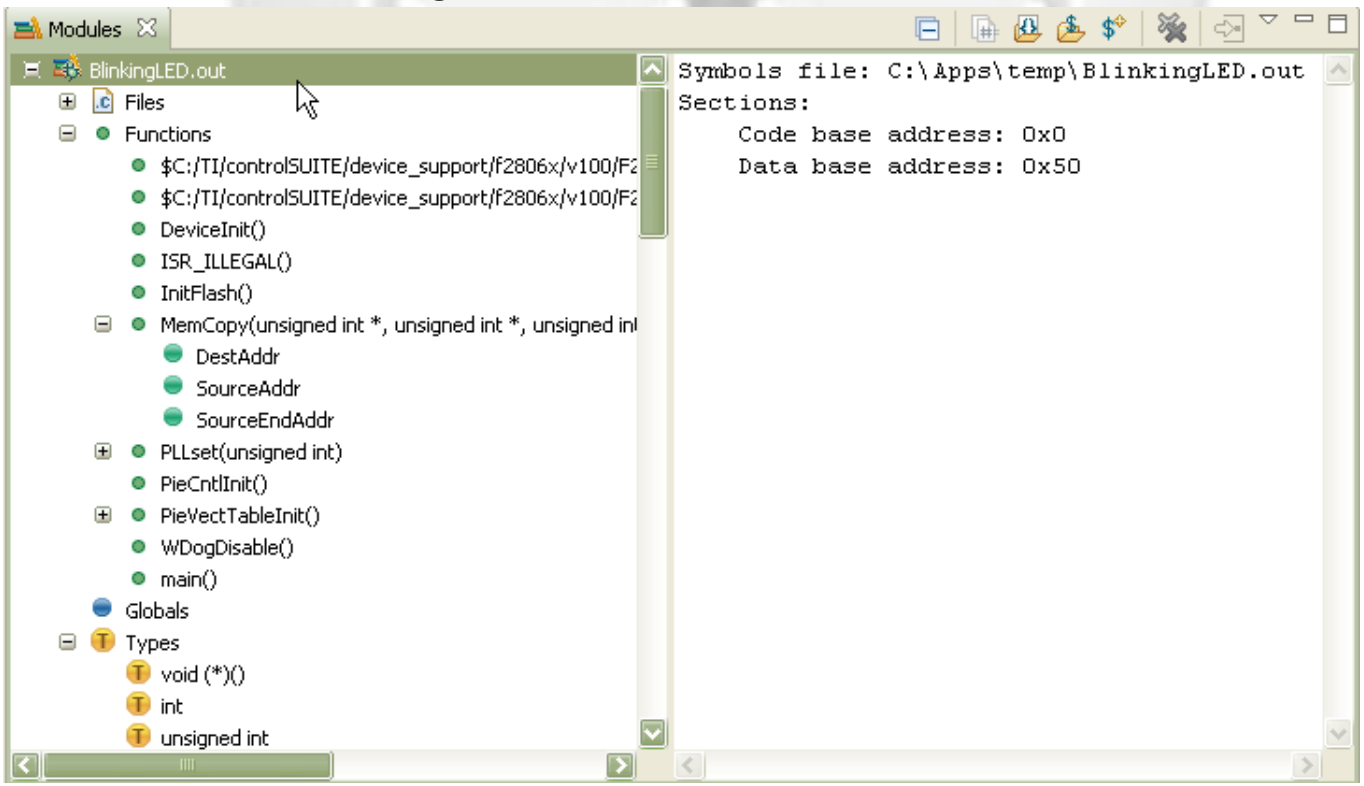
Console | Image | Properties

| Property | Value |
|---|---|
| General | |
| Title | Image |
| Background color | RGB {255, 255, 255} |
| Image format | YUV |
| YUV | |
| Number of pixels per line | 512 |
| Number of lines | 512 |
| Data format | Packed |
| Resolution | 4:2:2 |
| YUV order | UYVY |
| Y Pixel stride (bytes) | 2 |
| Y mask | 0xFF00 |
| U Pixel stride (bytes) | 4 |
| U mask | 0xFF000000 |
| V Pixel stride (bytes) | 4 |
| V mask | 0xFF000000 |
| Alpha Pixel stride (bytes) (if any) | 0 |
| Alpha mask (if any) | 0x00000000 |
| Line stride (bytes) | 1024 |
| Image source | Connected Device |
| Start address | 0x85400000 |
| Read data as | 8 bit data |



# 视图：模块(Module)

- 相当于CCSv3的“符号浏览器”(Symbol Browser)
  - 提供所有已加载的符号文件的信息
  - “查看 - >其他 - >调试 - >模块”
  - View -> Other… -> Debug -> Modules



# 脚本控制台
# SCRIPTING CONSOLE

# 脚本控制台 (Scripting Console)

- CCS的命令行操作
- 视图 - >脚本控制台 View->Scripting Console
- 按TAB查看命令列表
  - 按TAB使用命令自动完成功能
- 命令相关的文档
  - JS：> help <COMMAND>
- JavaScript和shell可以访问到所有的DSS API
- 从控制台运行DSS的脚本
- 创建自定义命令
  - 在*. js文件中创建一个JavaScript函数
  - 加载自定义的JavaScript文件
    - loadJSFile <full路径>/ myCustomConsoleCmd.js
    - 可选的第二个参数(布尔)，将决定是否自动加载脚本
  - 该函数现在可以从脚本控制台中调用

# 脚本控制台

- 视图 - >脚本控制台 View->Scripting Console
- 按TAB查看命令列表
  - 按TAB使用部分输入的命令自动完成功能
- 命令相关的文档
  - JS：> help <COMMAND>

# 脚本控制台

- 脚本控制台和GEL都可用于自动化
- GEL仅限于激活的调试会话, 并且（多数情况）只适用于一个调试上下文
- 脚本控制台随时都可以使用（虽然某些命令在没有激活的调试会话的情况下将无法正常工作）
- 脚本控制台和GEL都可以为自定义“脚本”添加菜单
  - GEL：hotmenu<function>
  - 脚本控制台：hotmenu.addJSFunction

# 脚本控制台

```
// Add entries to the 'Scripts' menu
hotmenu.addJSFunction("Launch TCI6488 Simulator, Little Endian", "tci6488_le_sim()");
hotmenu.addJSFunction("Launch TCI6488 Simulator, Big Endian", "tci6488_be_sim()");

// Path to folder with target setup ccxml files
var setupConfigFileFolder = "C:/Documents and Settings/login/user/CCSTargetConfigurations";

// configure for a TCI6488 Symmetric Simulator, Little Endian
function tci6488_le_sim()
{
   ds.setConfig(setupConfigFileFolder + "/tci6488_le_sim.ccxml");

   debugSessionCPU1 = ds.openSession("*", "C64+_0");
   debugSessionCPU2 = ds.openSession("*", "C64+_1");
   debugSessionCPU3 = ds.openSession("*", "C64+_2");
}

...
```