CCSv6 Tips & Tricks

Texas Instruments

Embedded Development Tools

Tips and Tricks

- **General**
  - Workspaces
  - Getting Started views
  - ‘Simple’ Perspective
  - Windows and Views (basics)
  - Installing Eclipse Plug-ins
- **Projects**
  - Working Sets
  - Editor Tips
  - Indexer
  - Useful CCS Edit Views
- **Debugging**
  - Debug Configurations
  - Debugging Without a Project
  - Useful Debug Views

GENERAL

Eclipse Concept: Workspaces

- Main working folder for CCS
- Contains information to manage all the projects defined to it
  - The default location of any new projects created
- User preferences, custom perspectives, cached data for plug-ins, etc are all stored in the workspace
- Multiple workspaces can be maintained
  - Only one can be active within each CCS instance
  - The same workspace cannot be shared by multiple running instances of CCS
  - It is not recommended to share workspaces amongst users

Use Multiple Workspaces

- **Multiple Users:** Keep separate workspaces for each user on a shared machine
  - Custom preferences, layouts, etc will be maintained on a per user basis
  - Each user can be working on specific project(s) that would only be applicable to a their workspace
- **Project Organization:** Break up all your CCS projects into separate workspaces for better maintenance
  - A workspace for each software release
  - A workspace for each module/feature of a release
- **Performance:** The larger the contents of the workspace (number of open projects), the greater the impact on performance of CCS

(Occasionally) Clean Your Workspace

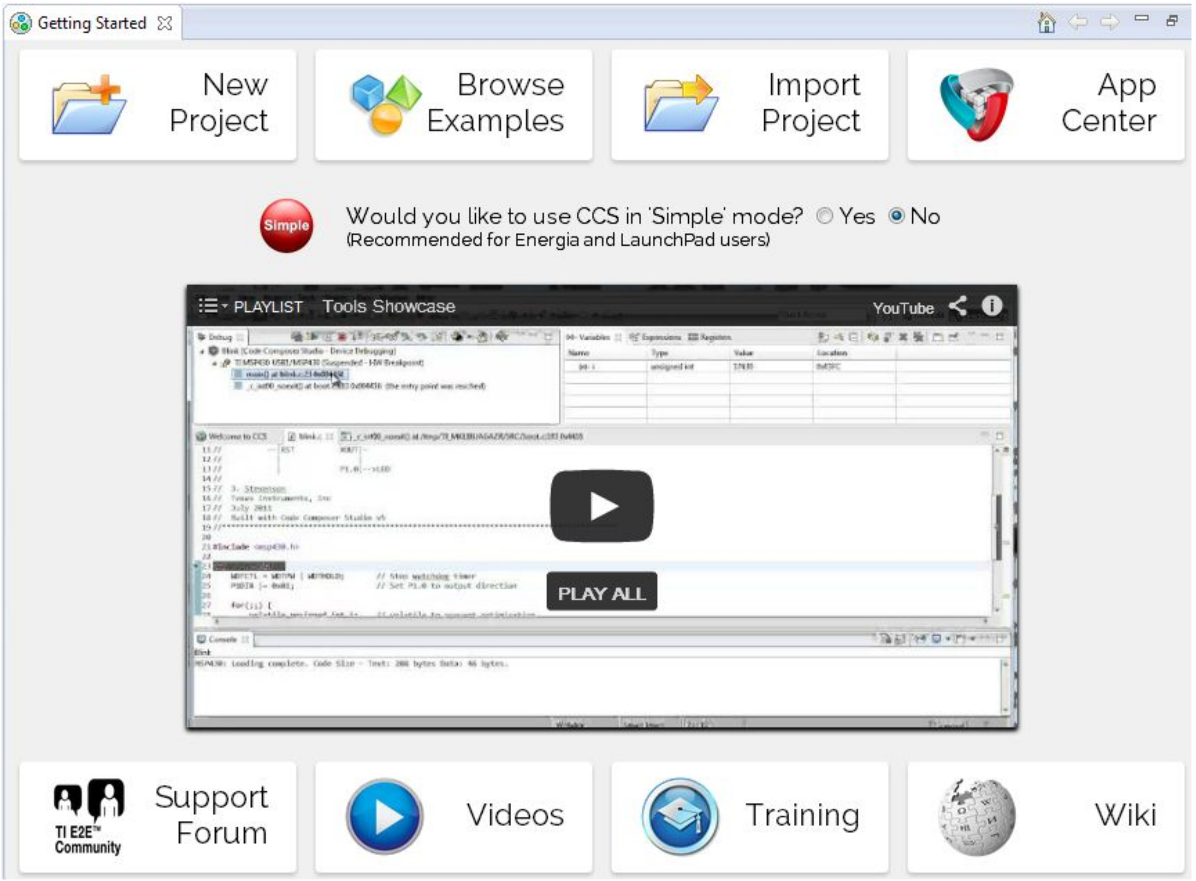
⚠ The workspace folder can get corrupted over time

💡 Good idea to periodically clean your workspace for best CCS (Eclipse) performance and stability

- To clean workspace, either:
  - Delete the **.metadata** folder in workspace folder
  - Use a new workspace folder
- Before cleaning, save current workspace settings so they can be imported into the new workspace
  - Save settings: *File->Export...->General->Preferences->To preference file*
  - Import Settings: *File->Import...->General->Preferences->From preference file*
- Any projects will have to be re-imported after cleaning the workspace

View: Getting Started

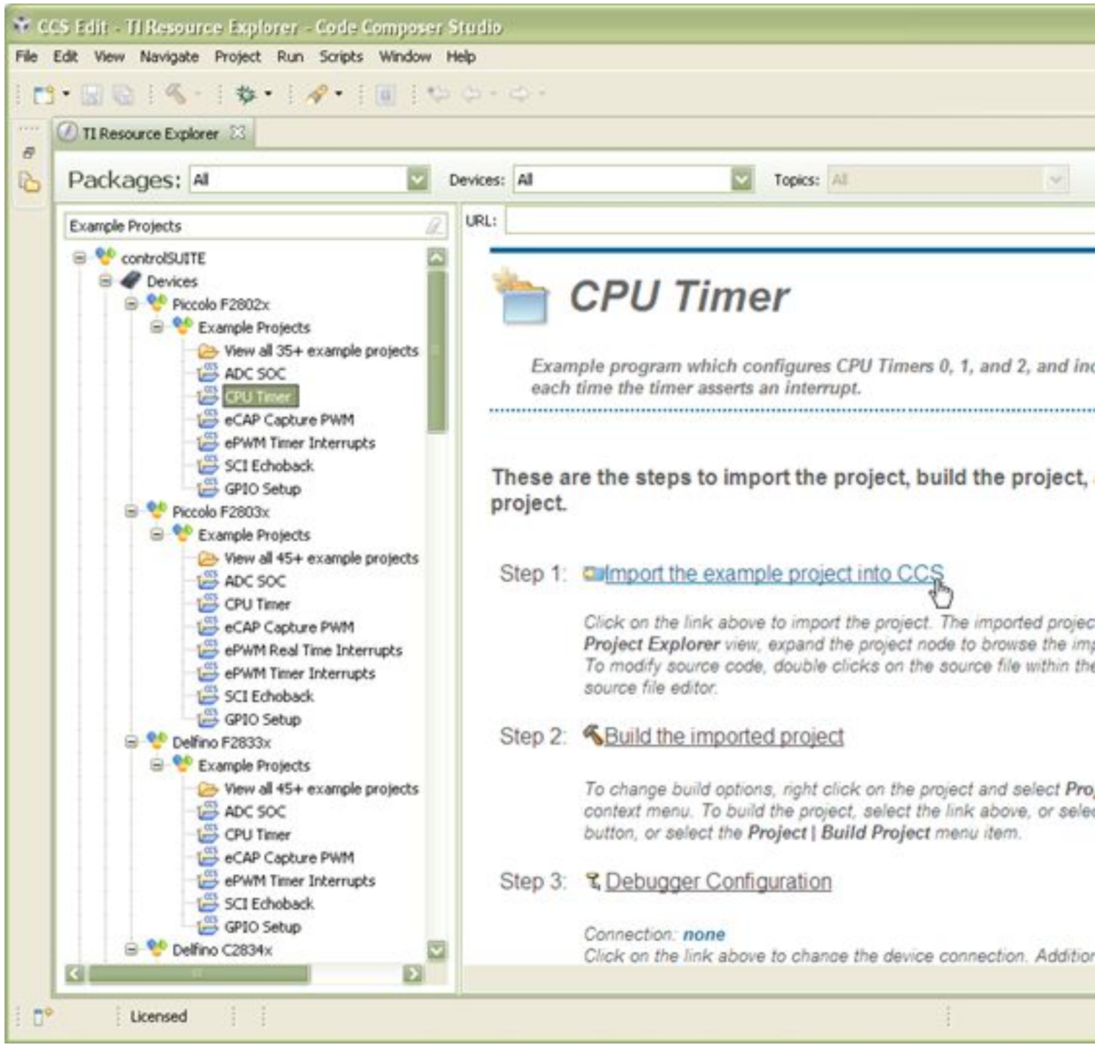
- Access a variety of resources from the **Getting Started** page:
  - Code examples
  - Support forums
  - Training material
  - CCS Videos
  - Additional TI content (App Center)
  - Create/Import projects
- Available from the **View** menu



Would you like to use CCS in 'Simple' mode? Yes [✓ No]
(Recommended for Energia and LaunchPad users)

View: TI Resource Explorer

- Easily access a broad selection of software packages including:
  - controlSUITE
  - MSP430ware
  - TivaWare
  - TI-RTOS
- Guides you step by step through using examples
- Browse resources:
  - Documentation
  - Videos


CCS Edit - TI Resource Explorer - Code Composer Studio
File Edit View Navigate Project Run Scripts Window Help
TI Resource Explorer
Licensed

Packages: All Devices: All Topics: All

Example Projects

- controlSUITE
  - Devices
    - Piccolo F2802x
      - Example Projects
        - View all 35+ example projects
        - ADC SOC
        - CPU Timer
        - eCAP Capture PWM
        - ePWM Timer Interrupts
        - SCI Echoback
        - GPIO Setup
    - Piccolo F2803x
      - Example Projects
        - View all 45+ example projects
        - ADC SOC
        - CPU Timer
        - eCAP Capture PWM
        - ePWM Real Time Interrupts
        - ePWM Timer Interrupts
        - SCI Echoback
        - GPIO Setup
    - Delfino F2833x
      - Example Projects
        - View all 45+ example projects
        - ADC SOC
        - CPU Timer
        - eCAP Capture PWM
        - ePWM Timer Interrupts
        - SCI Echoback
        - GPIO Setup
    - Delfino C2834x

URL:

CPU Timer

Example program which configures CPU Timers 0, 1, and 2, and in
each time the timer asserts an interrupt.

These are the steps to import the project, build the project,
project.

Step 1: Import the example project into CCS

Click on the link above to import the project. The imported projec
***Project Explorer** view, expand the project node to browse the imp*
To modify source code, double clicks on the source file within the
source file editor.

Step 2: Build the imported project

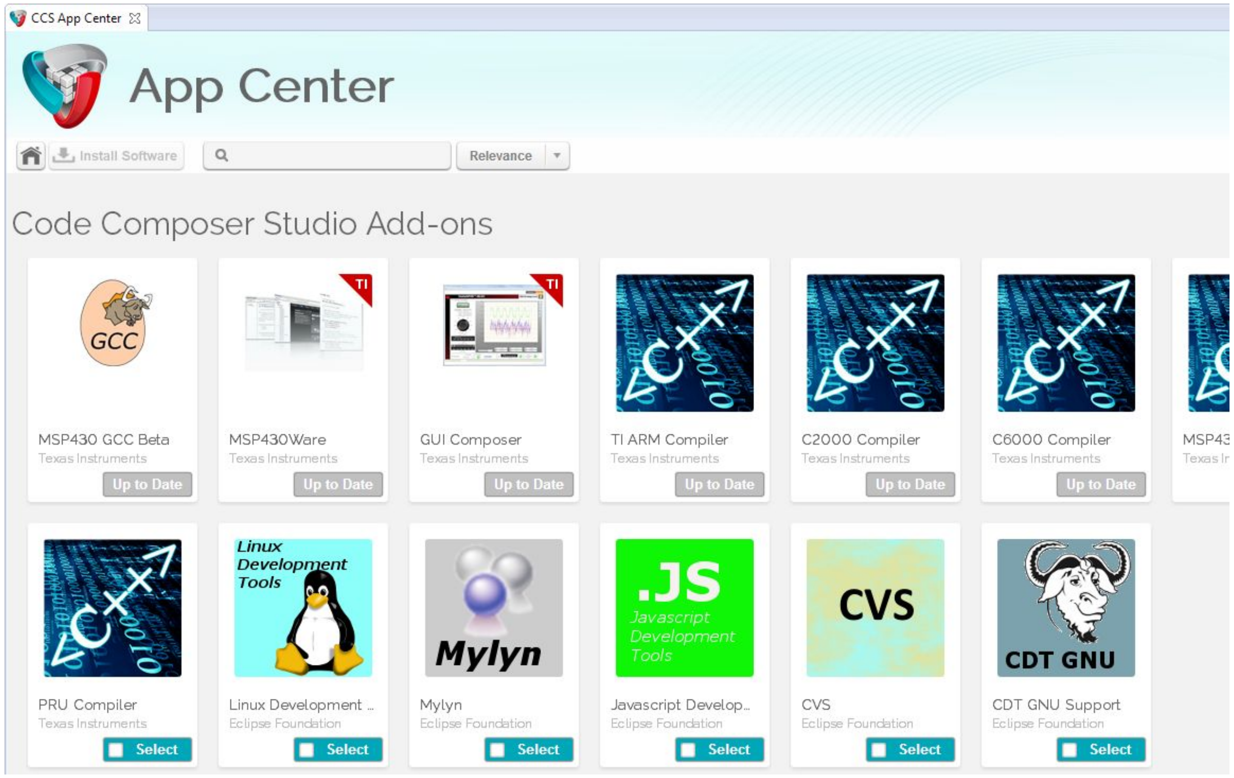
To change build options, right click on the project and select Pro
context menu. To build the project, select the link above, or sele
*button, or select the **Project | Build Project** menu item.*

Step 3: Debugger Configuration

*Connection: **none***
Click on the link above to change the device connection. Additio

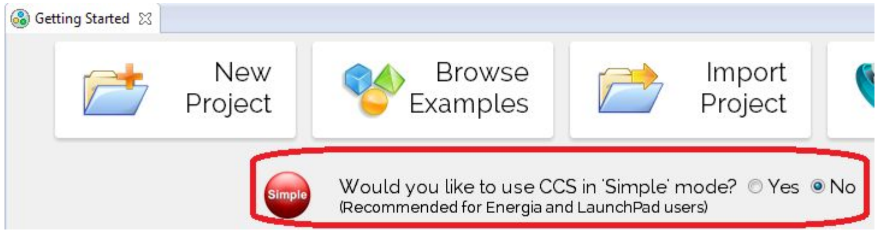
View: App Center

- Access a variety of TI content from the **CCS App Center** page
  - Browse/install additional content (plug-ins, compilers, code examples)
  - Check for additional updates to installed content
  - Available from the **View** menu

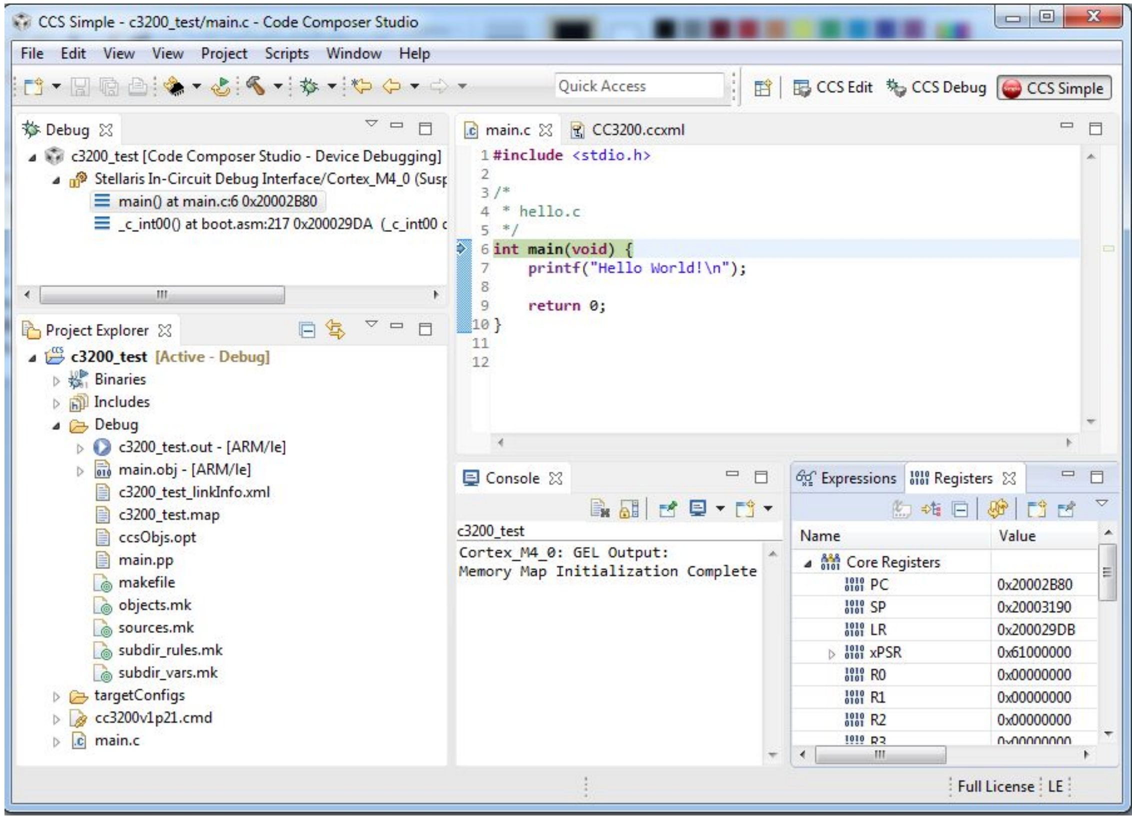


Perspective: Simple

- Single perspective that combines just the most common features of the **CCS Edit** and **CCS Debug** perspectives
  - Simplifies the environment for new users
  - Avoid perspective switching when starting a debug session
- Open the **Simple** perspective from the **Getting Started** page



Would you like to use CCS in 'Simple' mode? Yes [✓ No]
(Recommended for Energia and LaunchPad users)



Debug:
- c3200_test [Code Composer Studio - Device Debugging]
  - Stellaris In-Circuit Debug Interface/Cortex_M4_0 (Susp
    - main() at main.c:6 0x20002B80
    - _c_int00() at boot.asm:217 0x200029DA (_c_int00 c

Project Explorer:
- c3200_test [Active - Debug]
  - Binaries
  - Includes
  - Debug
    - c3200_test.out - [ARM/le]
    - main.obj - [ARM/le]
    - c3200_test_linkInfo.xml
    - c3200_test.map
    - ccsObjs.opt
    - main.pp
    - makefile
    - objects.mk
    - sources.mk
    - subdir_rules.mk
    - subdir_vars.mk
  - targetConfigs
  - cc3200v1p21.cmd
  - main.c

```
#include <stdio.h>

/*
 * hello.c
 */
int main(void) {
    printf("Hello World!\n");

    return 0;
}
```

Console:

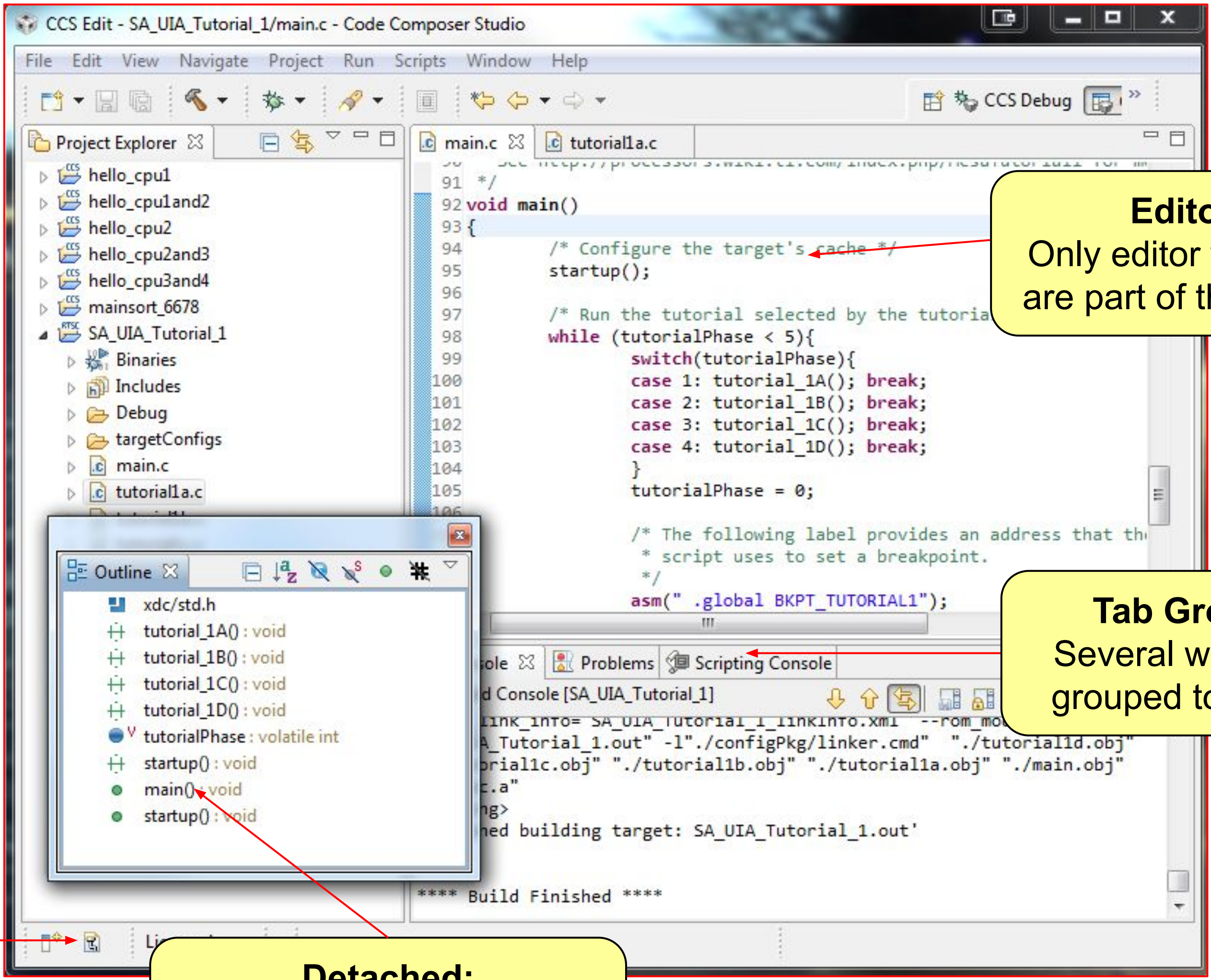
```
c3200_test
Cortex_M4_0: GEL Output:
Memory Map Initialization Complete
```

Registers:

Name	Value
Core Registers	
PC	0x20002B80
SP	0x20003190
LR	0x200029DB
xPSR	0x61000000
R0	0x00000000
R1	0x00000000
R2	0x00000000
R3	0x00000000

WINDOWS AND VIEWS

Window Types



```
 */
void main()
{
        /* Configure the target's cache */
        startup();

        /* Run the tutorial selected by the tutoria
        while (tutorialPhase < 5){
                switch(tutorialPhase){
                case 1: tutorial_1A(); break;
                case 2: tutorial_1B(); break;
                case 3: tutorial_1C(); break;
                case 4: tutorial_1D(); break;
                }
                tutorialPhase = 0;

                /* The following label provides an address that th
                 * script uses to set a breakpoint.
                 */
                asm(" .global BKPT_TUTORIAL1");
```

```
link_info= SA_UIA_Tutorial_1_linkInfo.xml" --rom_mo
A_Tutorial_1.out" -l"./configPkg/linker.cmd"  "./tutorial1d.obj"
orial1c.obj" "./tutorial1b.obj" "./tutorial1a.obj" "./main.obj"
c.a"
ng>
hed building target: SA_UIA_Tutorial_1.out'

**** Build Finished ****
```

Editor:
Only editor windows are part of this group

Tab Group:
Several windows grouped together

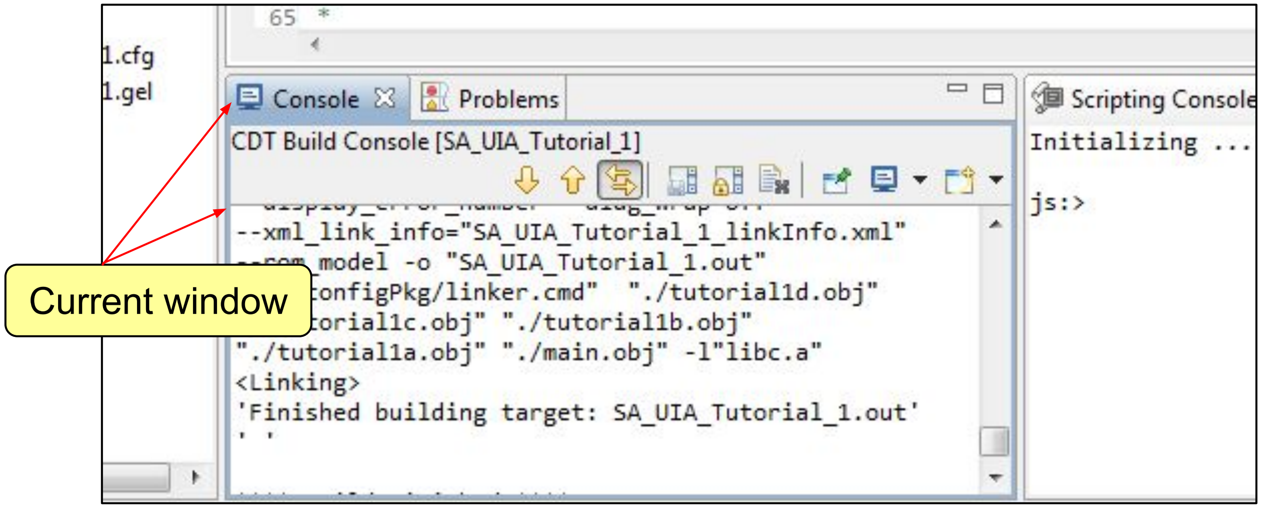
Fast view: Hidden until you click on the button to restore them. Click on another window to hide.

Detached:
Floating window not tied to Workbench

Texas Instruments

Windowing Tips

- Double-clicking on the title bar of a window will maximize the window
  - Double-clicking again will restore it to its previous size
- Fast-view windows are great for windows you use infrequently but need a lot of space when you do use them
- The window that has focus is indicated by a **blue** border and heading

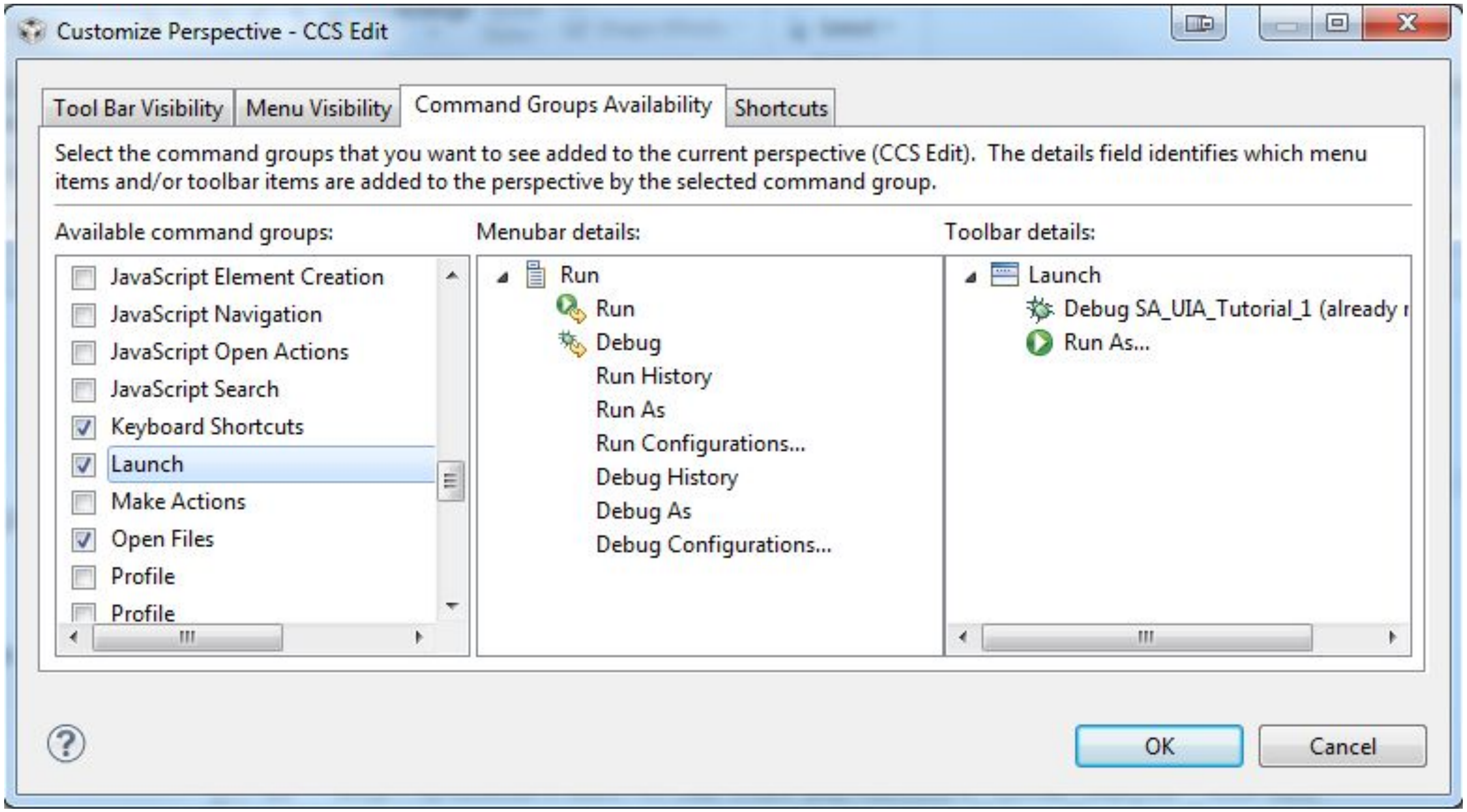


```
--xml_link_info="SA_UIA_Tutorial_1_linkInfo.xml"
--rom_model -o "SA_UIA_Tutorial_1.out"
 configPkg/linker.cmd"  "./tutorial1d.obj"
 orial1c.obj" "./tutorial1b.obj"
"./tutorial1a.obj" "./main.obj" -l"libc.a"
<Linking>
'Finished building target: SA_UIA_Tutorial_1.out'
' '
```

```
Initializing ...

js:>
```

Customizing Perspectives

- You can customize the menu items and toolbars in your perspective
  - Right click on the toolbar and select **Customize Perspective**



Customize Perspective - CCS Edit

Tool Bar Visibility | Menu Visibility | Command Groups Availability | Shortcuts

Select the command groups that you want to see added to the current perspective (CCS Edit). The details field identifies which menu items and/or toolbar items are added to the perspective by the selected command group.

Available command groups:
- [ ] JavaScript Element Creation
- [ ] JavaScript Navigation
- [ ] JavaScript Open Actions
- [ ] JavaScript Search
- [x] Keyboard Shortcuts
- [x] Launch
- [ ] Make Actions
- [x] Open Files
- [ ] Profile
- [ ] Profile

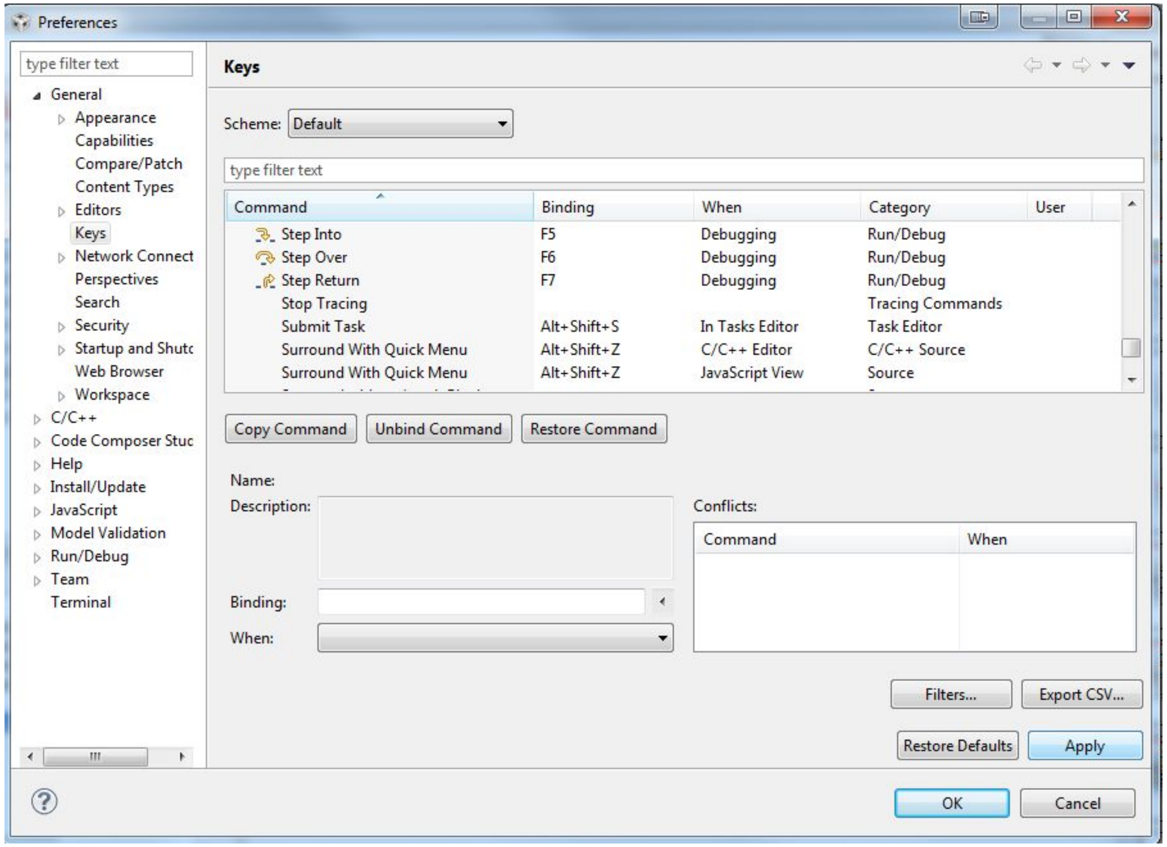
Menubar details:
```
Run
    Run
    Debug
    Run History
    Run As
    Run Configurations...
    Debug History
    Debug As
    Debug Configurations...
```

Toolbar details:
```
Launch
    Debug SA_UIA_Tutorial_1 (already r
    Run As...
```

OK | Cancel

Using the Keyboard…

- All key bindings can be viewed and modified: *Window -> Preferences -> General -> Keys*
- Key bindings are part of the general preferences that can be exported to a preferences file and imported



Scheme: Default

Command	Binding	When	Category	User
Step Into	F5	Debugging	Run/Debug	
Step Over	F6	Debugging	Run/Debug	
Step Return	F7	Debugging	Run/Debug	
Stop Tracing			Tracing Commands	
Submit Task	Alt+Shift+S	In Tasks Editor	Task Editor	
Surround With Quick Menu	Alt+Shift+Z	C/C++ Editor	C/C++ Source	
Surround With Quick Menu	Alt+Shift+Z	JavaScript View	Source	

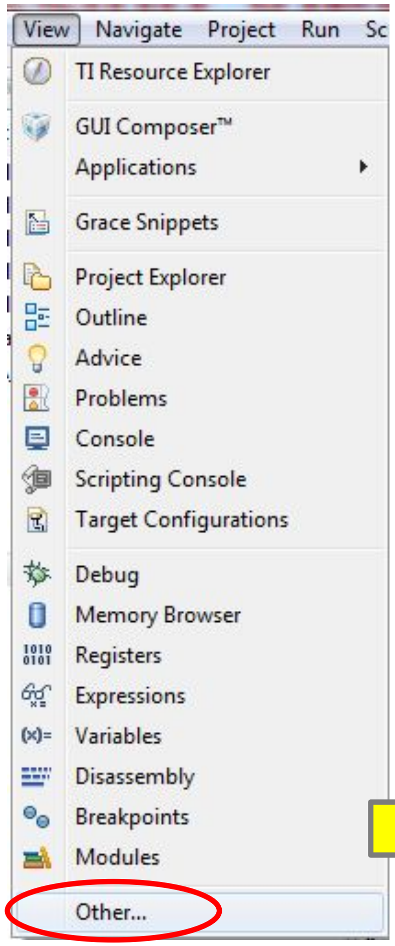
Name:
Description:
Binding:
When:

Conflicts:

Command	When

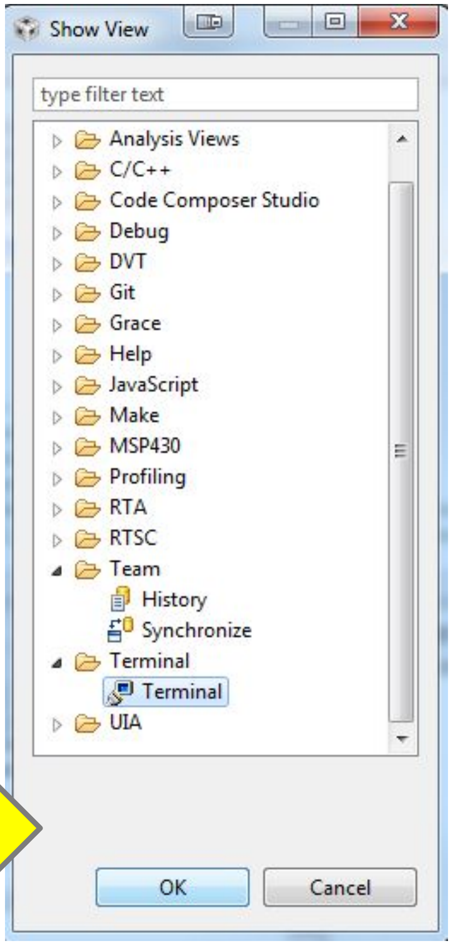
Accessing Views

- To open a new view go to the **View** menu
  - List the most commonly used views with CCS
- To access views that are not listed select **Other…**
- Many useful “hidden” views…



View Navigate Project Run Sc

- TI Resource Explorer
- GUI Composer™
- Applications
- Grace Snippets
- Project Explorer
- Outline
- Advice
- Problems
- Console
- Scripting Console
- Target Configurations
- Debug
- Memory Browser
- Registers
- Expressions
- Variables
- Disassembly
- Breakpoints
- Modules
- Other...



Show View

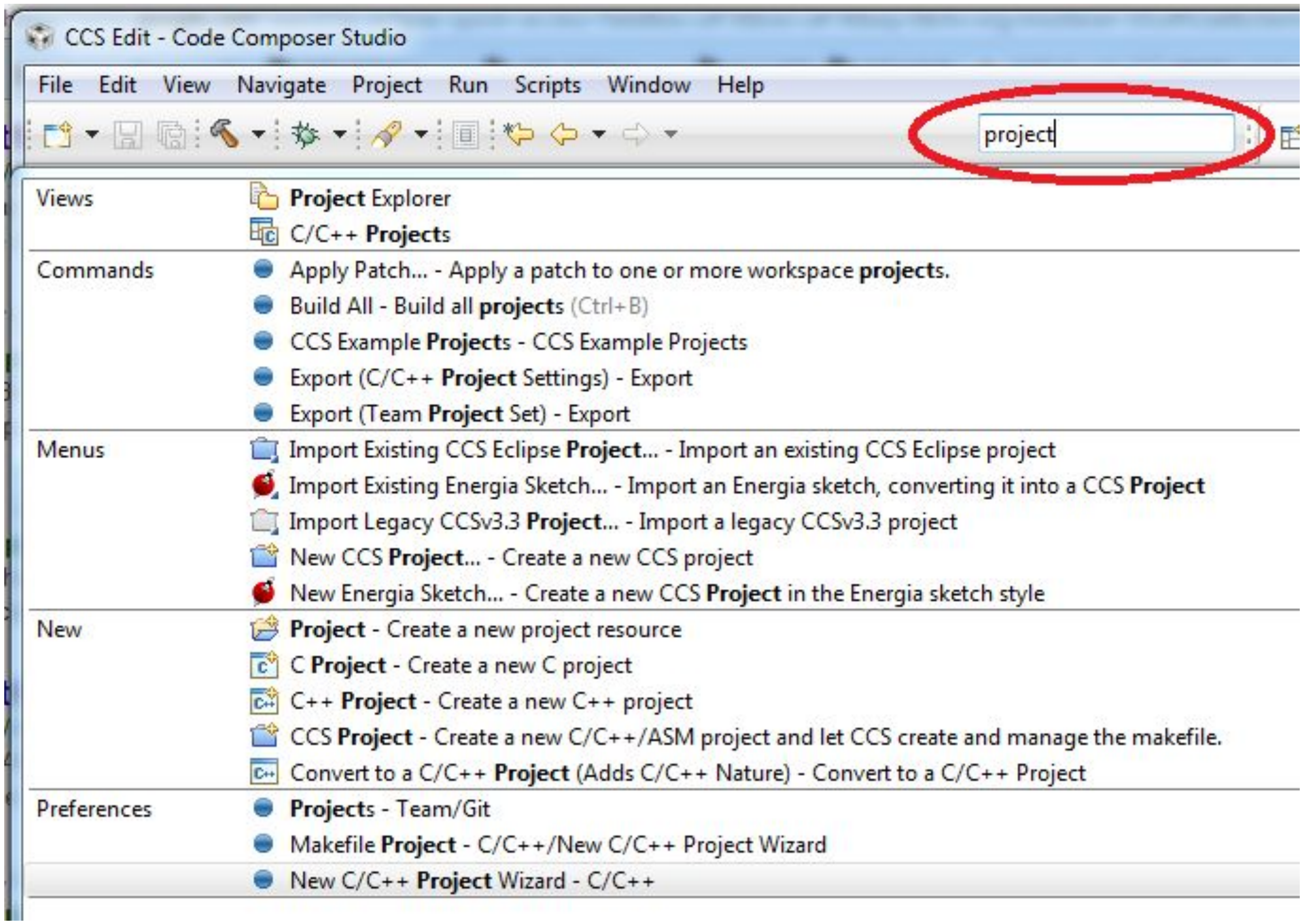
type filter text

- Analysis Views
- C/C++
- Code Composer Studio
- Debug
- DVT
- Git
- Grace
- Help
- JavaScript
- Make
- MSP430
- Profiling
- RTA
- RTSC
- Team
  - History
  - Synchronize
- Terminal
  - Terminal
- UIA

OK Cancel

Quick Access Field

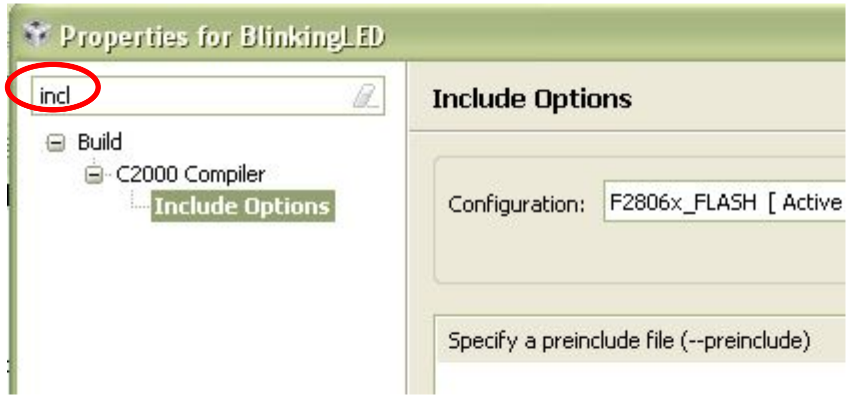
- Global search field that will search all of CCS
- Search for:
  - Views
  - Commands
  - Menu items
  - Preferences


CCS Edit - Code Composer Studio
File Edit View Navigate Project Run Scripts Window Help
project

Views	**Project** Explorer
	C/C++ **Projects**
Commands	Apply Patch... - Apply a patch to one or more workspace **projects**.
	Build All - Build all **projects** (Ctrl+B)
	CCS Example **Projects** - CCS Example Projects
	Export (C/C++ **Project** Settings) - Export
	Export (Team **Project** Set) - Export
Menus	Import Existing CCS Eclipse **Project...** - Import an existing CCS Eclipse project
	Import Existing Energia Sketch... - Import an Energia sketch, converting it into a CCS **Project**
	Import Legacy CCSv3.3 **Project...** - Import a legacy CCSv3.3 project
	New CCS **Project...** - Create a new CCS project
	New Energia Sketch... - Create a new CCS **Project** in the Energia sketch style
New	**Project** - Create a new project resource
	C **Project** - Create a new C project
	C++ **Project** - Create a new C++ project
	CCS **Project** - Create a new C/C++/ASM project and let CCS create and manage the makefile.
	Convert to a C/C++ **Project** (Adds C/C++ Nature) - Convert to a C/C++ Project
Preferences	**Projects** - Team/Git
	Makefile **Project** - C/C++/New C/C++ Project Wizard
	New C/C++ **Project** Wizard - C/C++

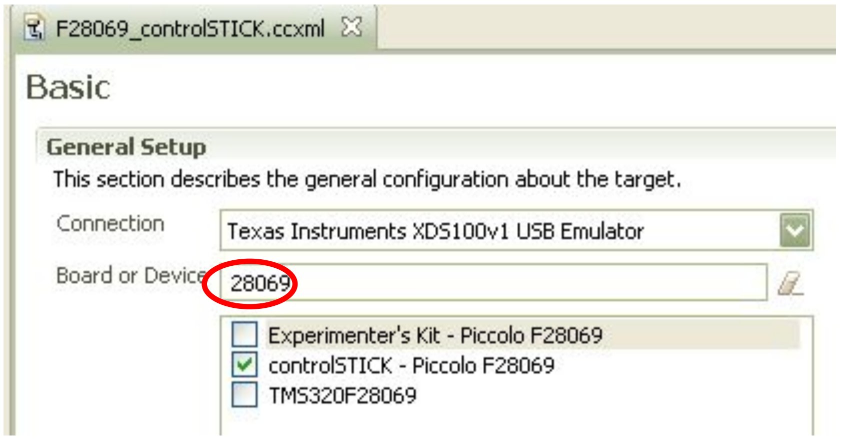
Filter Field

- Use it to find options/properties faster in the scope of a specific dialog/view
  - Narrows list of options down depending on characters entered in the field
- Available in many dialogs
  - Project Properties
  - Window Properties
  - Workspace Properties..
  - Target Configuration view



Configuration: F2806x_FLASH [Active

Specify a preinclude file (--preinclude)

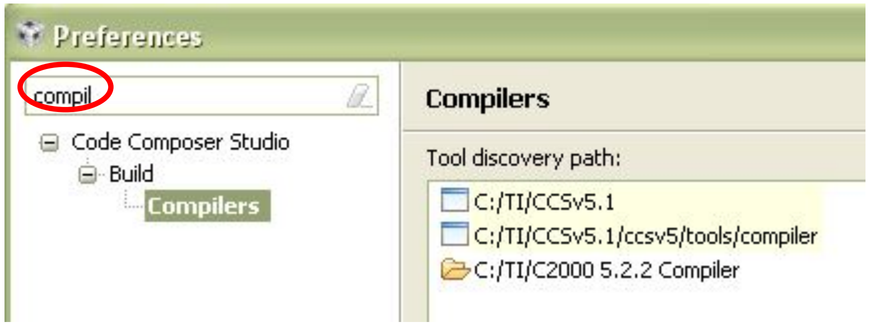


This section describes the general configuration about the target.

Connection: Texas Instruments XDS100v1 USB Emulator

Board or Device: 28069

- [ ] Experimenter's Kit - Piccolo F28069
- [✓] controlSTICK - Piccolo F28069
- [ ] TMS320F28069



Tool discovery path:

C:/TI/CCSv5.1
C:/TI/CCSv5.1/ccsv5/tools/compiler
C:/TI/C2000 5.2.2 Compiler

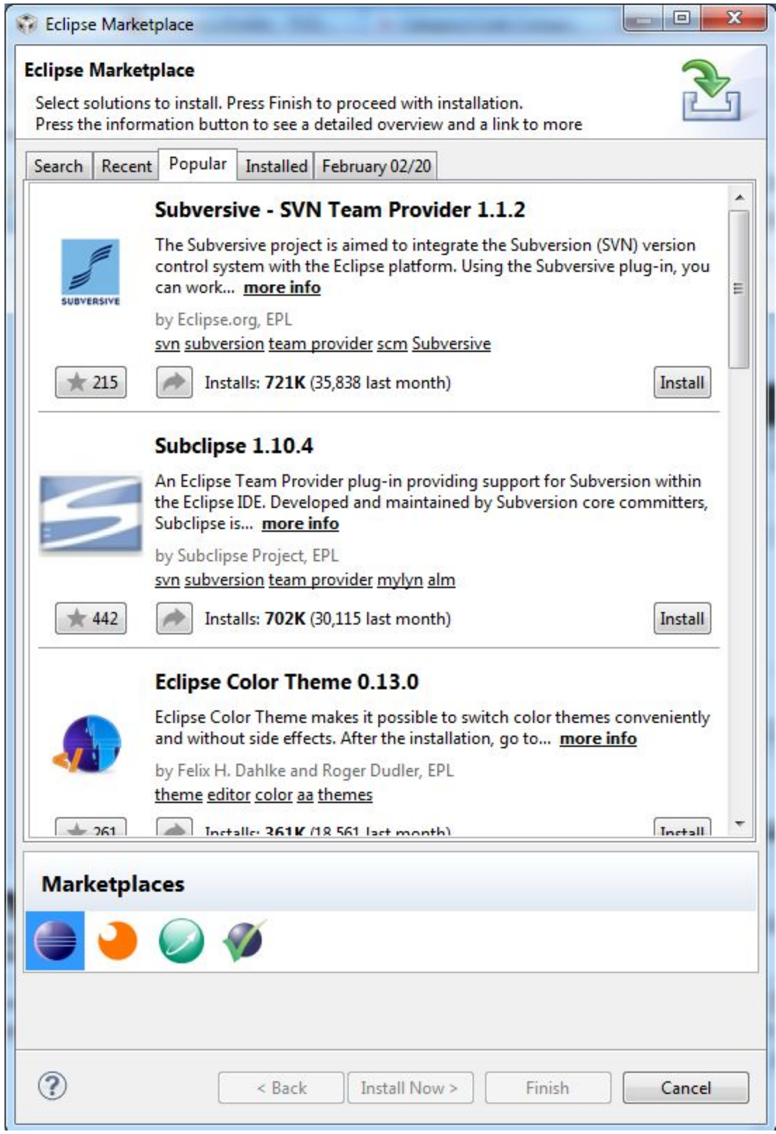
ECLIPSE PLUG-INS

Eclipse Plug-ins - Basics

- CCSv6 is based on Eclipse and is able to leverage many of the huge selection of 3rd party Eclipse plug-ins available
  - http://marketplace.eclipse.org/
- CCSv6.0 is based off Eclipse 4.3 and CDT 8.2
  - Look for plug-ins that support this version for best compatibility
- CCS **App Center** focuses on TI specific content and only shows a tiny fraction of available Eclipse plug-ins

Eclipse Plug-ins - Marketplace

- Find additional Eclipse plug-ins from within CCS with the **Eclipse Marketplace** plug-in
  - *Help -> Eclipse Marketplace…*
  - Browse through a list of featured or popular plug-ins
  - Search for plug-ins with a keyword search
  - Easy install of any plug-in found in the marketplace


Eclipse Marketplace

Eclipse Marketplace

Select solutions to install. Press Finish to proceed with installation.
Press the information button to see a detailed overview and a link to more

Search | Recent | Popular | Installed | February 02/20

Subversive - SVN Team Provider 1.1.2

The Subversive project is aimed to integrate the Subversion (SVN) version control system with the Eclipse platform. Using the Subversive plug-in, you can work... more info

by Eclipse.org, EPL

svn subversion team provider scm Subversive

215 Installs: **721K** (35,838 last month) Install

Subclipse 1.10.4

An Eclipse Team Provider plug-in providing support for Subversion within the Eclipse IDE. Developed and maintained by Subversion core committers, Subclipse is... more info

by Subclipse Project, EPL

svn subversion team provider mylyn alm

442 Installs: **702K** (30,115 last month) Install

Eclipse Color Theme 0.13.0

Eclipse Color Theme makes it possible to switch color themes conveniently and without side effects. After the installation, go to... more info

by Felix H. Dahlke and Roger Dudler, EPL

theme editor color aa themes

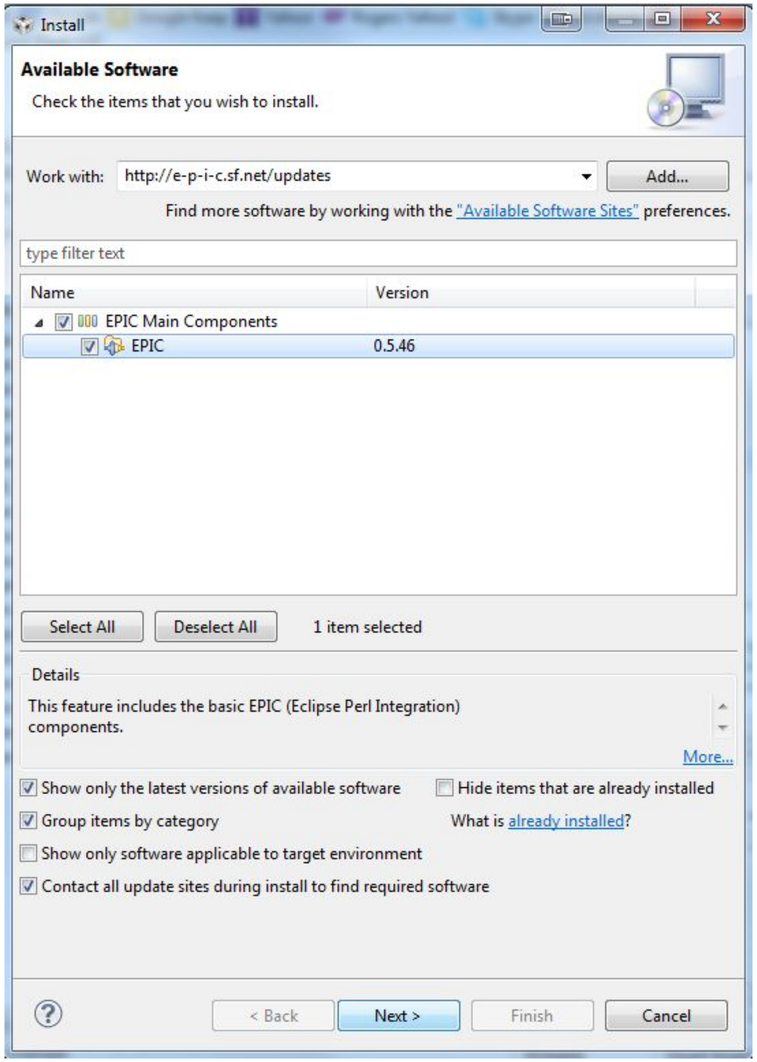
261 Installs: 361K (18,561 last month) Install

Marketplaces

< Back | Install Now > | Finish | Cancel

Eclipse Plug-ins - Installation

- Use the **Eclipse Update Manager** to install plug-ins if the plug-in update site is already known
  - *Help -> Install New Software* for new updates to install (specify remote site (URL) or local site (directory))
- Drop-in plugins manually
  - Many plug-ins can be simply downloaded as an archive and copied into the *.\ccsv6\eclipse\dropins* folder



Work with: http://e-p-i-c.sf.net/updates

Find more software by working with the "Available Software Sites" preferences.

Name	Version
☑ EPIC Main Components	
☑ EPIC	0.5.46

1 item selected

This feature includes the basic EPIC (Eclipse Perl Integration) components.

☑ Show only the latest versions of available software ☐ Hide items that are already installed

☑ Group items by category What is already installed?

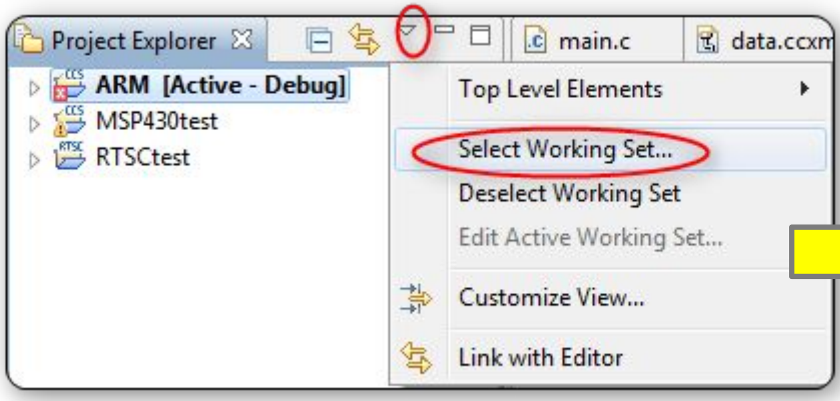
☐ Show only software applicable to target environment

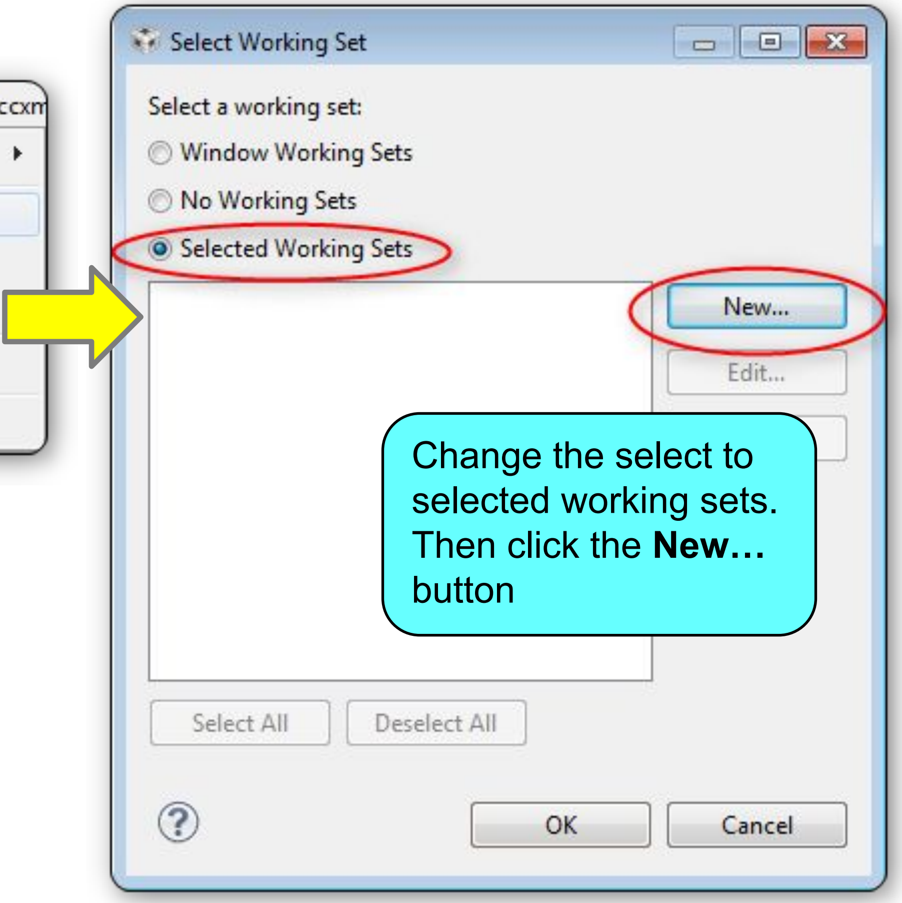
☑ Contact all update sites during install to find required software

PROJECTS

Working Sets (1)

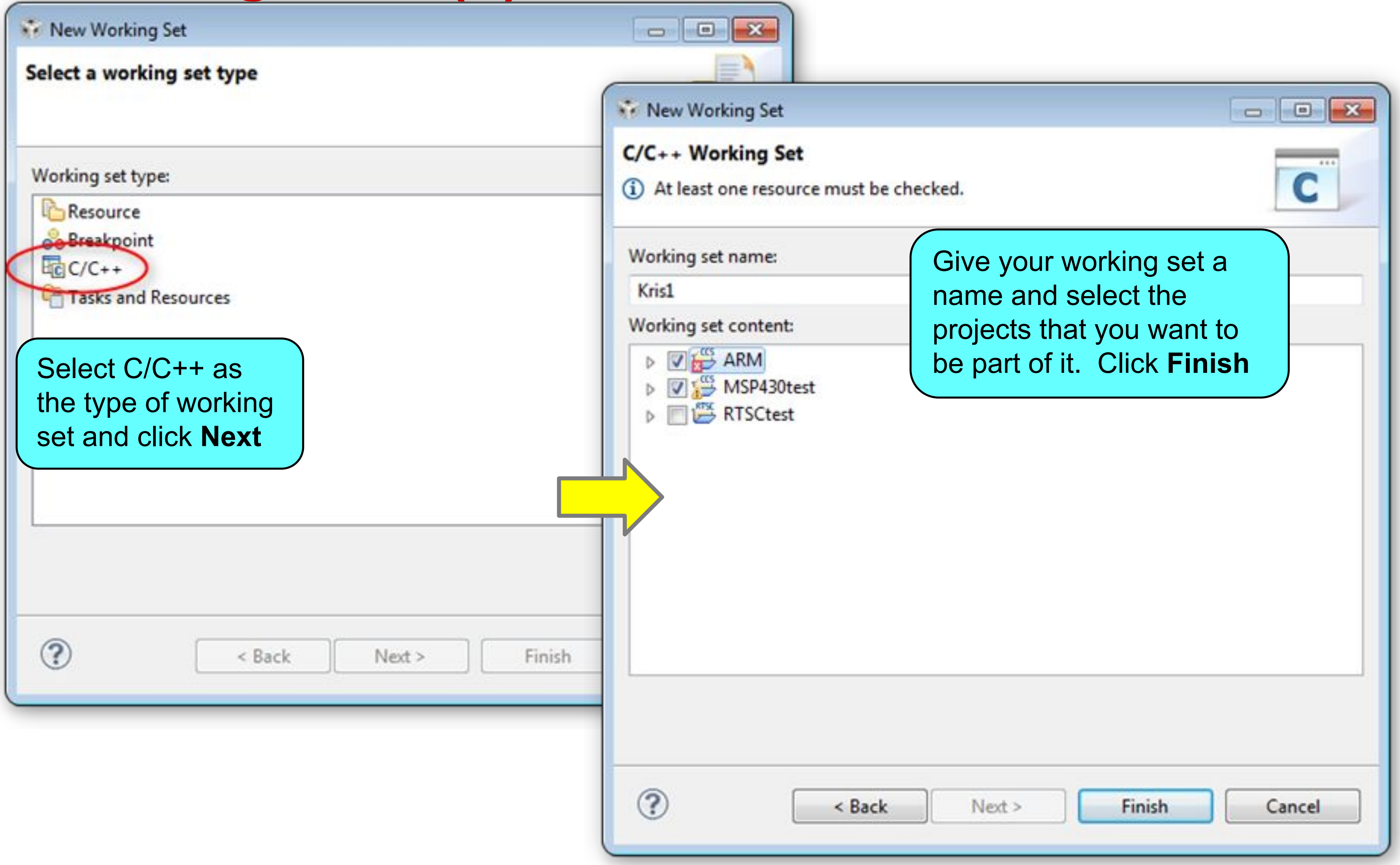
- Use working sets to group projects inside the Project Explorer
  - Useful for multi-core environments to clearly differentiate projects by which core they are for





Change the select to selected working sets. Then click the **New…** button

Working Sets (2)


New Working Set

Select a working set type

Working set type:

- Resource
- Breakpoint
- C/C++
- Tasks and Resources

Select C/C++ as the type of working set and click **Next**

New Working Set

C/C++ Working Set

At least one resource must be checked.

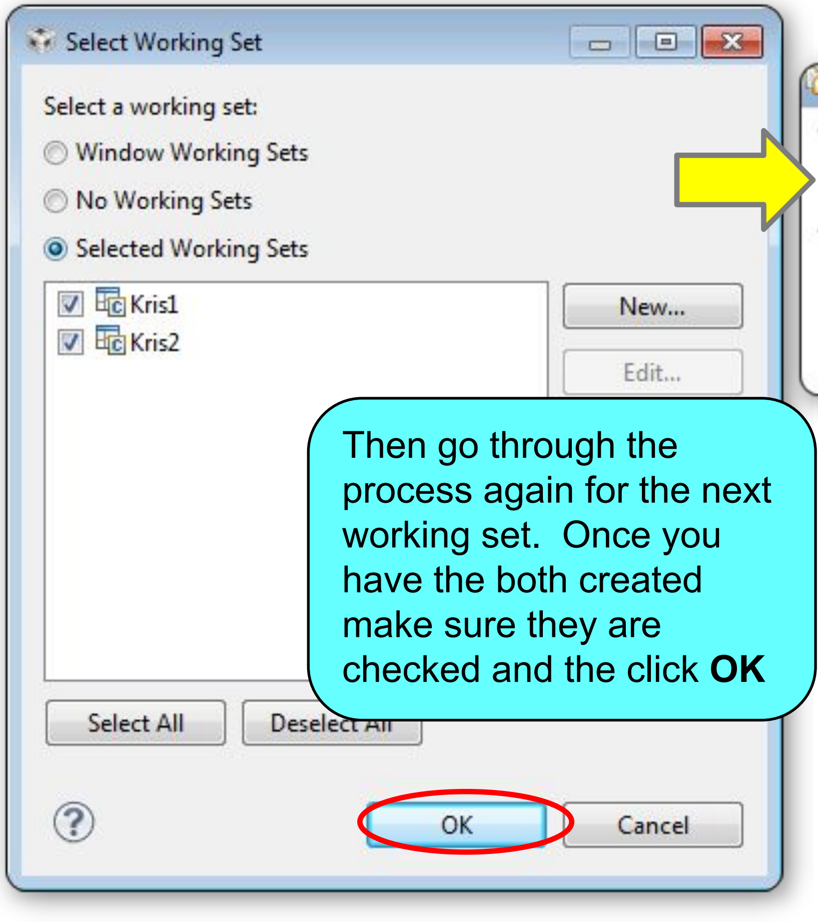
Working set name: Kris1

Working set content:

- [✓] ARM
- [✓] MSP430test
- [ ] RTSCtest

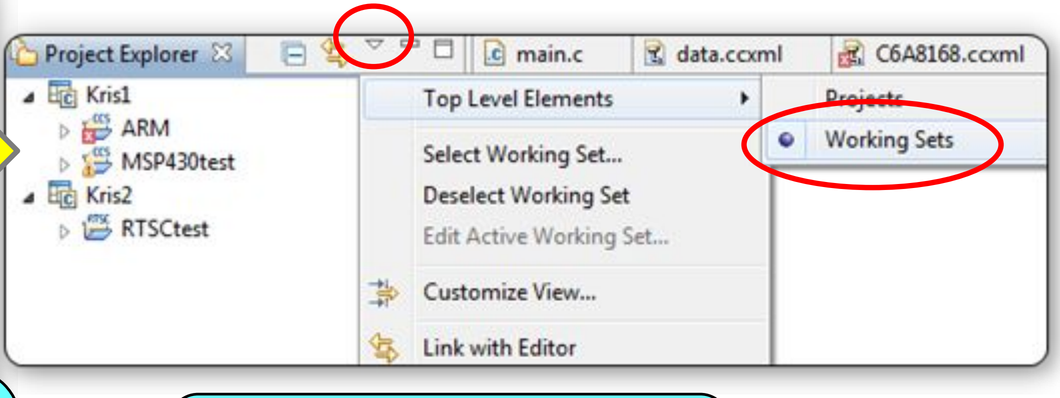
Give your working set a name and select the projects that you want to be part of it. Click **Finish**

Working Sets (3)



Select a working set:
- ( ) Window Working Sets
- ( ) No Working Sets
- [✓ Selected Working Sets]
- [✓] Kris1
- [✓] Kris2

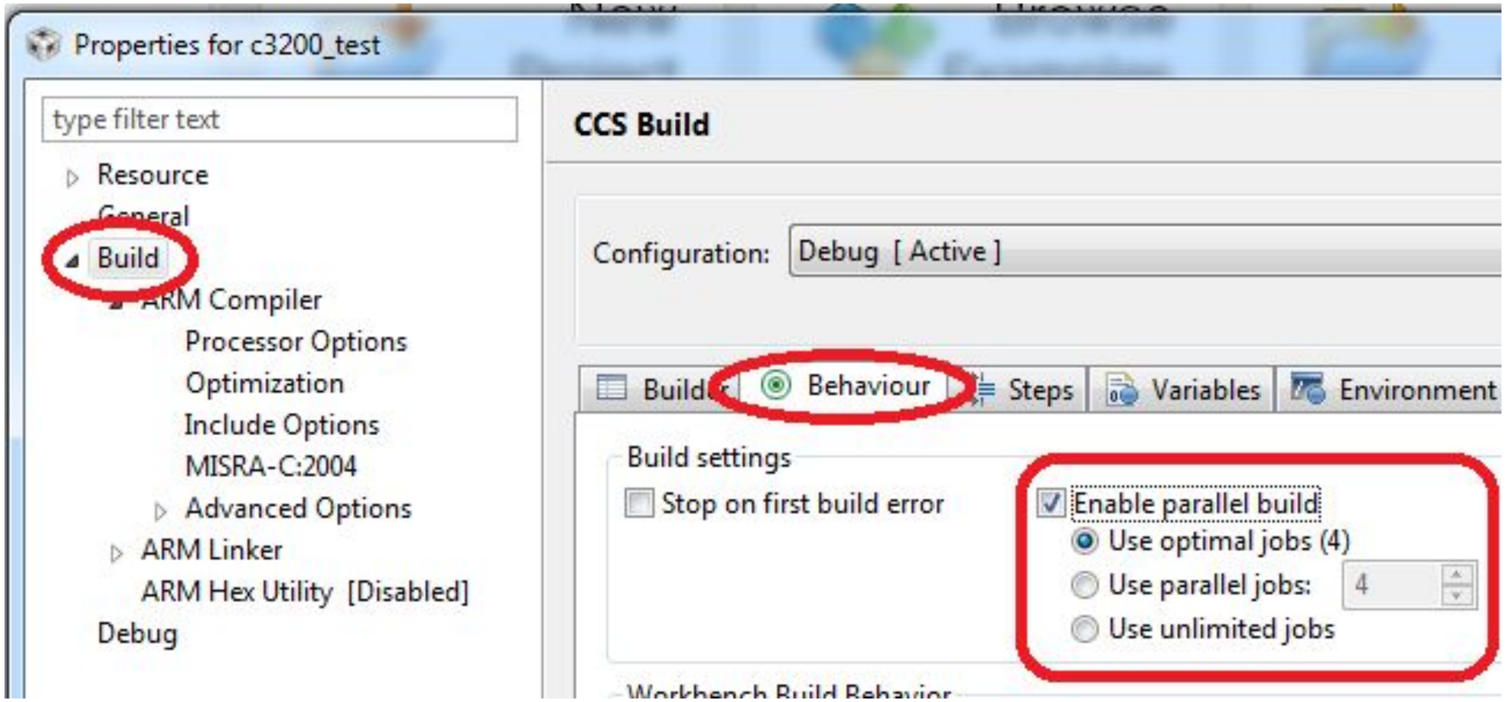
Then go through the process again for the next working set. Once you have the both created make sure they are checked and the click **OK**



Now you need to change the project explorer to organize by the working set

Parallel Builds

- Have a multi-core PC? Take advantage of all those cores to speed up your CCS project builds with parallel builds!
- In the project properties, under the **Behavior** tab under **Build**, turn on the **Enable parallel build** option



Properties for c3200_test

type filter text

- Resource
- General
- Build
  - ARM Compiler
    - Processor Options
    - Optimization
    - Include Options
    - MISRA-C:2004
    - Advanced Options
  - ARM Linker
  - ARM Hex Utility [Disabled]
- Debug

CCS Build

Configuration: Debug [Active]

Build settings

- [ ] Stop on first build error
- [✓] Enable parallel build
  - [● Use optimal jobs (4)]
  - Use parallel jobs: 4
  - Use unlimited jobs

Workbench Build Behavior

Source Code Editor

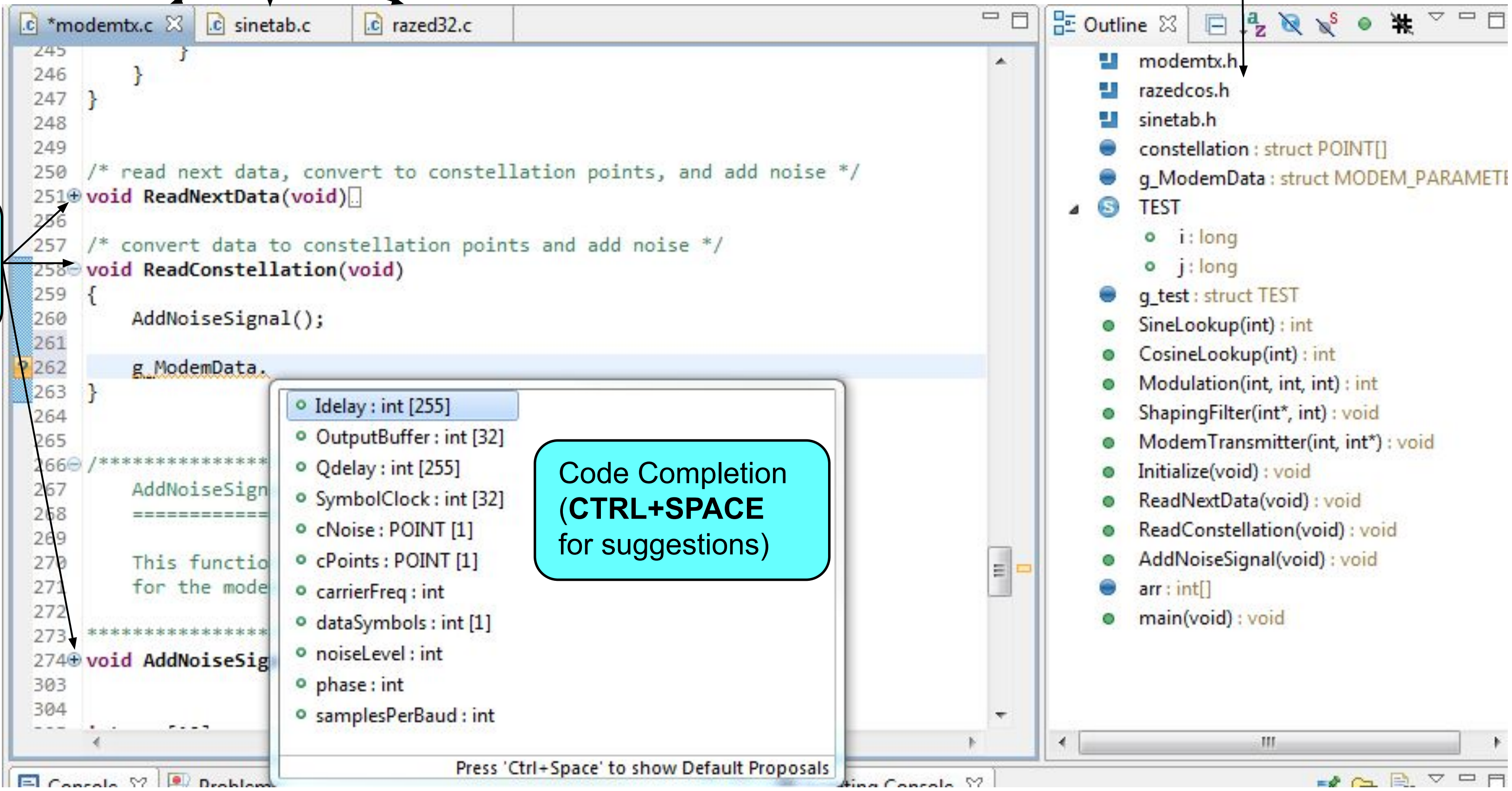
- CCS comes with an excellent, feature rich editor

Outline view displays lists structural elements for the selected source file

Editor tabs

Collapse and expand functions

Code Completion (**CTRL+SPACE** for suggestions)



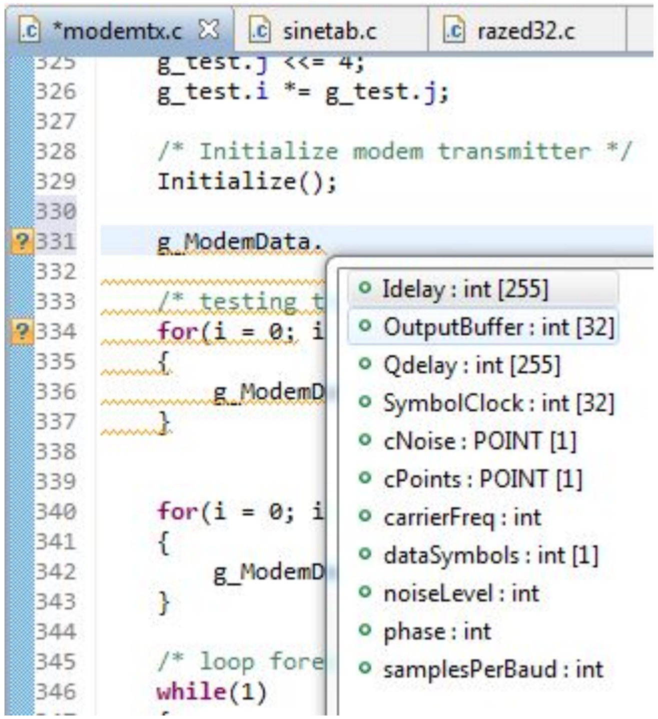
```
        }
}


/* read next data, convert to constellation points, and add noise */
void ReadNextData(void)

/* convert data to constellation points and add noise */
void ReadConstellation(void)
{
    AddNoiseSignal();

    g_ModemData.
}


/***************
    AddNoiseSign
    ============

    This functio
    for the mode

***************
void AddNoiseSig
```

```
Idelay : int [255]
OutputBuffer : int [32]
Qdelay : int [255]
SymbolClock : int [32]
cNoise : POINT [1]
cPoints : POINT [1]
carrierFreq : int
dataSymbols : int [1]
noiseLevel : int
phase : int
samplesPerBaud : int
Press 'Ctrl+Space' to show Default Proposals
```

```
modemtx.h
razedcos.h
sinetab.h
constellation : struct POINT[]
g_ModemData : struct MODEM_PARAMETE
TEST
    i : long
    j : long
g_test : struct TEST
SineLookup(int) : int
CosineLookup(int) : int
Modulation(int, int, int) : int
ShapingFilter(int*, int) : void
ModemTransmitter(int, int*) : void
Initialize(void) : void
ReadNextData(void) : void
ReadConstellation(void) : void
AddNoiseSignal(void) : void
arr : int[]
main(void) : void
```

Advanced Editor Features

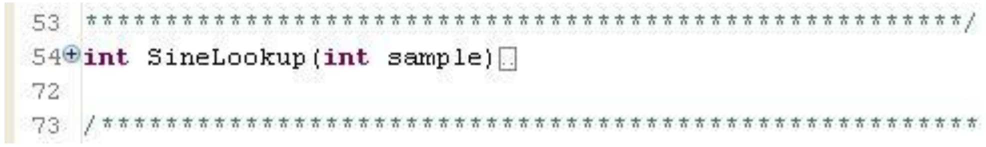
- Code Completion
  - Complete word
  - Auto-member information
  - Auto-parameter information
- Navigation
  - Back/Forward buttons
  - Back to last edit button
  - Go to definition
  - Go to declaration
- Code Folding
  - Collapse functions



*modemtx.c | sinetab.c | razed32.c

```
    g_test.j <<= 4;
    g_test.i *= g_test.j;

    /* Initialize modem transmitter */
    Initialize();

    g_ModemData.

    /* testing t
    for(i = 0; i
    {
        g_ModemD
    }


    for(i = 0; i
    {
        g_ModemD
    }

    /* loop fore
    while(1)
```

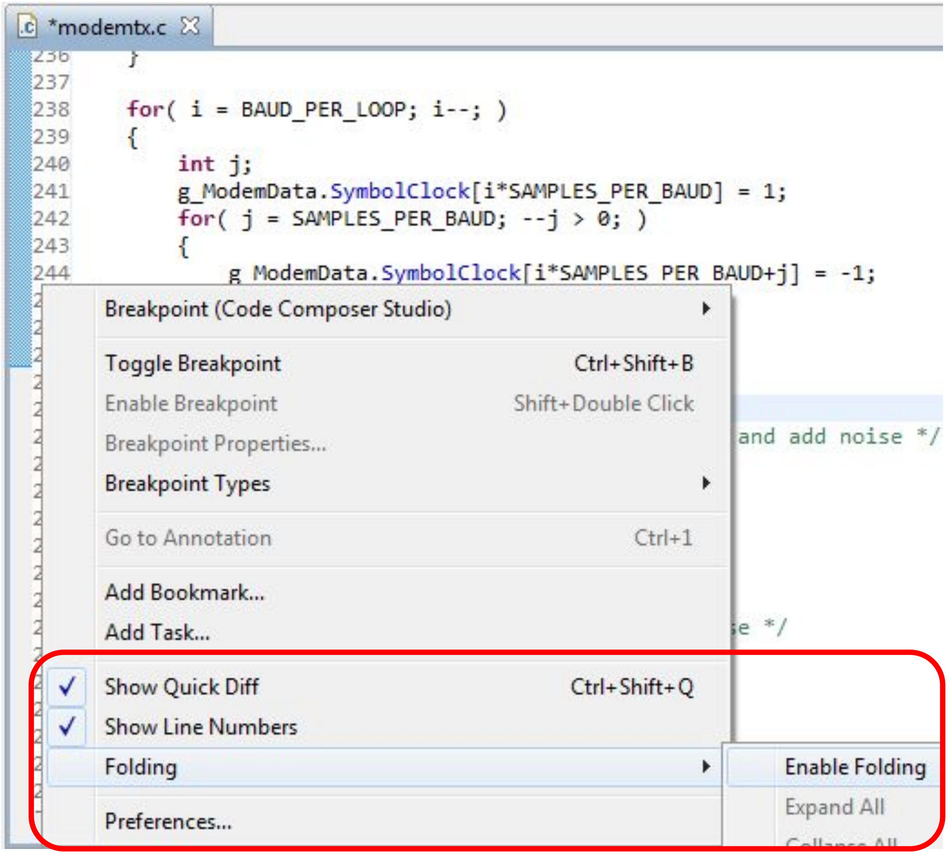
- Idelay : int [255]
- OutputBuffer : int [32]
- Qdelay : int [255]
- SymbolClock : int [32]
- cNoise : POINT [1]
- cPoints : POINT [1]
- carrierFreq : int
- dataSymbols : int [1]
- noiseLevel : int
- phase : int
- samplesPerBaud : int



```
*****************************************************************/
int SineLookup(int sample)

/*****************************************************************
```

Advanced Editor Features

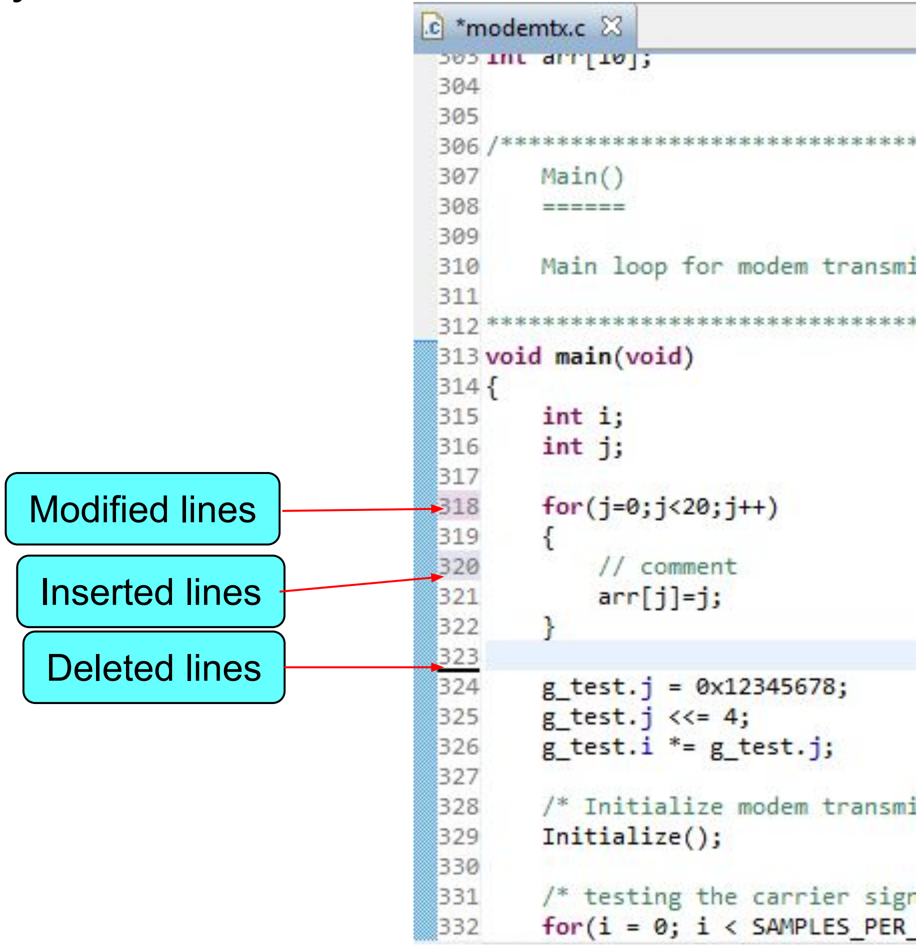
- Right-click in the editor margin to:
  - Toggle line numbers in the editor margin
  - Enable/Disable code **Folding**
  - Enable/Disable **Quick Diff**
  - Open editor **Preferences…** to access more options to configure:
    - **Content Assist** (Code Completion)
    - **Folding**
    - **Syntax Coloring**
    - **Hovers** (Cursor "hover over" behavior)
    - **Typing** behavior
    - etc



```
    }

    for( i = BAUD_PER_LOOP; i--; )
    {
        int j;
        g_ModemData.SymbolClock[i*SAMPLES_PER_BAUD] = 1;
        for( j = SAMPLES_PER_BAUD; --j > 0; )
        {
            g_ModemData.SymbolClock[i*SAMPLES_PER_BAUD+j] = -1;
```

Edit Markers

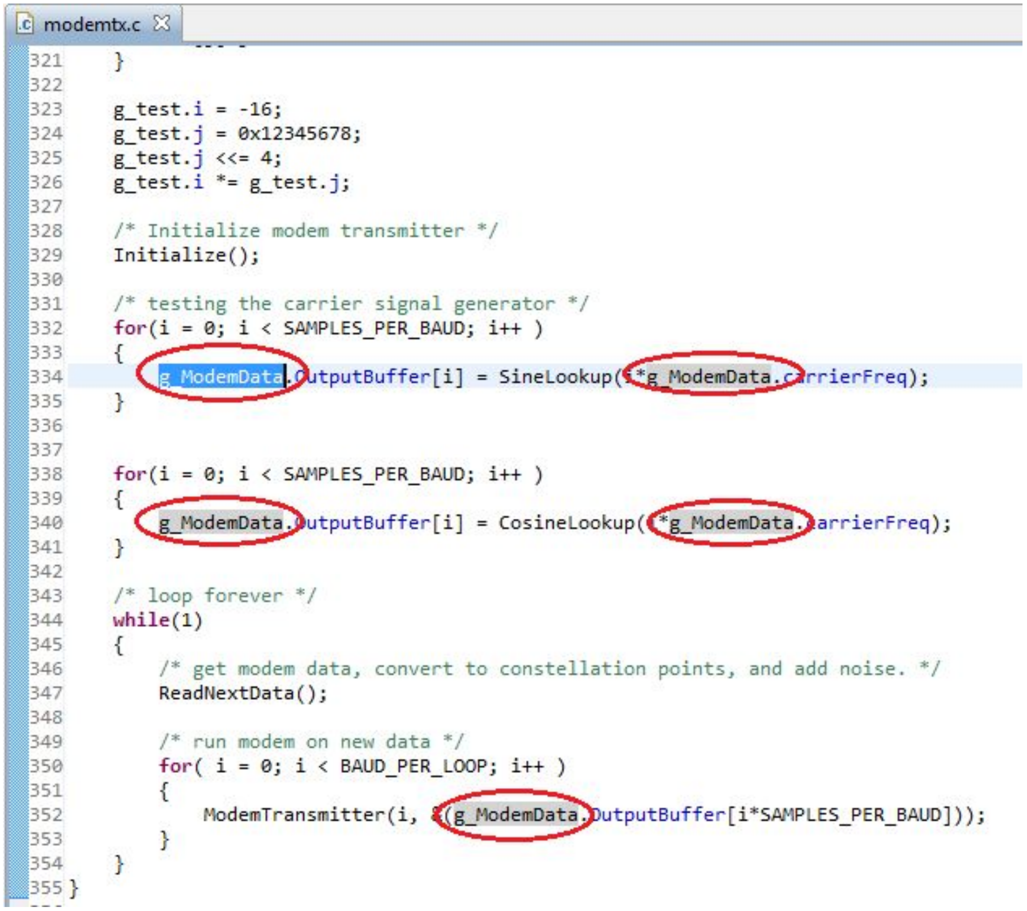
- If you have the line number column on, it also indicates changes in your source file since your last save



```
303 int arr[10];
304
305
306 /*****************************
307     Main()
308     ======
309
310     Main loop for modem transmi
311
312 *****************************
313 void main(void)
314 {
315     int i;
316     int j;
317
318     for(j=0;j<20;j++)
319     {
320         // comment
321         arr[j]=j;
322     }
323
324     g_test.j = 0x12345678;
325     g_test.j <<= 4;
326     g_test.i *= g_test.j;
327
328     /* Initialize modem transmi
329     Initialize();
330
331     /* testing the carrier sigr
332     for(i = 0; i < SAMPLES_PER_
```

Variable Highlighting

- Highlighting a variable in the editor will highlight all instances of the variable in the editor

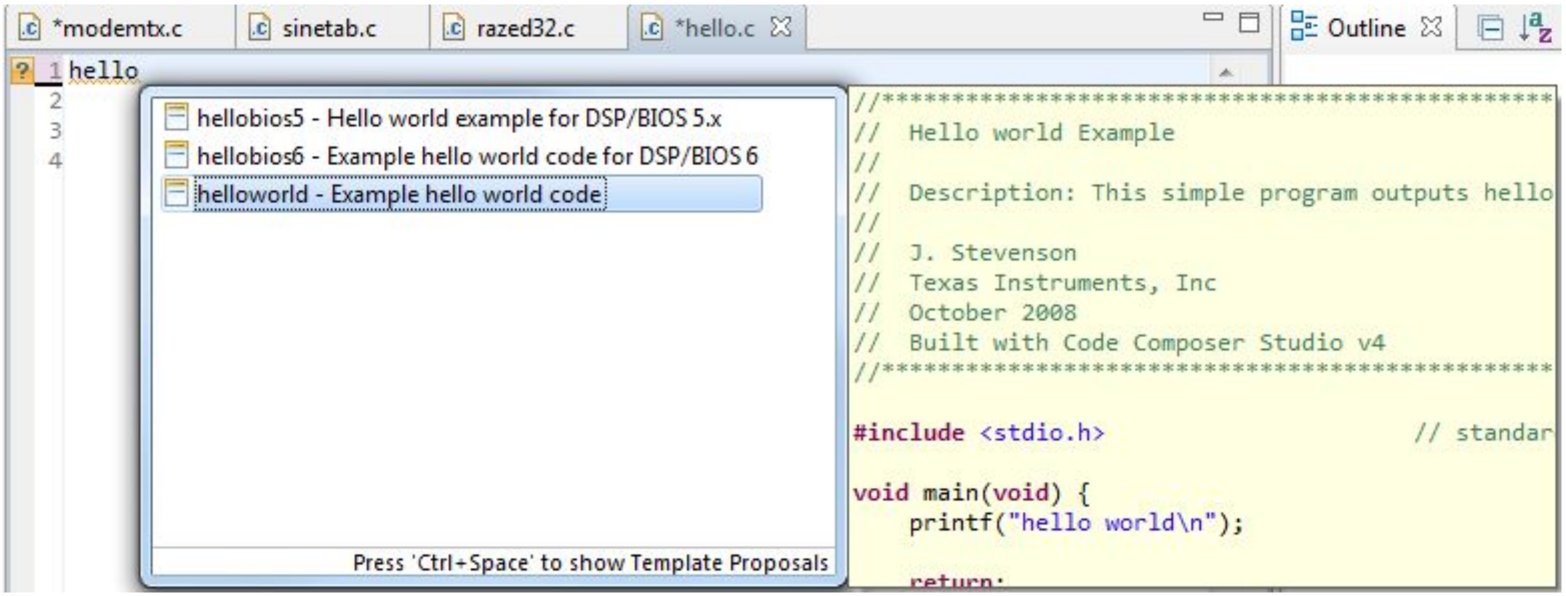


modemtx.c

```
    }

    g_test.i = -16;
    g_test.j = 0x12345678;
    g_test.j <<= 4;
    g_test.i *= g_test.j;

    /* Initialize modem transmitter */
    Initialize();

    /* testing the carrier signal generator */
    for(i = 0; i < SAMPLES_PER_BAUD; i++ )
    {
        g_ModemData.OutputBuffer[i] = SineLookup(&(*g_ModemData.carrierFreq);
    }


    for(i = 0; i < SAMPLES_PER_BAUD; i++ )
    {
        g_ModemData.OutputBuffer[i] = CosineLookup(&(*g_ModemData.carrierFreq);
    }

    /* loop forever */
    while(1)
    {
        /* get modem data, convert to constellation points, and add noise. */
        ReadNextData();

        /* run modem on new data */
        for( i = 0; i < BAUD_PER_LOOP; i++ )
        {
            ModemTransmitter(i, &(g_ModemData.OutputBuffer[i*SAMPLES_PER_BAUD]));
        }
    }
}
```

Source Templates

- CCS provides code templates
  - Ex: Hello World
    - Type in **h** in the editor and use **Content Assist** by pressing **CTRL+SPACE** keys (can also right-click in the editor and select **Content Assist** from the context menu)
  - Create custom templates for commonly used source code blocks or customize existing templates
    - *Window->Preferences…->C++->Editor->Templates*



```
hello
```

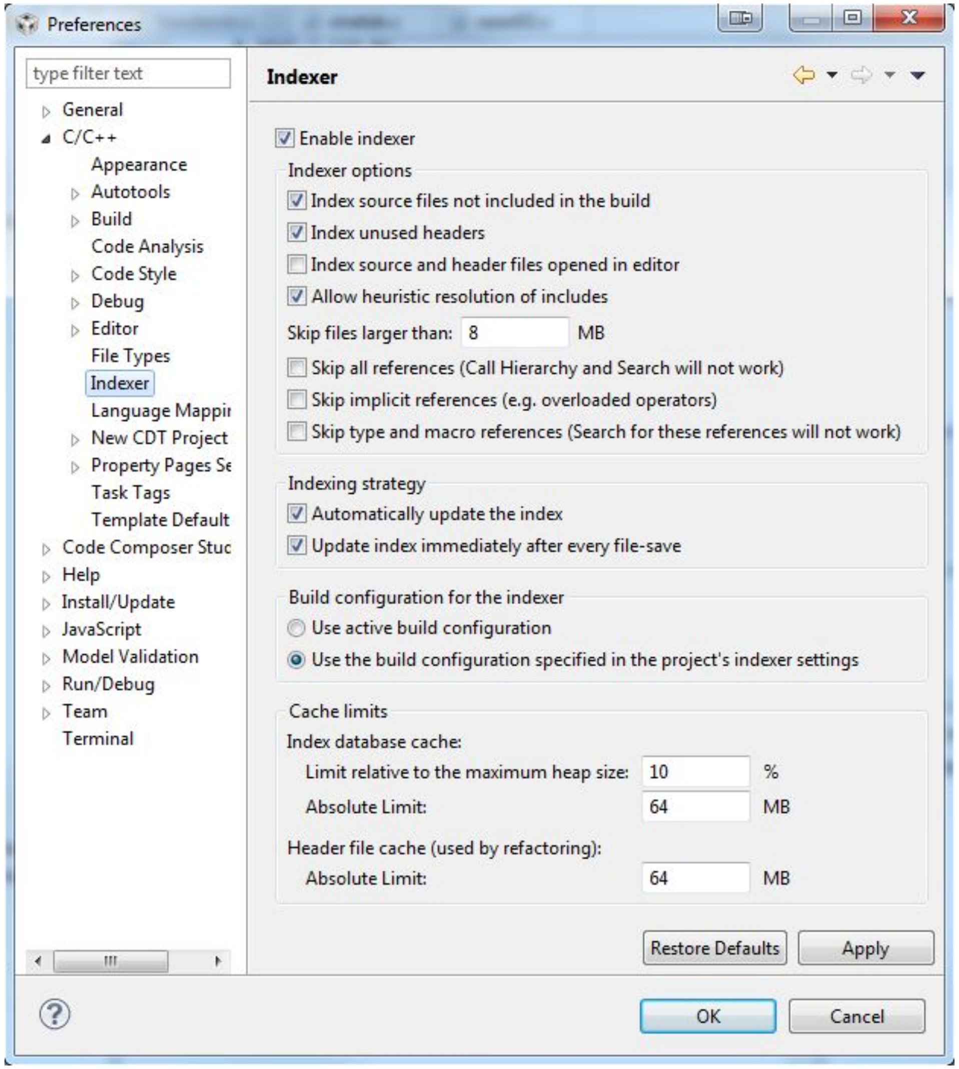
hellobios5 - Hello world example for DSP/BIOS 5.x
hellobios6 - Example hello world code for DSP/BIOS 6
helloworld - Example hello world code

Press 'Ctrl+Space' to show Template Proposals

```
//*********************************************
//  Hello world Example
//
//  Description: This simple program outputs hello
//
//  J. Stevenson
//  Texas Instruments, Inc
//  October 2008
//  Built with Code Composer Studio v4
//*********************************************

#include <stdio.h>                        // standar

void main(void) {
    printf("hello world\n");

    return;
```

Indexer

- The advanced editor features rely on a database of the source and header files of the project that provides the basis for C/C++ search, navigation features and parts of content assist (code completion)
- The C/C++ **Indexer** creates this database by parsing all of the source and header files of the projects open in the workspace
- Configure the **Indexer**:
  - *Window -> Preferences -> C/C++ -> Indexer*



Indexer

✓ Enable indexer

Indexer options

✓ Index source files not included in the build
✓ Index unused headers
☐ Index source and header files opened in editor
✓ Allow heuristic resolution of includes

Skip files larger than: 8 MB

☐ Skip all references (Call Hierarchy and Search will not work)
☐ Skip implicit references (e.g. overloaded operators)
☐ Skip type and macro references (Search for these references will not work)

Indexing strategy

✓ Automatically update the index
✓ Update index immediately after every file-save

Build configuration for the indexer

○ Use active build configuration
● Use the build configuration specified in the project's indexer settings

Cache limits

Index database cache:

Limit relative to the maximum heap size: 10 %

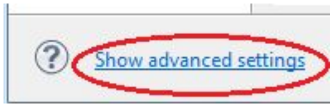
Absolute Limit: 64 MB

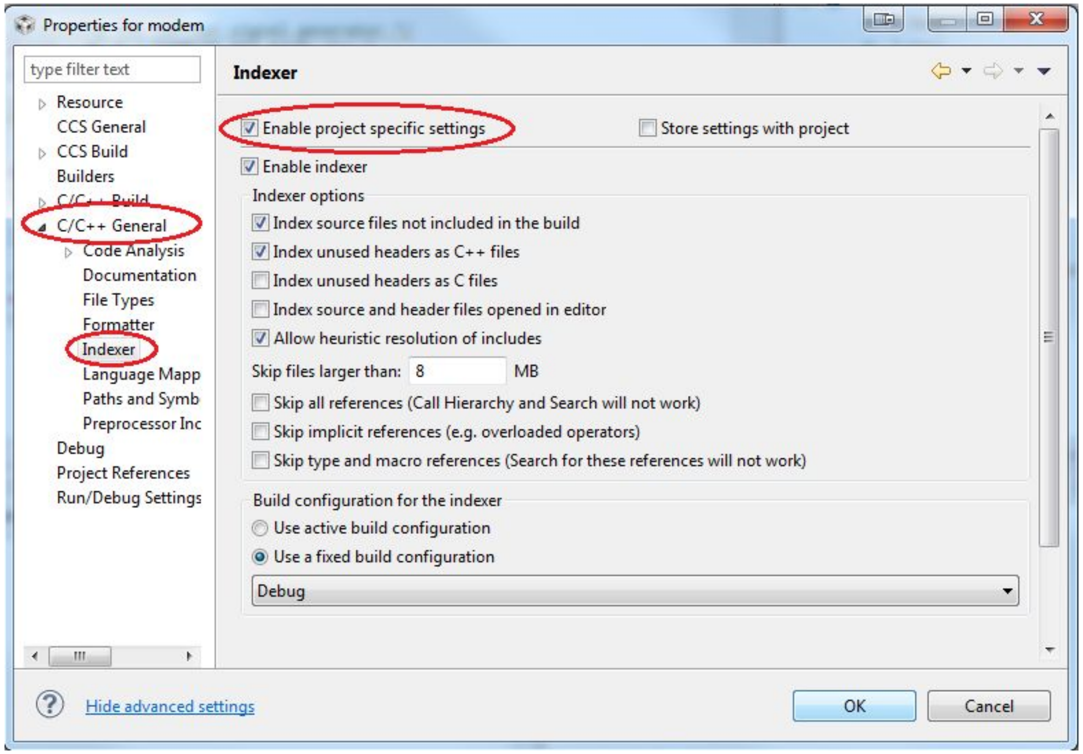
Header file cache (used by refactoring):

Absolute Limit: 64 MB

Indexer

- The Indexer can also be configured on a per project basis in the project properties
  - Must click on the Show Advanced Settings link in the lower left corner of the project properties dialog to expose the options for the Indexer
  - *C/C++ General -> Indexer*





Properties for modem

Indexer

[✓] Enable project specific settings
[ ] Store settings with project

[✓] Enable indexer

Indexer options

[✓] Index source files not included in the build
[✓] Index unused headers as C++ files
[ ] Index unused headers as C files
[ ] Index source and header files opened in editor
[✓] Allow heuristic resolution of includes

Skip files larger than: 8 MB

[ ] Skip all references (Call Hierarchy and Search will not work)
[ ] Skip implicit references (e.g. overloaded operators)
[ ] Skip type and macro references (Search for these references will not work)

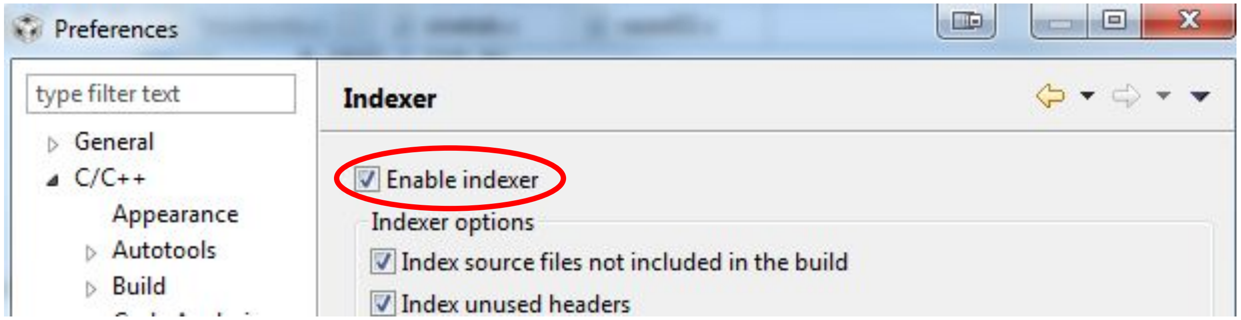
Build configuration for the indexer

( ) Use active build configuration
(●) Use a fixed build configuration

Debug

Performance Tip: Turn off the *Indexer*

- Don't use the CCS editor or don't need the advanced editor features?
- Turn off the Indexer!
  - The indexer constantly scans all open projects to support some advanced editor features
  - The indexer can use a decent amount of system resources, causing CCS to appear sluggish
    - This is most evident with large projects or many open projects in the workspace (or both)
  - The default CCS setting is to have the indexer enabled



☑ Enable indexer

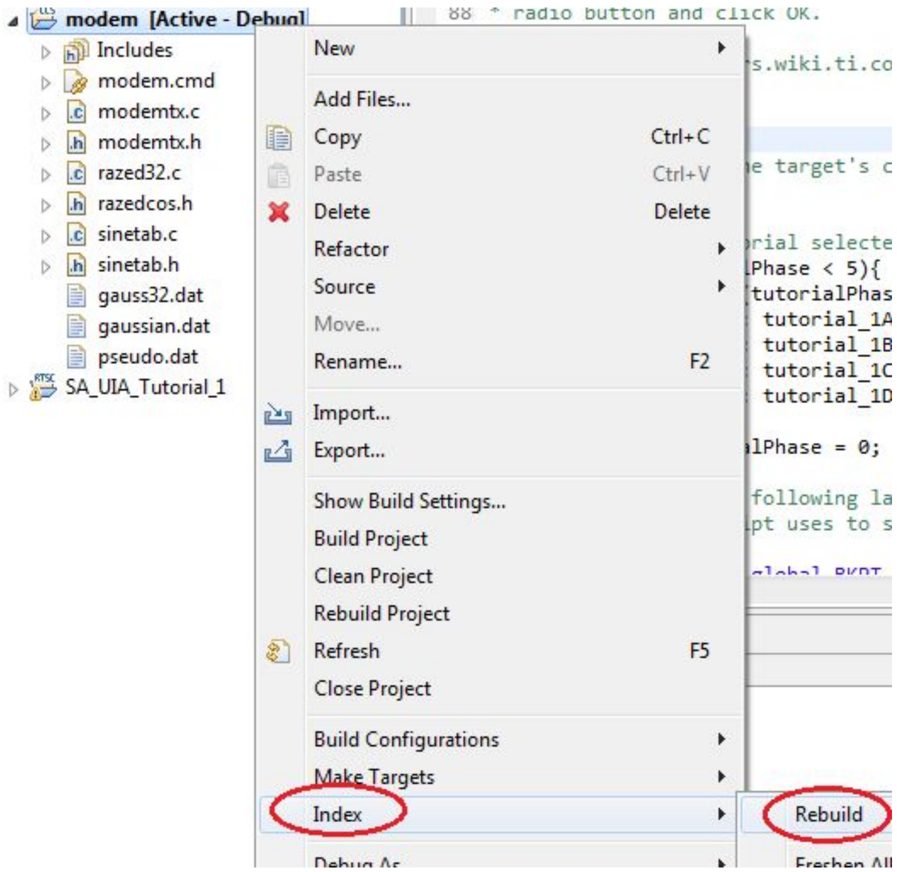
Indexer options

☑ Index source files not included in the build

☑ Index unused headers

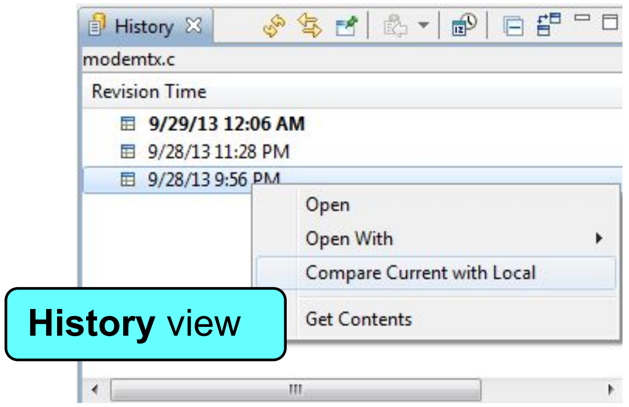
Troubleshooting Tip: Rebuild the *Index*

- Advanced editor feature not working right?
  - Code completion not working or bringing up the wrong suggestions?
  - Open declaration not finding the declaration?
  - Outline/Hierarchy views showing incorrect information?
- Indexed database/cache may have gotten corrupted or is out of date!
- Rebuild it!
  - Right-click on a project and select *Index -> Rebuild* to rebuild the indexed database for that project



View: History

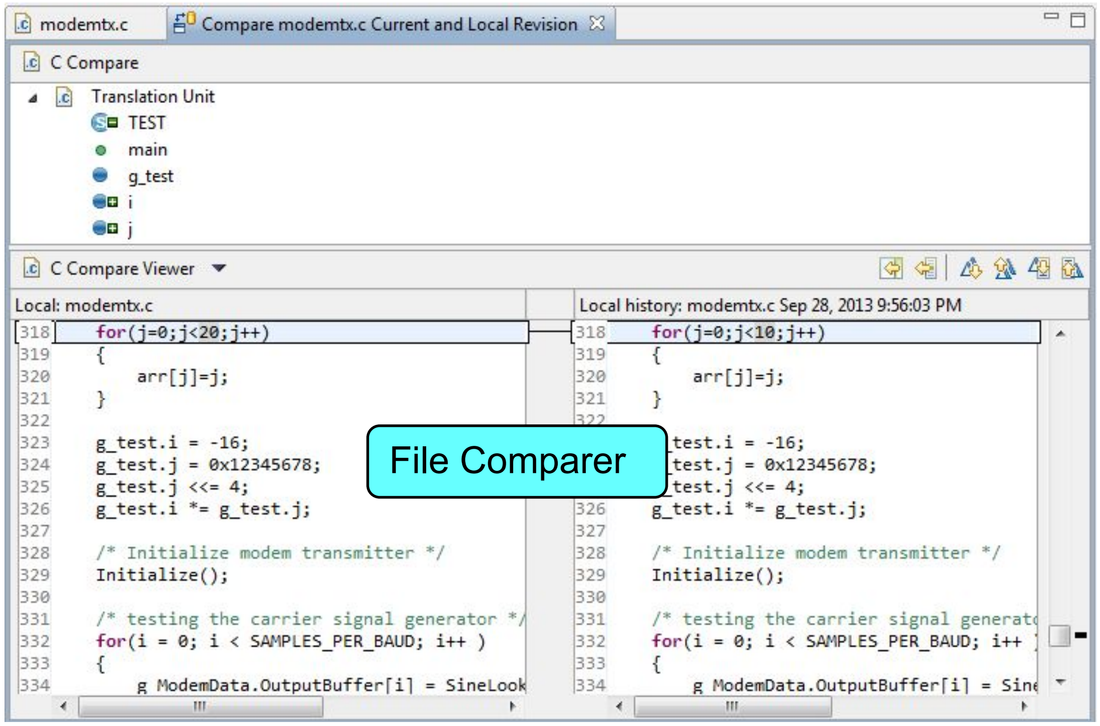
- CCS keeps a local history of source changes
  - Switch to the **CCS Edit** perspective
  - Right-click on a file in the editor an select *Team -> Show Local History*
    - Opens **History** view



History
modemtx.c
Revision Time
9/29/13 12:06 AM
9/28/13 11:28 PM
9/28/13 9:56 PM

Open
Open With
Compare Current with Local
Get Contents

- Use the **History** view to compare the current source file against any previous version or replace it with any previous version
  - Double-click on a revision to open it in the editor
  - Right-click on a revision to compare that revision to the current version



modemtx.c | Compare modemtx.c Current and Local Revision

C Compare
Translation Unit
TEST
main
g_test
i
j

C Compare Viewer

Local: modemtx.c

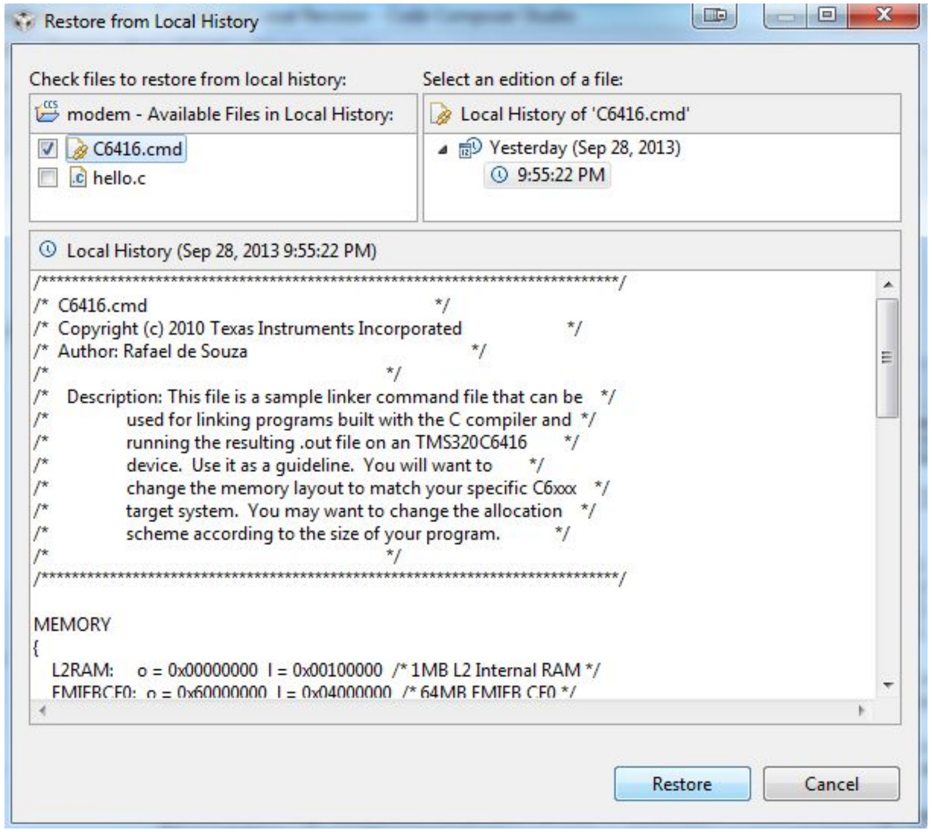
```
    for(j=0;j<20;j++)
    {
        arr[j]=j;
    }

    g_test.i = -16;
    g_test.j = 0x12345678;
    g_test.j <<= 4;
    g_test.i *= g_test.j;

    /* Initialize modem transmitter */
    Initialize();

    /* testing the carrier signal generator */
    for(i = 0; i < SAMPLES_PER_BAUD; i++ )
    {
        g_ModemData.OutputBuffer[i] = SineLook
```

Local history: modemtx.c Sep 28, 2013 9:56:03 PM

```
    for(j=0;j<10;j++)
    {
        arr[j]=j;
    }

    test.i = -16;
    test.j = 0x12345678;
    test.j <<= 4;
    g_test.i *= g_test.j;

    /* Initialize modem transmitter */
    Initialize();

    /* testing the carrier signal generat
    for(i = 0; i < SAMPLES_PER_BAUD; i++
    {
        g_ModemData.OutputBuffer[i] = Sine
```

Local History - Project

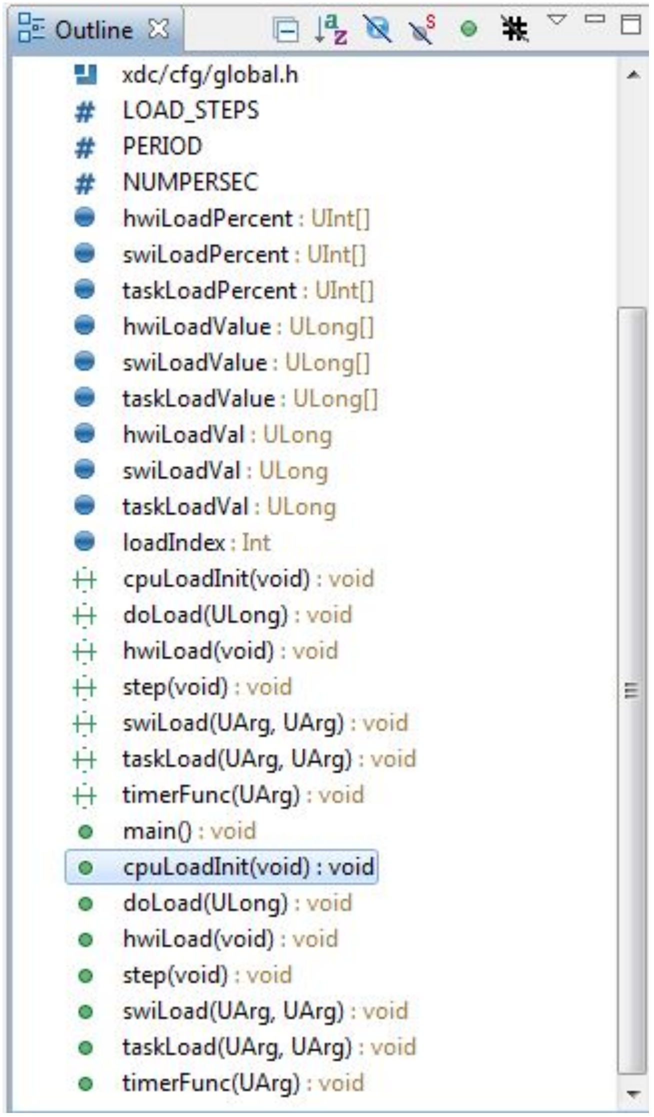
- CCS keeps a local history of files for the project
  - Recover files deleted from the project
  - Right-click on the project and select **Recover from Local History** in the context menu


Restore from Local History
Check files to restore from local history:
Select an edition of a file:
modem - Available Files in Local History:
C6416.cmd
hello.c
Local History of 'C6416.cmd'
Yesterday (Sep 28, 2013)
9:55:22 PM
Local History (Sep 28, 2013 9:55:22 PM)
Restore
Cancel

```
/****************************************************************************/
/*  C6416.cmd                                                               */
/*  Copyright (c) 2010 Texas Instruments Incorporated                       */
/*  Author: Rafael de Souza                                                 */
/*                                                                          */
/*    Description: This file is a sample linker command file that can be   */
/*               used for linking programs built with the C compiler and    */
/*               running the resulting .out file on an TMS320C6416          */
/*               device.  Use it as a guideline.  You will want to          */
/*               change the memory layout to match your specific C6xxx      */
/*               target system.  You may want to change the allocation      */
/*               scheme according to the size of your program.              */
/*                                                                          */
/****************************************************************************/

MEMORY
{
   L2RAM:    o = 0x00000000  l = 0x00100000  /* 1MB L2 Internal RAM */
   EMIFBCE0:  o = 0x60000000  l = 0x04000000  /* 64MB EMIFB CE0 */
```

View: Outline

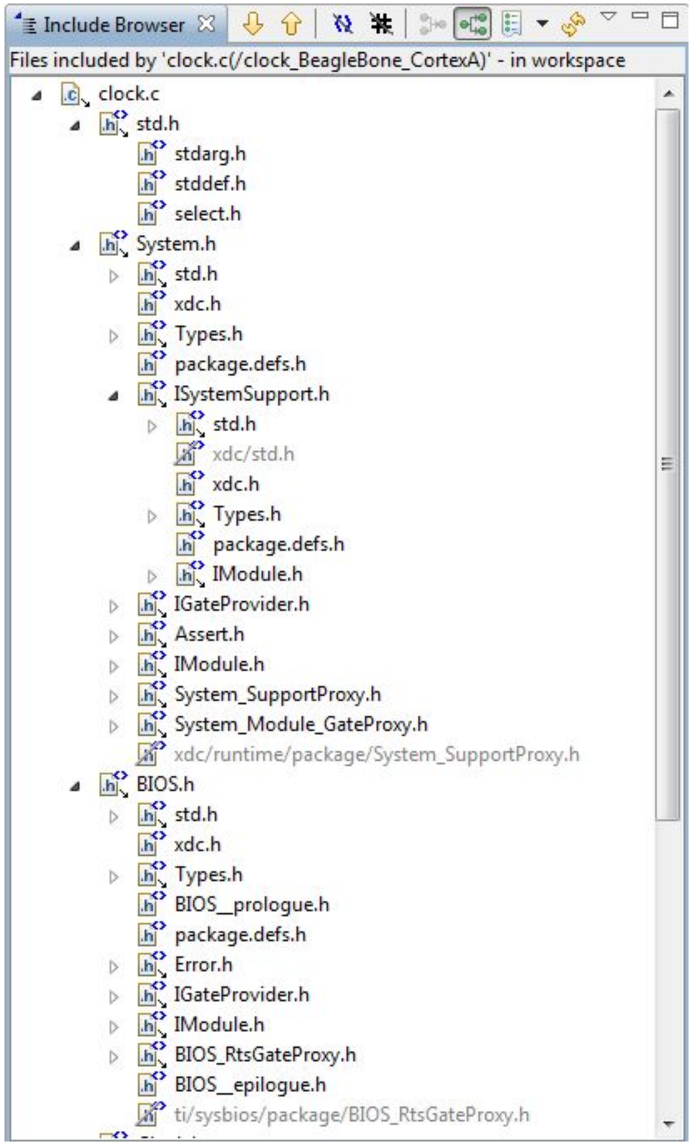
- Displays an outline of a structured file that is currently open in the editor area, and lists structural elements
- *View -> Outline*



```
xdc/cfg/global.h
LOADSTEPS
PERIOD
NUMPERSEC
hwiLoadPercent : UInt[]
swiLoadPercent : UInt[]
taskLoadPercent : UInt[]
hwiLoadValue : ULong[]
swiLoadValue : ULong[]
taskLoadValue : ULong[]
hwiLoadVal : ULong
swiLoadVal : ULong
taskLoadVal : ULong
loadIndex : Int
cpuLoadInit(void) : void
doLoad(ULong) : void
hwiLoad(void) : void
step(void) : void
swiLoad(UArg, UArg) : void
taskLoad(UArg, UArg) : void
timerFunc(UArg) : void
main() : void
cpuLoadInit(void) : void
doLoad(ULong) : void
hwiLoad(void) : void
step(void) : void
swiLoad(UArg, UArg) : void
taskLoad(UArg, UArg) : void
timerFunc(UArg) : void
```

View: Include Browser

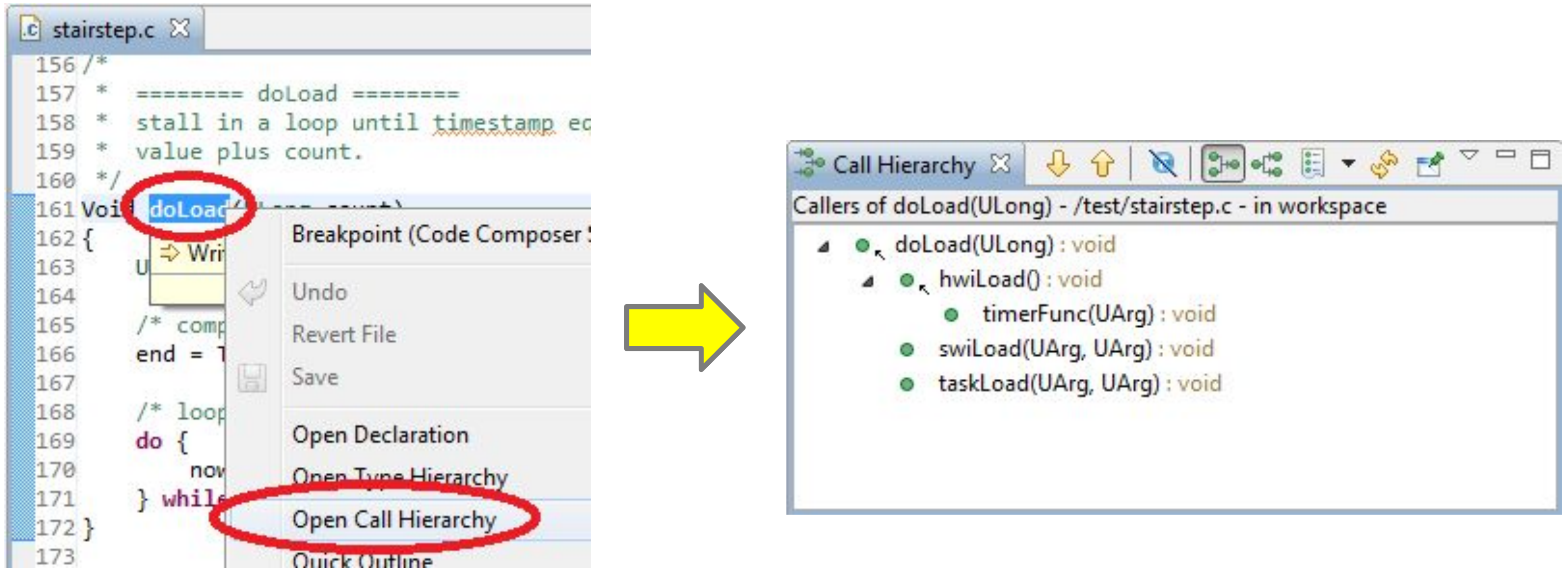
- *Which header files is this source file including? Who is including this header file? Directly? Indirectly?*
- Show hierarchy of included header files for a source file
  - *View -> Other… -> C/C++ -> Include Browser*



```
Files included by 'clock.c(/clock_BeagleBone_CortexA)' - in workspace
clock.c
  std.h
    stdarg.h
    stddef.h
    select.h
  System.h
    std.h
    xdc.h
    Types.h
    package.defs.h
    ISystemSupport.h
      std.h
      xdc/std.h
      xdc.h
      Types.h
      package.defs.h
      IModule.h
    IGateProvider.h
    Assert.h
    IModule.h
    System_SupportProxy.h
    System_Module_GateProxy.h
    xdc/runtime/package/System_SupportProxy.h
  BIOS.h
    std.h
    xdc.h
    Types.h
    BIOS__prologue.h
    package.defs.h
    Error.h
    IGateProvider.h
    IModule.h
    BIOS_RtsGateProxy.h
    BIOS__epilogue.h
    ti/sysbios/package/BIOS_RtsGateProxy.h
```

View: Call Hierarchy

- Displays callers and callees for a selected function
- Right-click on a function and then select **Open Call Hierarchy** in the context menu



```
/*
 *  ======== doLoad ========
 *  stall in a loop until timestamp ec
 *  value plus count.
 */
Voi  doLoad
{
    U

    /* comp
    end = T

    /* loop
    do {
        now
    } whil
}
```

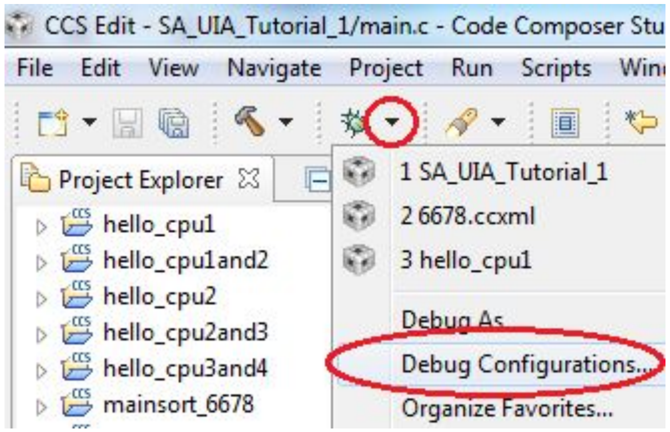
Callers of doLoad(ULong) - /test/stairstep.c - in workspace

```
◢ ● doLoad(ULong) : void
    ◢ ● hwiLoad() : void
          ● timerFunc(UArg) : void
      ● swiLoad(UArg, UArg) : void
      ● taskLoad(UArg, UArg) : void
```

DEBUGGING

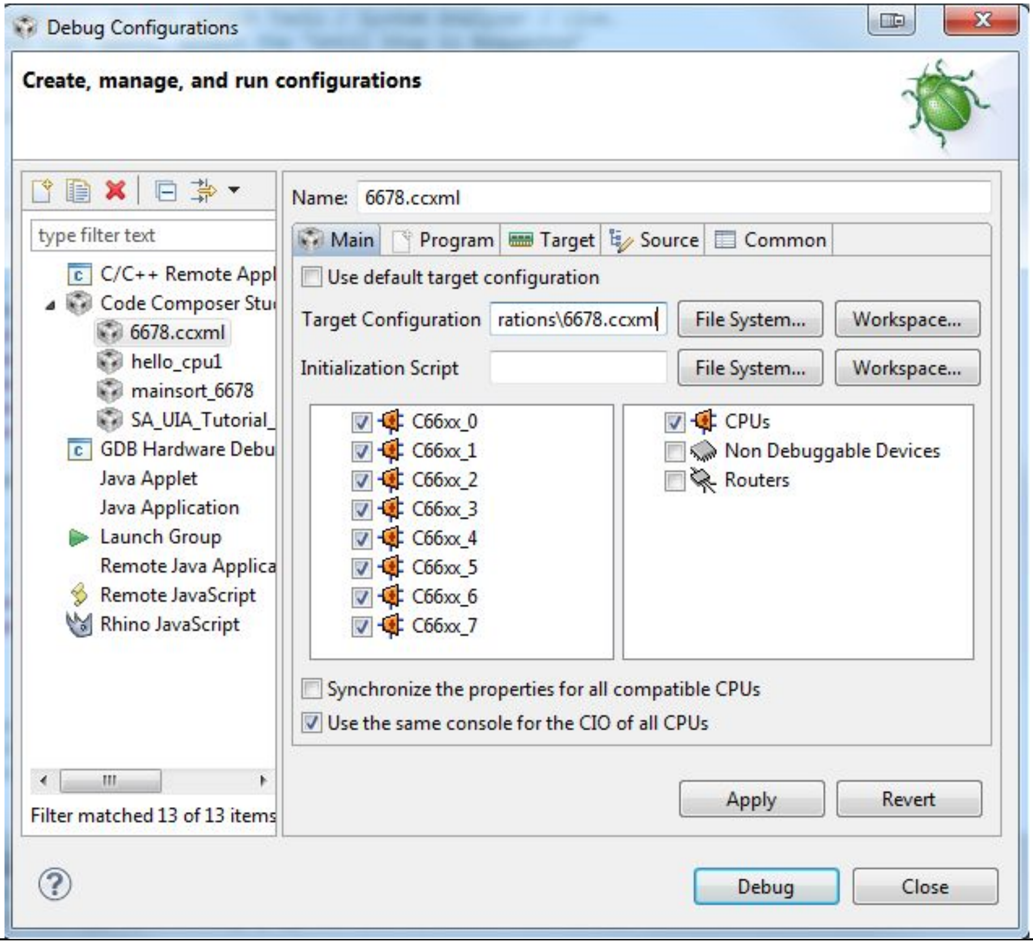
Debug Configurations

- Debug information created when a debug session is first launched for a project or target configuration
  - Information stored includes which target configuration to use, debug settings…
- Both project and project-less debug sessions launch debug sessions using a debug configuration
  - A debug session can be started by explicitly launching a debug configuration
- To configure: Use drop-down menu next to the **Debug As** button and select **Debug Configurations…**



Debug Configurations

- Interface to manage existing configuration or to create new ones
- Existing debug configurations are configurable


Debug Configurations

Create, manage, and run configurations

type filter text

- C/C++ Remote Appl
- Code Composer Stu
  - 6678.ccxml
  - hello_cpu1
  - mainsort_6678
  - SA_UIA_Tutorial_
- GDB Hardware Debu
- Java Applet
- Java Application
- Launch Group
- Remote Java Applica
- Remote JavaScript
- Rhino JavaScript

Filter matched 13 of 13 items

Name: 6678.ccxml

Main | Program | Target | Source | Common

☐ Use default target configuration

Target Configuration: rations\6678.ccxml

Initialization Script:

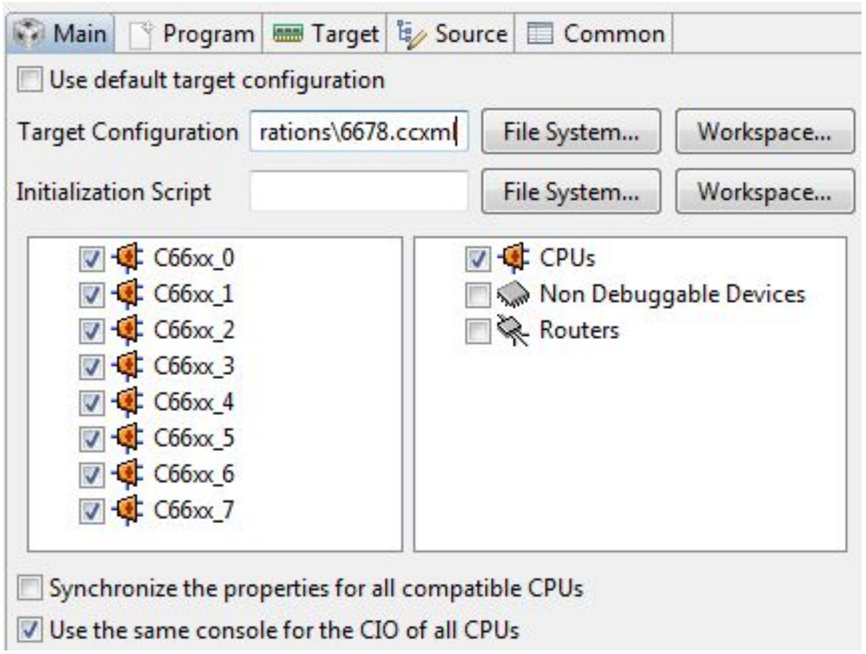
☑ C66xx_0
☑ C66xx_1
☑ C66xx_2
☑ C66xx_3
☑ C66xx_4
☑ C66xx_5
☑ C66xx_6
☑ C66xx_7

☑ CPUs
☐ Non Debuggable Devices
☐ Routers

☐ Synchronize the properties for all compatible CPUs
☑ Use the same console for the CIO of all CPUs

Debug Configurations – Main Options

- Use the **Main** tab to:
  - Which target configuration to use
  - Use the **Initialization Script** field to specify a DSS JavaScript for target initialization
  - Specify which devices on the JTAG scan chain will be visible by in the **Debug** view by default
  - Specify if all CPUs share the same console for C I/O (for multi-core debug)
    - C I/O will be interleaved in the same console (preceded with the CPU name)
    - Uncheck the option to create a separate C I/O console for each CPU



☐ Use default target configuration

Target Configuration: rations\6678.ccxml

Initialization Script:

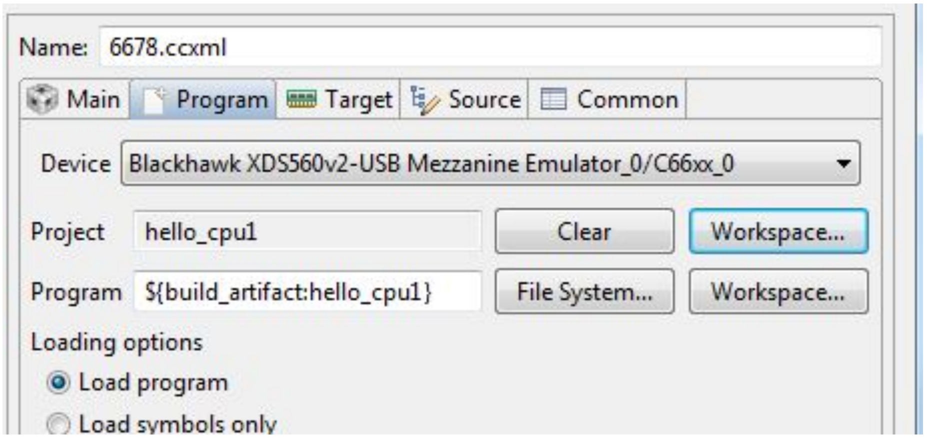
✓ C66xx_0	✓ CPUs
✓ C66xx_1	☐ Non Debuggable Devices
✓ C66xx_2	☐ Routers
✓ C66xx_3	
✓ C66xx_4	
✓ C66xx_5	
✓ C66xx_6	
✓ C66xx_7	

☐ Synchronize the properties for all compatible CPUs

✓ Use the same console for the CIO of all CPUs

Debug Configurations – Program Options

- Use the **Program** tab to:
  - Specify which CPU to load the executable on (for multi-core devices)
    - Can specify different programs for each CPU
  - Specify to load the program (default) or just symbols only (to debug code in flash, etc)



Name: 6678.ccxml

Device: Blackhawk XDS560v2-USB Mezzanine Emulator_0/C66xx_0

Project: hello_cpu1

Program: ${build_artifact:hello_cpu1}

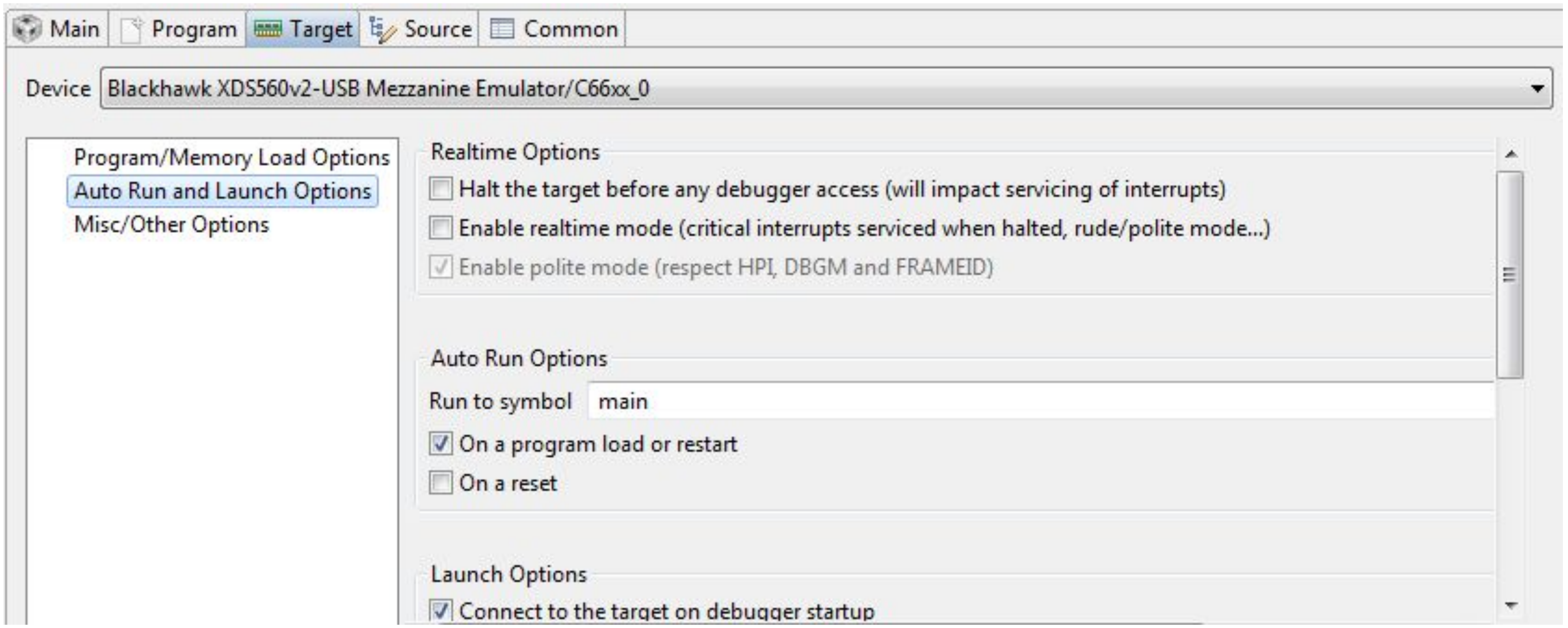
Loading options

[✓ Load program]

Load symbols only

Debug Configurations – Target Options

- The **Target** tab can be used to set a variety of debug options like auto-run to *main*, auto-connect to a HW target, real-time options, program verification on load, etc…
- **Flash Programmer** options are available for applicable devices



Device: Blackhawk XDS560v2-USB Mezzanine Emulator/C66xx_0

Program/Memory Load Options
Auto Run and Launch Options
Misc/Other Options

Realtime Options
- [ ] Halt the target before any debugger access (will impact servicing of interrupts)
- [ ] Enable realtime mode (critical interrupts serviced when halted, rude/polite mode...)
- [x] Enable polite mode (respect HPI, DBGM and FRAMEID)

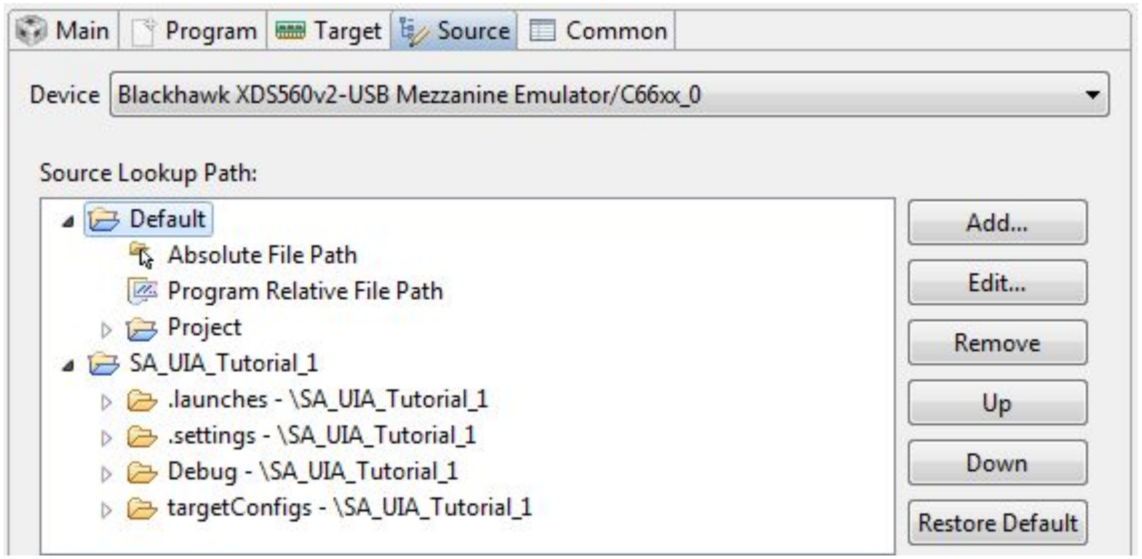
Auto Run Options

Run to symbol: main
- [x] On a program load or restart
- [ ] On a reset

Launch Options
- [x] Connect to the target on debugger startup

Debug Configurations – Source Options

- The **Source** tab allows you to add additional source lookup search paths
  - Specify paths on a per CPU granularity (for multi-core devices)
  - All paths to any source files in your project are automatically added by default



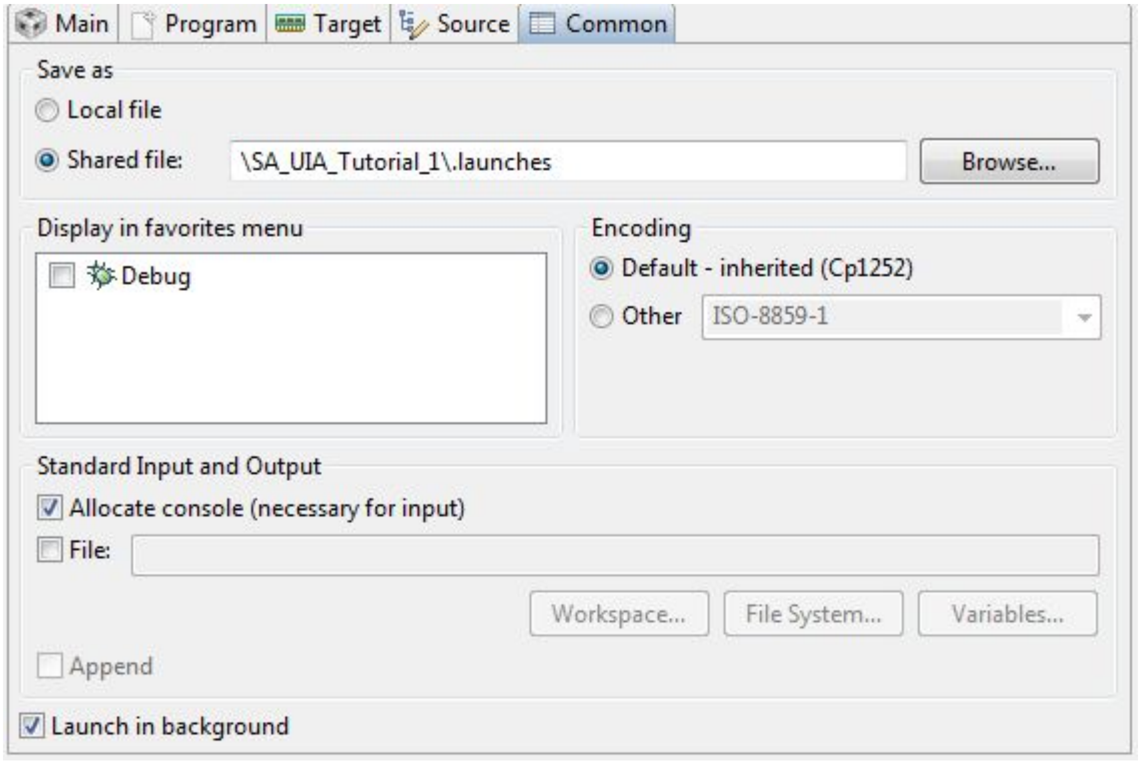
Device: Blackhawk XDS560v2-USB Mezzanine Emulator/C66xx_0

Source Lookup Path:

- Default
  - Absolute File Path
  - Program Relative File Path
  - Project
- SA_UIA_Tutorial_1
  - .launches - \SA_UIA_Tutorial_1
  - .settings - \SA_UIA_Tutorial_1
  - Debug - \SA_UIA_Tutorial_1
  - targetConfigs - \SA_UIA_Tutorial_1

Debug Configurations – Common Options

- The **Common** tab contains a collection of miscellaneous options
  - Can specify the debugger to send all CIO to a file instead of the console
  - Specify character encoding



Save as
( ) Local file
(●) Shared file: \SA_UIA_Tutorial_1\.launches

Display in favorites menu
☐ Debug

Encoding
(●) Default - inherited (Cp1252)
( ) Other ISO-8859-1

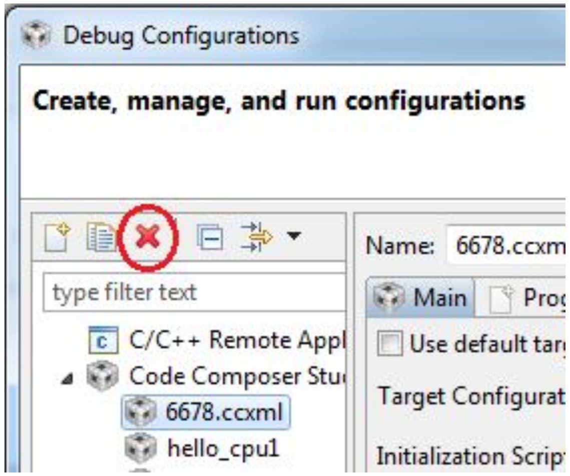
Standard Input and Output
☑ Allocate console (necessary for input)
☐ File:

☐ Append

☑ Launch in background

Troubleshooting Tip: Debug Configuration

- Having some strange debugger issues?
  - Having problems launching a debug session?
  - Can’t connect to the target anymore?
  - Is the debug session unstable?
- Debug Configuration may have gotten corrupted!
- Delete it and have CCS generate a new one
  - Select the **Debug Configuration** to delete in the **Debug Configurations** dialog and press the delete button



Troubleshooting Tip: Test Connectivity

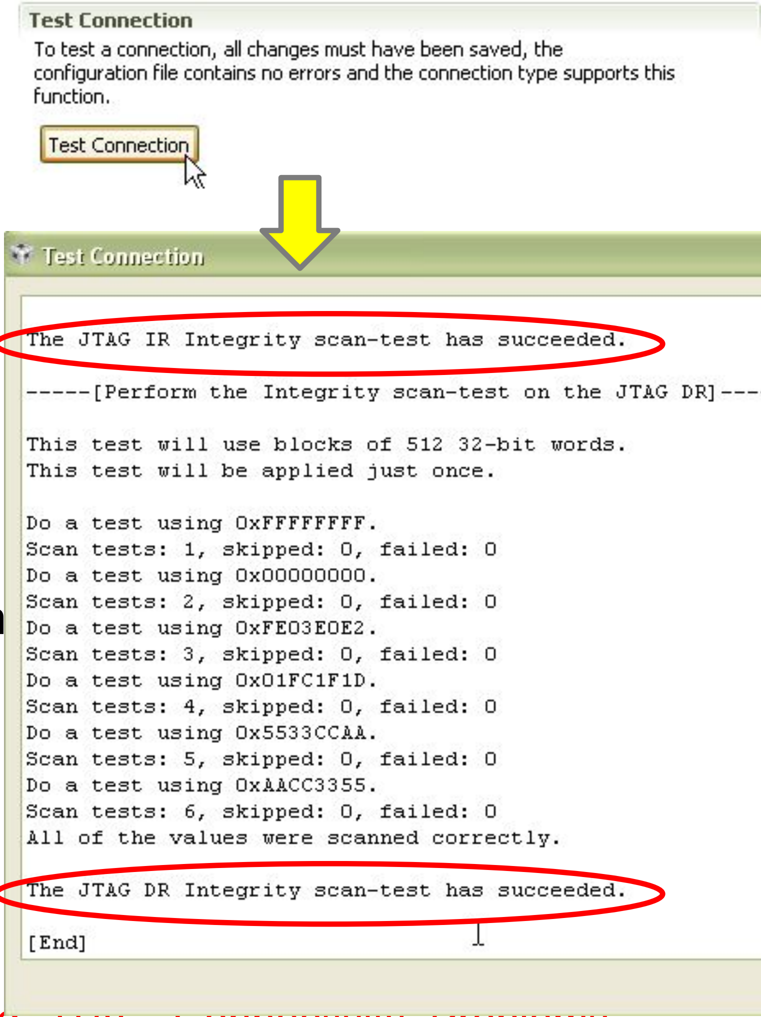
Having target connection issues?

Validate your physical JTAG connection to the device from the driver

- Use the **Test Connection** button in the **Advanced** tab of the target configuration
  - Will test both the target configuration file and the JTAG connectivity between CCS and the target by running some diagnostics
  - If all test pass, then the physical connection between the device and the driver is fine and the issue is with the debugger
  - If there are failures, send the results to TI support

Not all emulators support this feature

- http://processors.wiki.ti.com/index.php/Debugging_JTAG_Connectivity_Problems



Test Connection

To test a connection, all changes must have been saved, the configuration file contains no errors and the connection type supports this function.

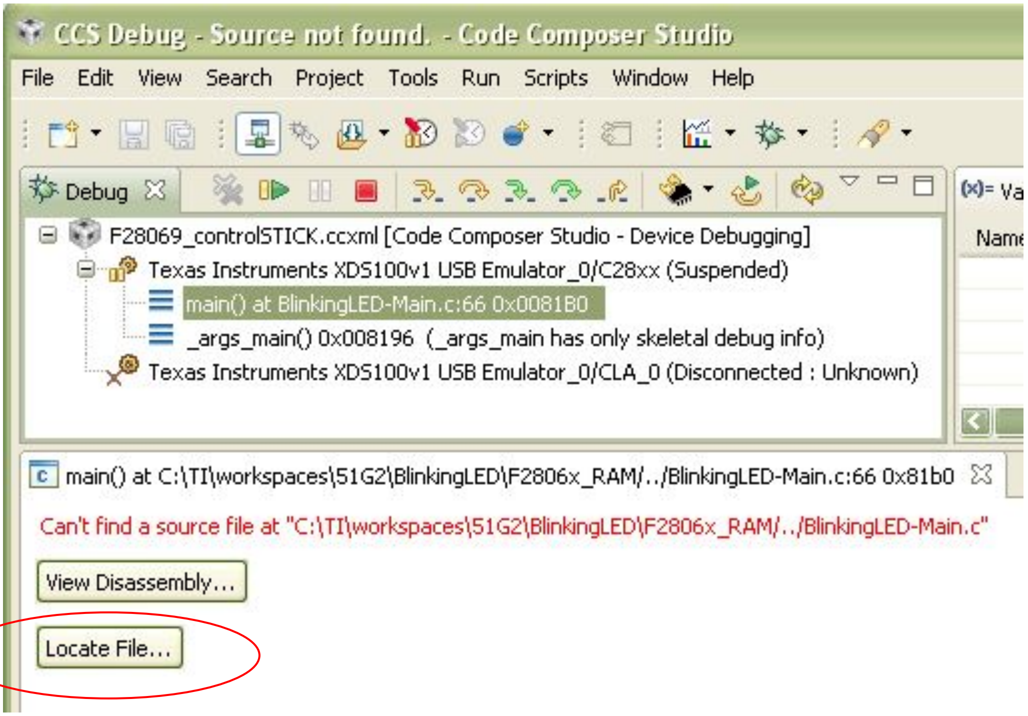
```
The JTAG IR Integrity scan-test has succeeded.

-----[Perform the Integrity scan-test on the JTAG DR]---

This test will use blocks of 512 32-bit words.
This test will be applied just once.

Do a test using 0xFFFFFFFF.
Scan tests: 1, skipped: 0, failed: 0
Do a test using 0x00000000.
Scan tests: 2, skipped: 0, failed: 0
Do a test using 0xFE03E0E2.
Scan tests: 3, skipped: 0, failed: 0
Do a test using 0x01FC1F1D.
Scan tests: 4, skipped: 0, failed: 0
Do a test using 0x5533CCAA.
Scan tests: 5, skipped: 0, failed: 0
Do a test using 0xAACC3355.
Scan tests: 6, skipped: 0, failed: 0
All of the values were scanned correctly.

The JTAG DR Integrity scan-test has succeeded.

[End]
```

Debugging Without a Project

- For project-less debug sessions, CCS will look for source files using relative path information stored in the debug symbols
  - CCS will find the source files if the executable and source files are in the exact same location as when the executable was originally built
- If the location of the executable file or source (or both) has changed, CCS may not be able to find the source files
- CCS can be instructed where to find the source files one of two ways:
  - Tell CCS where the first file is and let CCS find the rest of the files using relative path information in the symbols (recommended method)
  - Set **Source Lookup Paths** for CCS to scan when looking for source files:
    - Set for current debug session
    - Set for **Debug Configuration** - apply for every debug session launched by the debug configuration (under the **Source** options}
    - Set at global (workspace) level – apply for any debug session started with this workspace

Method #1 (Recommended)

- If a source file cannot be found during debug, it will be indicated in the editor
- Use **Locate the Source File…** button to browse to the location of the source file
  - The debugger can then find other source files in the same location or use relative path information to find files relative to the current file
  - Location is remembered for future loads of the same program
- This method has the best performance for finding source files


CCS Debug - Source not found. - Code Composer Studio
File Edit View Search Project Tools Run Scripts Window Help
Debug
Name
View Disassembly...
Locate File...

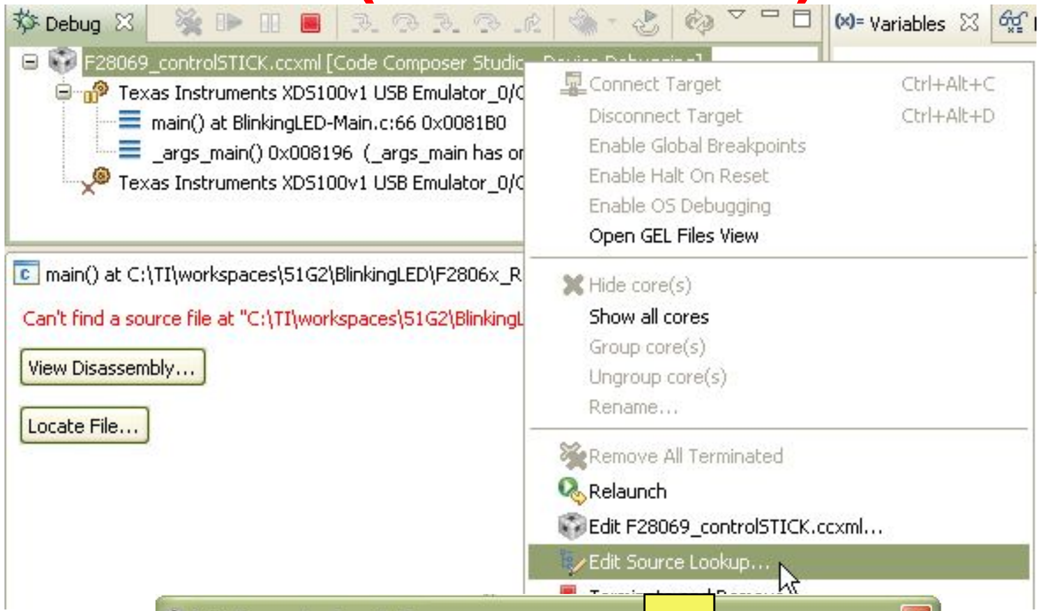
F28069_controlSTICK.ccxml [Code Composer Studio - Device Debugging]
Texas Instruments XDS100v1 USB Emulator_0/C28xx (Suspended)
main() at BlinkingLED-Main.c:66 0x0081B0
_args_main() 0x008196 (_args_main has only skeletal debug info)
Texas Instruments XDS100v1 USB Emulator_0/CLA_0 (Disconnected : Unknown)

main() at C:\TI\workspaces\51G2\BlinkingLED\F2806x_RAM/../BlinkingLED-Main.c:66 0x81b0

Can't find a source file at "C:\TI\workspaces\51G2\BlinkingLED\F2806x_RAM/../BlinkingLED-Main.c"

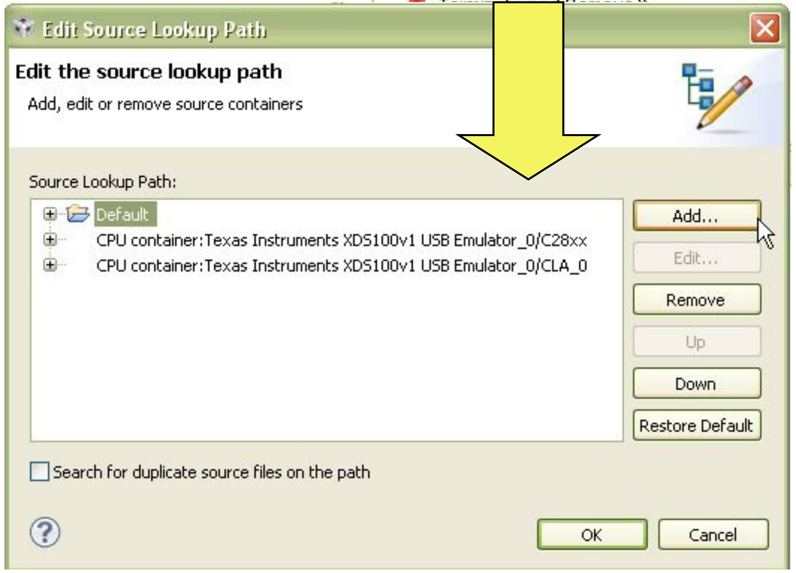
Method #2 – Debug Context (CCSv6.0.x)

- Source lookup paths can also be explicitly specified for each debug context
- Right-click in the **Debug** view and select **Edit Source Lookup…** in the context menu
- To add a file system path, select **File System Directory** to browse to and add paths
- For multi-core debugging, each debug context has its own set of source lookup paths
- **NOTE**: This option is only available for CCSv6.0.x


F28069_controlSTICK.ccxml [Code Composer Studio - Device Debugging]
Texas Instruments XDS100v1 USB Emulator_0/C
main() at BlinkingLED-Main.c:66 0x0081B0
_args_main() 0x008196 (_args_main has or
Texas Instruments XDS100v1 USB Emulator_0/C

main() at C:\TI\workspaces\51G2\BlinkingLED\F2806x_R

Can't find a source file at "C:\TI\workspaces\51G2\BlinkingL


Edit Source Lookup Path

Edit the source lookup path

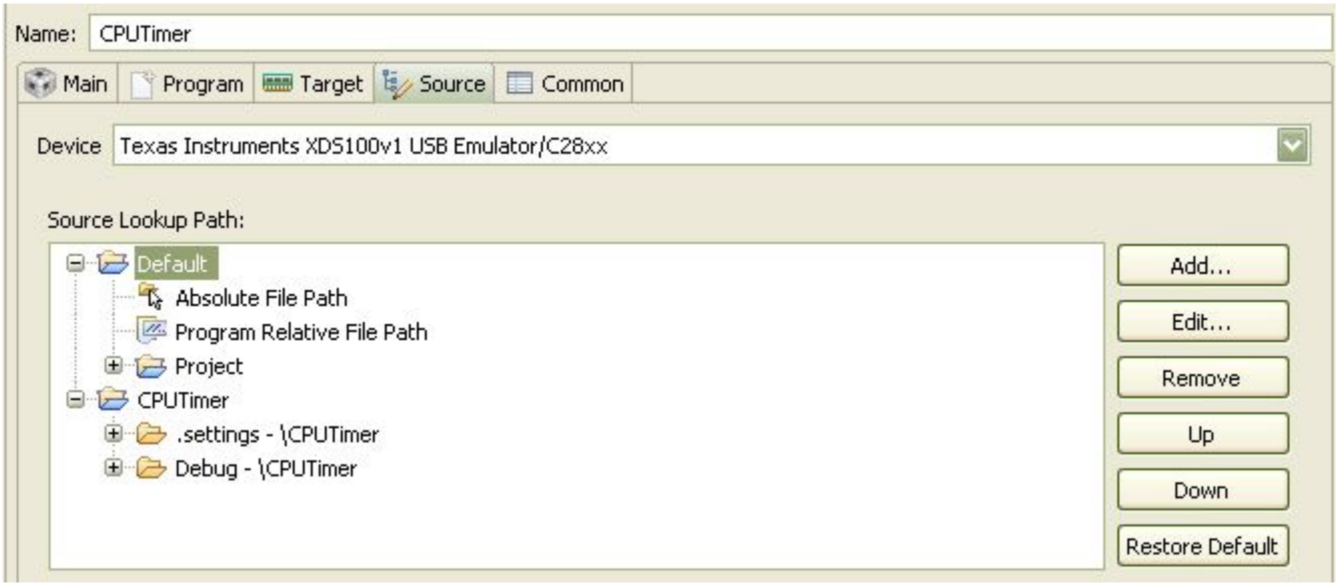
Add, edit or remove source containers

Source Lookup Path:

Default
CPU container:Texas Instruments XDS100v1 USB Emulator_0/C28xx
CPU container:Texas Instruments XDS100v1 USB Emulator_0/CLA_0

Method #3 – Debug Configuration

- The **Source** tab in the debug configuration allows you to add additional source lookup search paths
  - All paths to any source files in your project are automatically added by default



Name: CPUTimer

Device: Texas Instruments XDS100v1 USB Emulator/C28xx

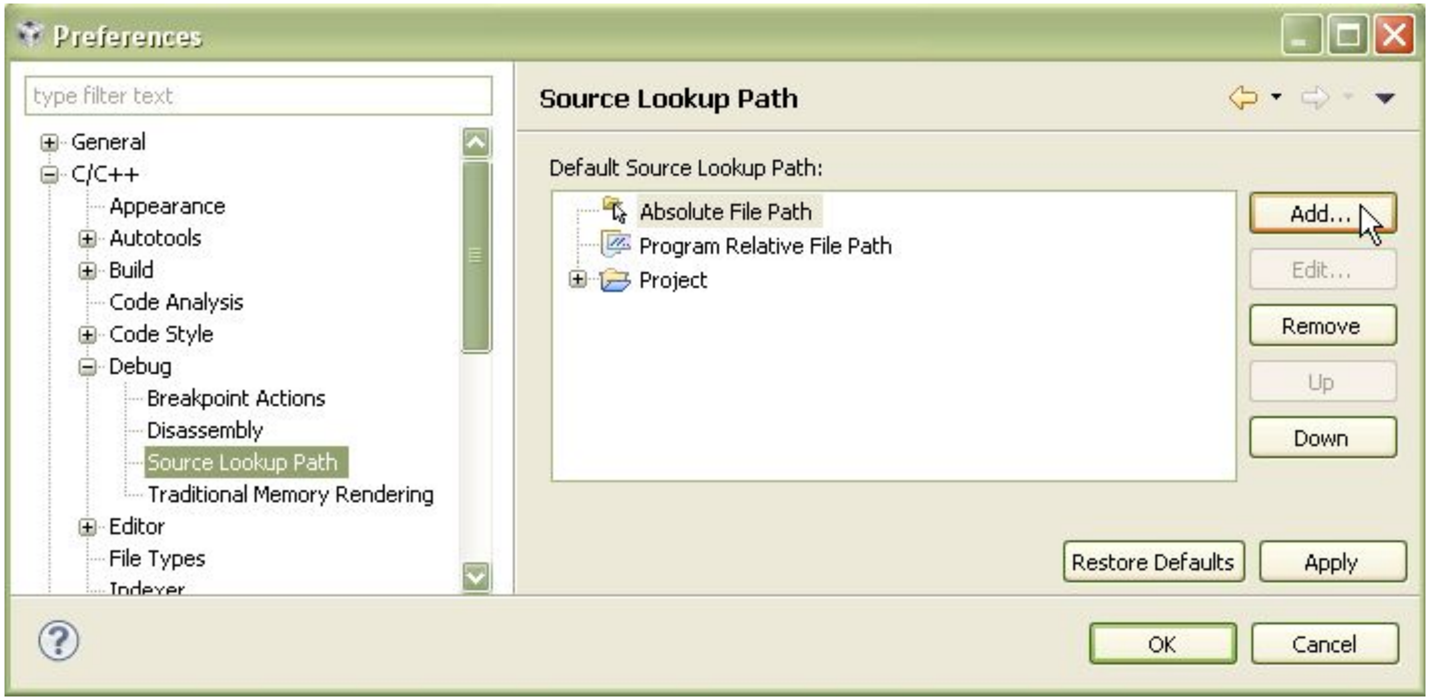
Source Lookup Path:

```
Default
    Absolute File Path
    Program Relative File Path
    Project
CPUTimer
    .settings - \CPUTimer
    Debug - \CPUTimer
```

  - **Debug Configurations** may also be accessed during an active debug session by right-clicking in the debug view and select "Edit <Debug Configuration>“ in the context menu

Method #4 – Global (Workspace)

- Source lookup paths can also be set globally to apply for all debug contexts (in a multi-core environment) and debug sessions
  - *Windows -> Preferences -> C/C++ -> Debug -> Source Lookup Path*



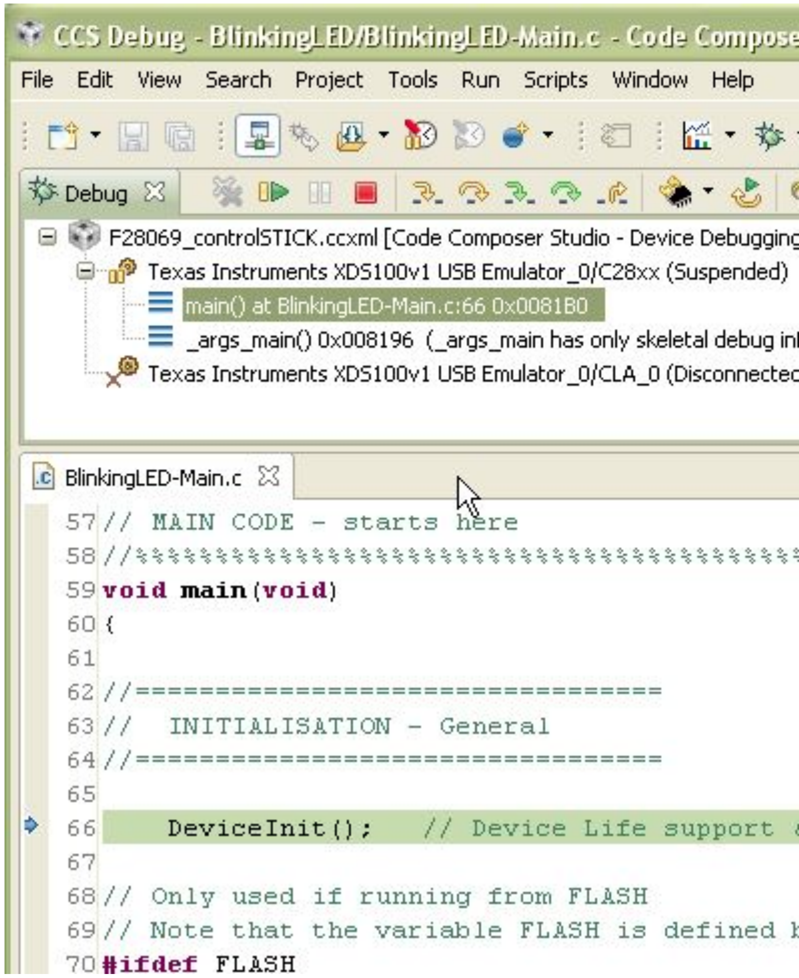
Source Lookup Path

Default Source Lookup Path:

- Absolute File Path
- Program Relative File Path
- Project

More Debugging: Source Lookup Paths

- Once the path is known to the debugger (using any method), the source file will be opened in the debugger
- **Method #1 is recommended due to having the best performance**
- The other methods will do recursive searches inside the specified directories when searching for files. If the directories have many subfolders and many files inside, the search may be slow and thus lead to slow performance when looking for the source files

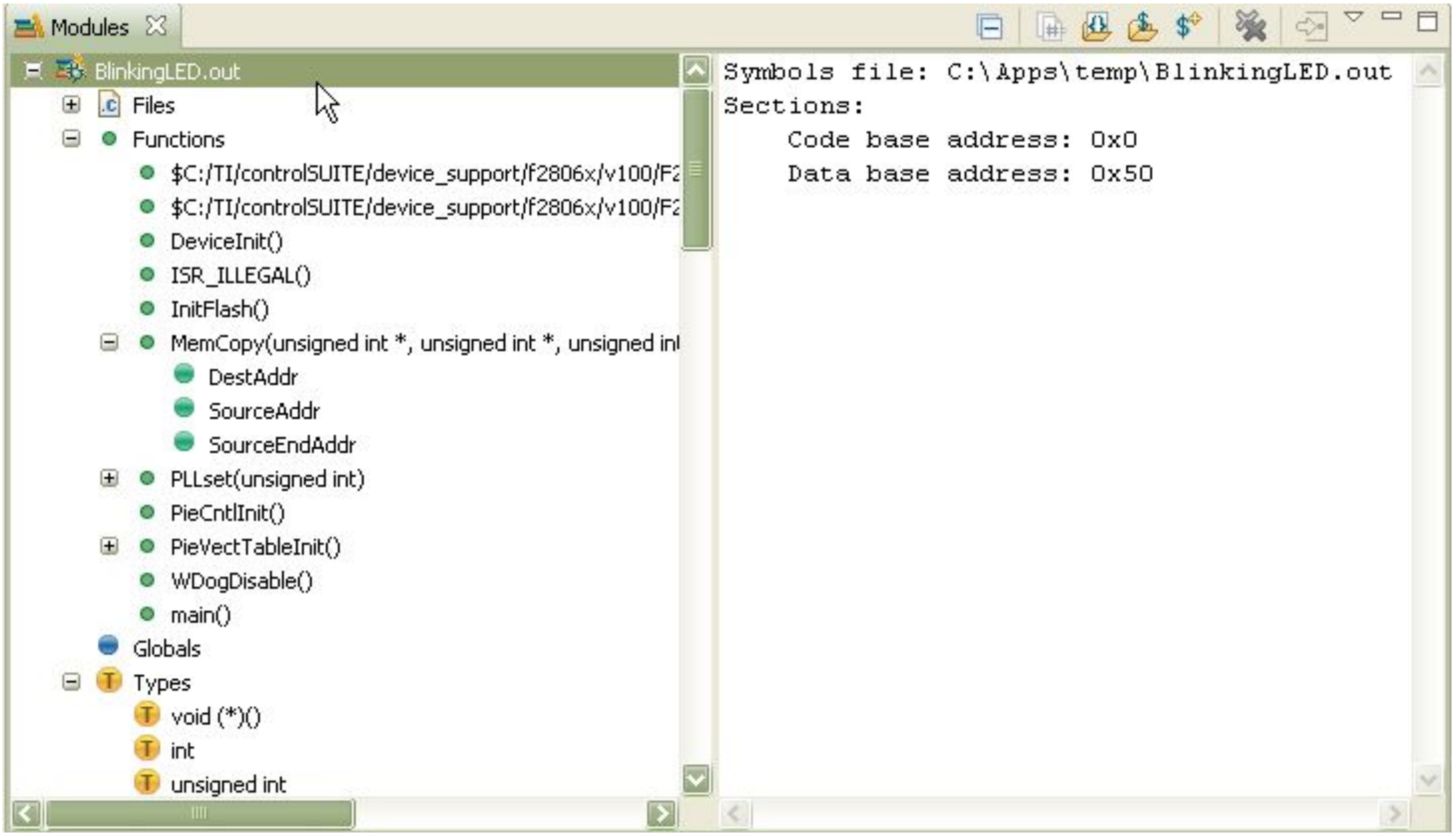

CCS Debug - BlinkingLED/BlinkingLED-Main.c - Code Compose
File Edit View Search Project Tools Run Scripts Window Help
Debug
BlinkingLED-Main.c

F28069_controlSTICK.ccxml [Code Composer Studio - Device Debugging
Texas Instruments XDS100v1 USB Emulator_0/C28xx (Suspended)
main() at BlinkingLED-Main.c:66 0x0081B0
_args_main() 0x008196 (_args_main has only skeletal debug in
Texas Instruments XDS100v1 USB Emulator_0/CLA_0 (Disconnecte

```
// MAIN CODE - starts here
//%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
void main(void)
{

//===================================
//   INITIALISATION - General
//===================================

    DeviceInit();    // Device Life support &

// Only used if running from FLASH
// Note that the variable FLASH is defined b
#ifdef FLASH
```

View: Modules

- Provides information for all loaded symbol files
- *View -> Modules*

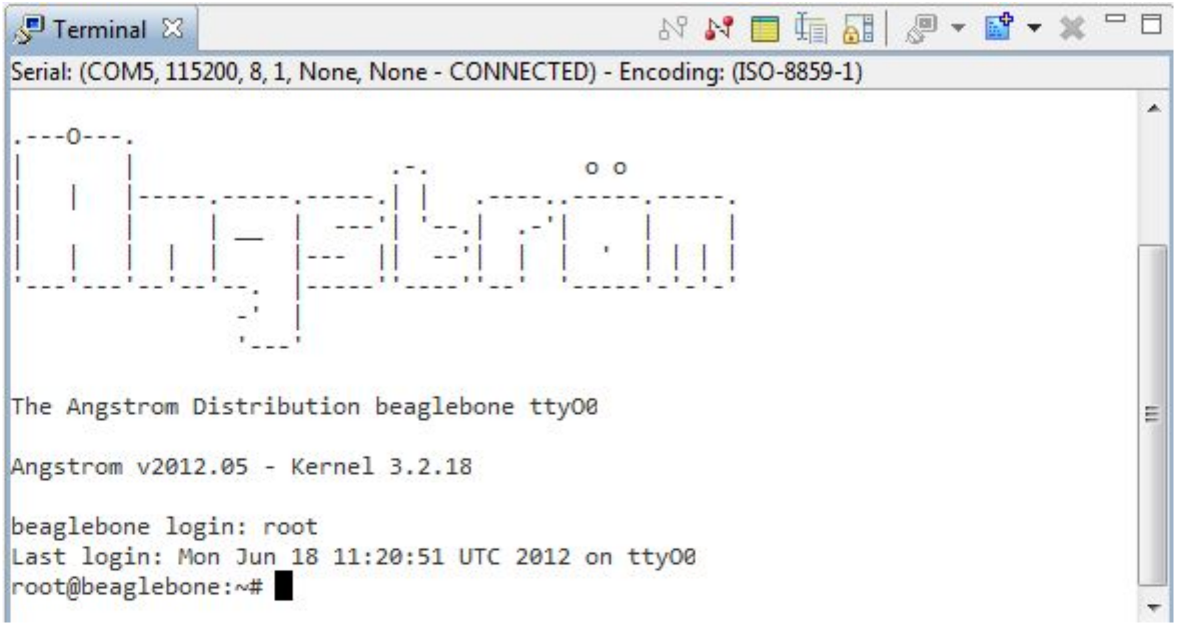


```
BlinkingLED.out
    Files
    Functions
        $C:/TI/controlSUITE/device_support/f2806x/v100/F2
        $C:/TI/controlSUITE/device_support/f2806x/v100/F2
        DeviceInit()
        ISR_ILLEGAL()
        InitFlash()
        MemCopy(unsigned int *, unsigned int *, unsigned in
            DestAddr
            SourceAddr
            SourceEndAddr
        PLLset(unsigned int)
        PieCntlInit()
        PieVectTableInit()
        WDogDisable()
        main()
    Globals
    Types
        void (*)()
        int
        unsigned int
```

```
Symbols file: C:\Apps\temp\BlinkingLED.out
Sections:
    Code base address: 0x0
    Data base address: 0x50
```

View: Terminal

- Terminal emulator that can connect to a remote target via a serial port or over TCP/IP using the TELNET or SSH protocol.
- *View -> Other… -> Terminal -> Terminal*

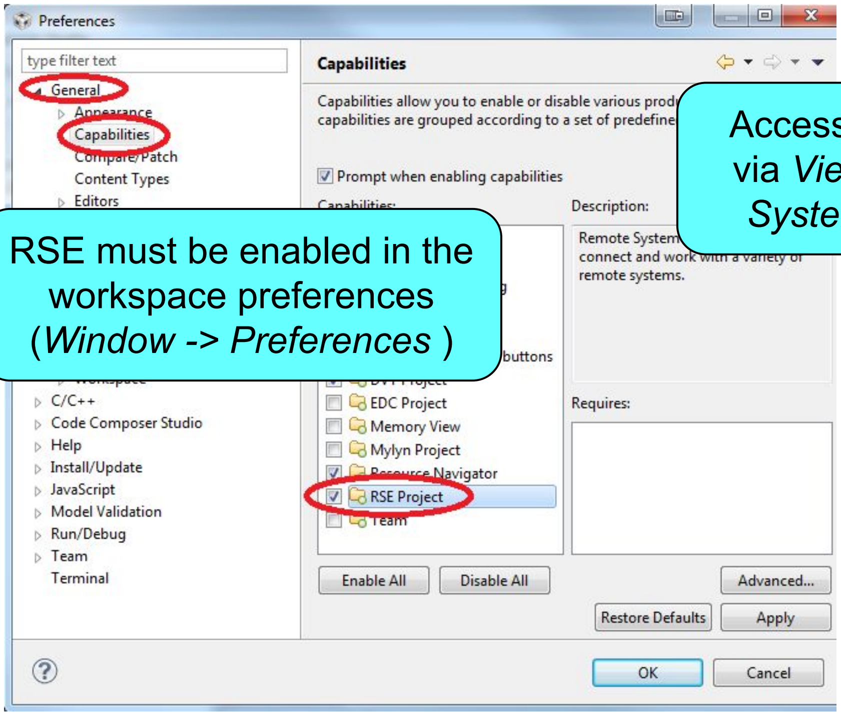


Serial: (COM5, 115200, 8, 1, None, None - CONNECTED) - Encoding: (ISO-8859-1)

```
.---O---.
|       |                  .-.           o o
|   |   |-----.-----.-----.| |   .----..-----.-----.
|       |     | __  |  ---'| '--.|  .-'|     |     |
|   |   |  |  |     |---  ||  --'|  |  |  '  | | | |
'---'---'--'--'--.  |-----''----''--'  '-----'-'-'-'
                -'  |
                '---'

The Angstrom Distribution beaglebone ttyO0

Angstrom v2012.05 - Kernel 3.2.18

beaglebone login: root
Last login: Mon Jun 18 11:20:51 UTC 2012 on ttyO0
root@beaglebone:~# 
```

View: Remote Systems

- Remote Systems Explorer (RSE) is an Eclipse plug-in that provides:
  - Drag-and-drop access to the remote file system
  - Remote shell execution
  - Remote terminal
  - Remote process monitor

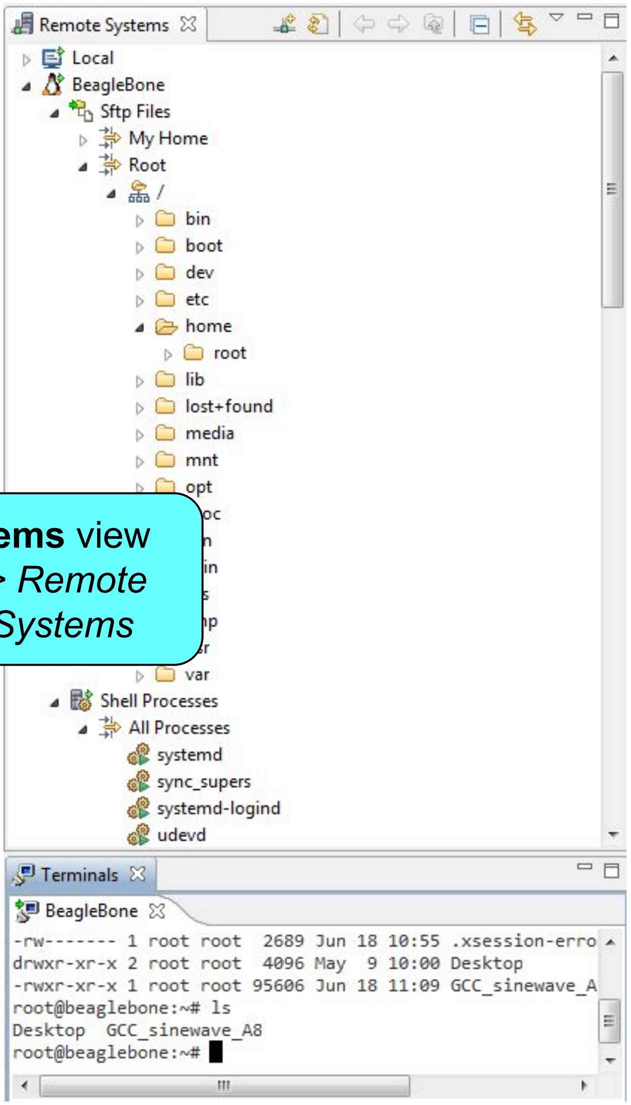


Capabilities allow you to enable or disable various prod
capabilities are grouped according to a set of predefine

Remote System
connect and work with a variety of
remote systems.

RSE must be enabled in the workspace preferences (*Window -> Preferences*)

Access **Remote Systems** view via *View -> Other… -> Remote Systems -> Remote Systems*



```
-rw------- 1 root root  2689 Jun 18 10:55 .xsession-erro
drwxr-xr-x 2 root root  4096 May  9 10:00 Desktop
-rwxr-xr-x 1 root root 95606 Jun 18 11:09 GCC_sinewave_A
root@beaglebone:~# ls
Desktop  GCC_sinewave_A8
root@beaglebone:~# 
```