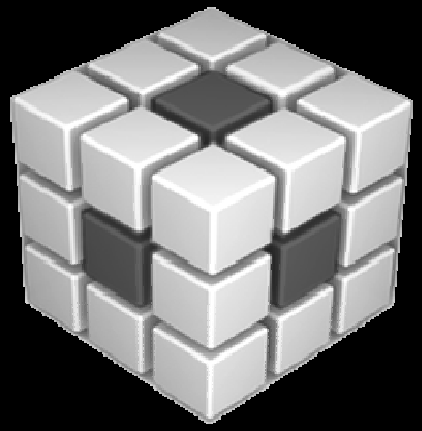
Code Composer Studio™



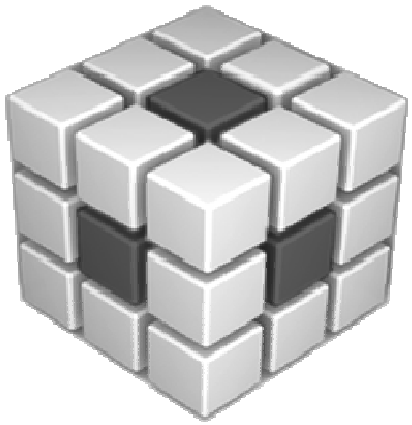
Texas Instruments

Fundamentals Workshop with the MSP430

Agenda

- Overview
- Hands-on workshop with MSP430
  - Getting Started
  - Watchpoints
  - Build properties
  - Debugging programs existing in flash
  - Portable Projects
  - ULP Advisor
  - GRACE basics

Code Composer Studio Overview



Code Composer Studio

- What is Code Composer Studio?
  - Integrated development environment for TI's embedded processors
  - A suite of tools including a debugger, compiler, editor, real-time operating system…
- Based on the Eclipse open source software framework
  - CCS leverages unmodified versions of Eclipse
    - Enables access to the latest Eclipse features
    - Ensures plug-in compatibility
  - TI contributes changes directly to the open source community
  - Extended by TI to support device capabilities
- Integrate additional tools & functionality
  - OS application development tools (Linux, Android…)
  - Code analysis, source control…

User interface makes development intuitive and easy

Simplified user interface	shows features as needed
Resource Explorer	facilitates use of example code
Development tools	for Windows and Linux operating systems
Eclipse open source framework	enables customization via latest plug-ins
SYS/BIOS Real-time Operating System	royalty free high performance RTOS
Training Material	explains how to get the most out of features

SYS/BIOS Real-Time Operating System

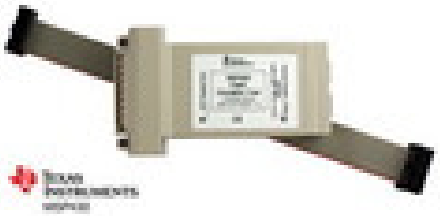
- Open-source RTOS for TI Embedded Processors
  - BSD licensed: all source code included with build support
  - 95% C code
- Designed for real-time applications
  - Scheduler is deterministic so kernel system calls complete operation in a predictable time
  - “Zero-latency Interrupts” enable kernel to be used in hard real-time applications
- Low footprint to meet memory constraints
  - Kernel is highly configurable so unneeded functions are excluded
  - Static configuration enables very low footprints by eliminating need for heaps or create/delete calls if desired
- Proven reliability
  - Latest generation in a family of RTOS products going back 20 years
- Free
  - No development or runtime licensing costs
- Processor support:
  - C6000, C2000, Stellaris, Sitara & MSP430

MSP430 JTAG Emulators

MSP-FET430UIF

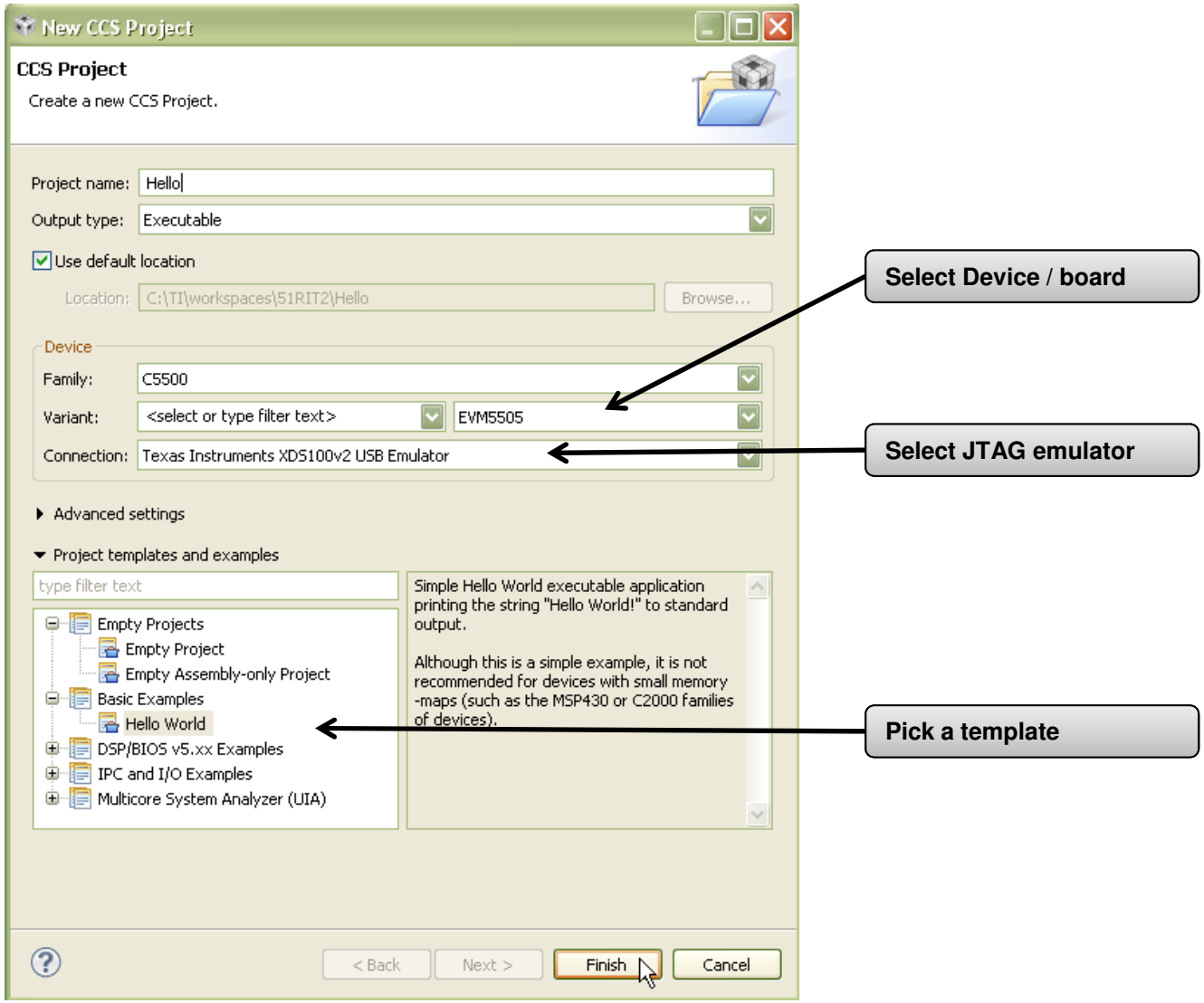
- USB Interface
- Compatible with CCS, IAR and other debuggers
- $99

MSP-FET430PIF



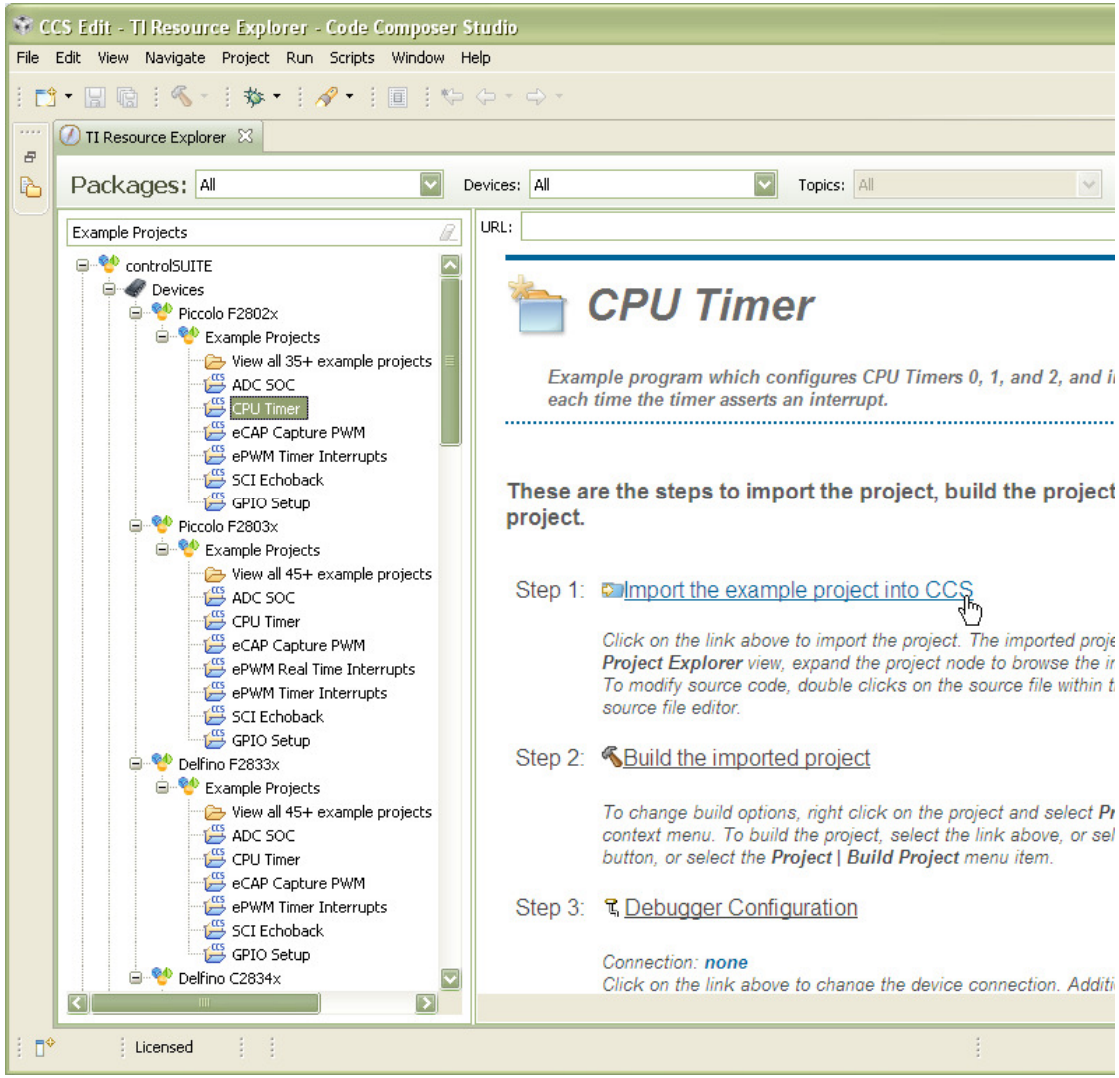
- Parallel Port Interface
- Compatible with CCS, IAR and other debuggers
- $49

Project Wizard

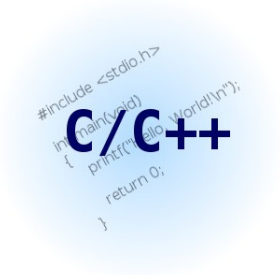


Resource Explorer

- Easily access a broad selection of packages such as controlSUITE, MSP430ware, StellarisWare, MCU SDK…
- Guides you step by step through using examples
- Provides links to documentation, videos and other collateral



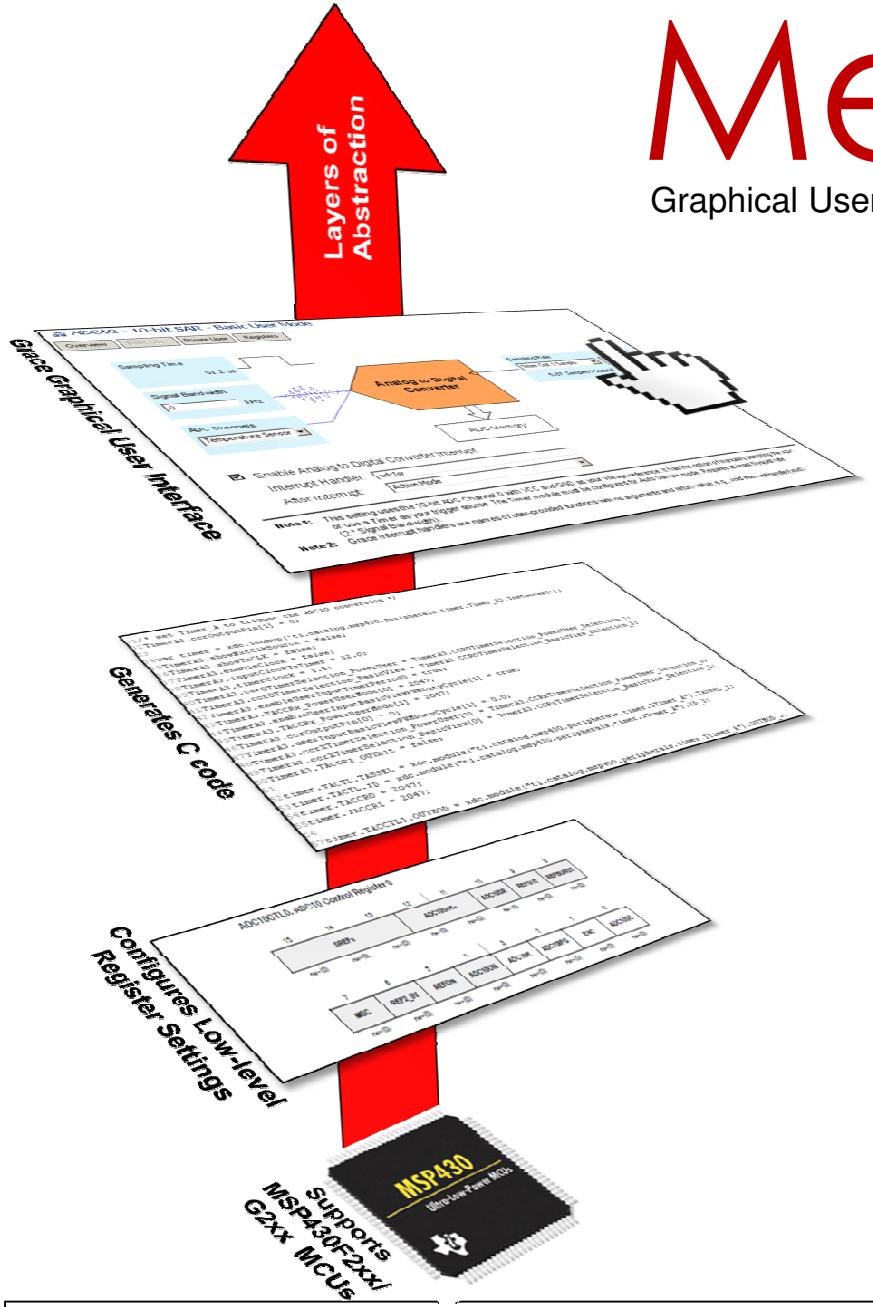
MCU Compiler



- Code-size conscious MCU compiler
  - Large range of performance vs. code size options
- Thorough Validation
  - Verified against industry-standard benchmarks (Plum Hall, Perennial, ACE…) and validation suites.
  - Continual evaluation of compiler's performance on thousands of benchmarks, ensuring performance stability of compiled code across releases

Meet Grace

Graphical User Interface for enabling and configuring MSP430 peripherals.



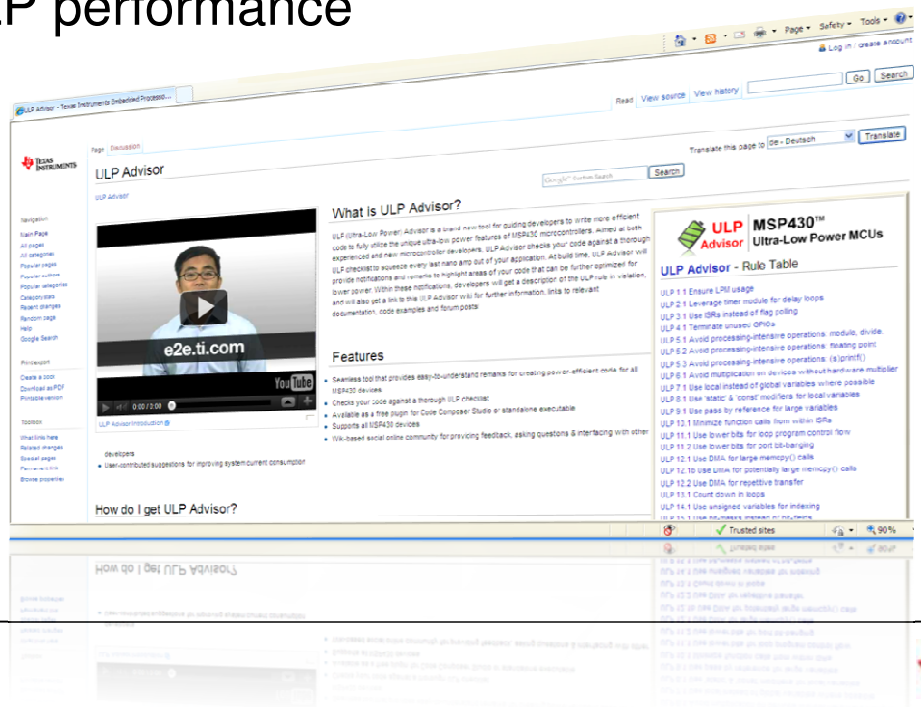
- **Visually configure MSP430 peripherals.**
  Enable and configure ADCs, DACs, timers, clocks, serial communication modules and more, by navigating buttons, drop-down menus, and text fields.
- **Generate code that is inserted directly into your Code Composer Studio project.**
  This code can be debugged and downloaded to your MSP430 device just like traditionally created code!
- **Helpful popups and tooltips are based on MSP430 documentation.**
  Spend less time flipping through pages of datasheets and user guides, and learn as you go!
- **Grace is seamlessly integrated into development environment**
  Grace supports all F2xx and G2xx Value Line microcontrollers, as well as MSP430's most popular hardware kits, including the $4.30 LaunchPad, the eZ430-F2013 and eZ430-RF2500.

www.ti.com/grace

Texas Instruments

ULP (Ultra-Low Power) Advisor

- Guides developers on how to write code that fully utilizes the unique Ultra Low Power features of MSP430 devices
- First true software-based teaching tool in the industry for enabling developers to always write code with Ultra-Low Power in mind
- Offers unique and not-easily-discovered ULP tips and tricks
- Developers will understand where they can improve their code line-by-line for ULP performance



www.ti.com/ulpadvisor

Texas Instruments

Code Composer Studio Licenses

- Free options:
  - 90 day evaluation
  - 16KB Code Size Limited – MSP430 Only
  - When using XDS100 JTAG emulators (currently not for MSP430)
  - Tied to development kits with onboard emulation (not for MSP430)
- Purchase professional tools:
  - Starting at $495
  - Node Locked License (tied to a PC)
  - Floating Licenses available (shared licenses)
  - www.ti.com/ccstudio

http://processors.wiki.ti.com/index.php/Download_CCS

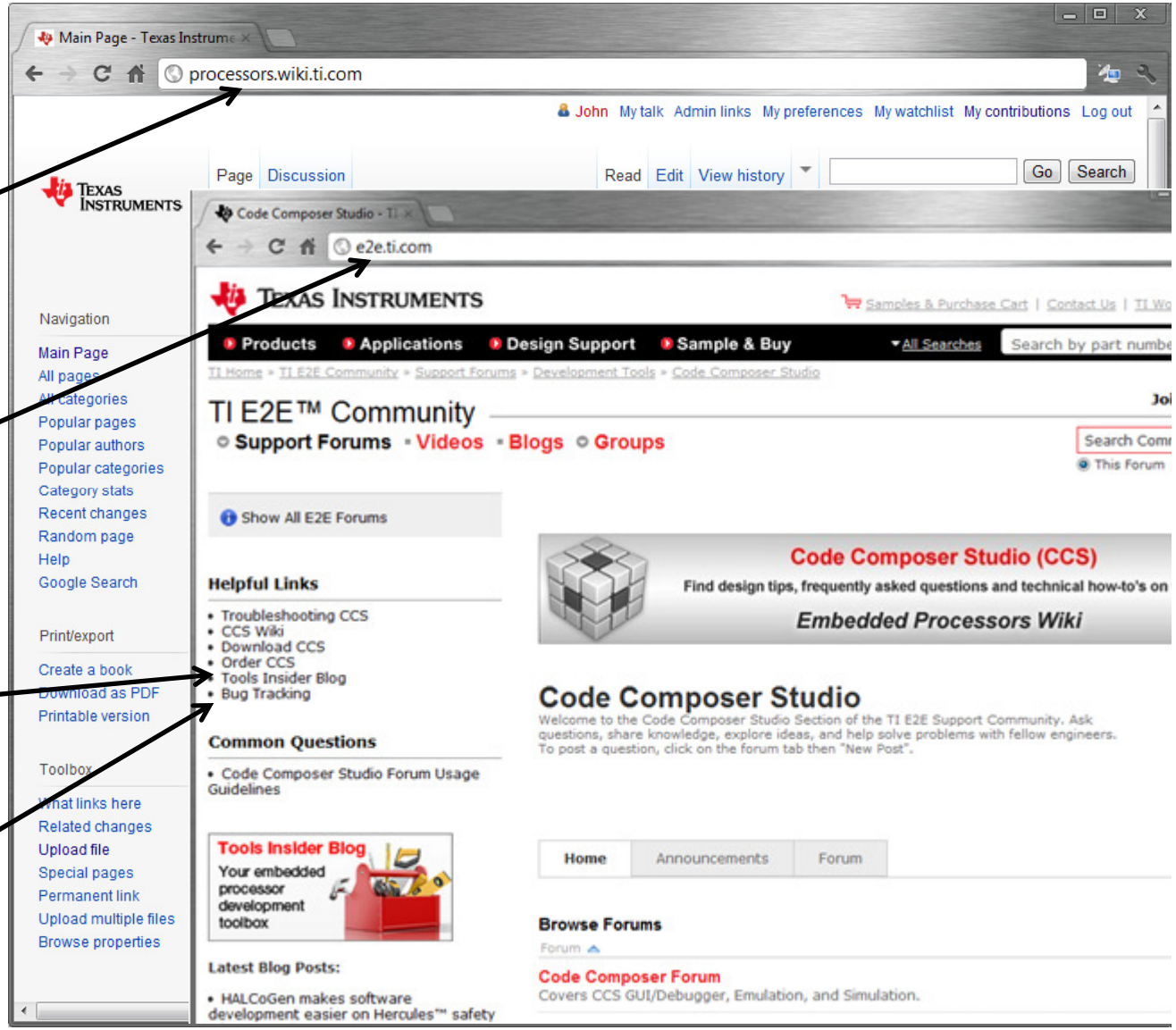
Support

Embedded Processors Wiki provides in depth information on using tools, software and devices

Ask questions and get support via the E2E community. Forums for Code Composer Studio, Compilers, BIOS and devices.

Learn about new software and tools via the Tools Insider Blog.

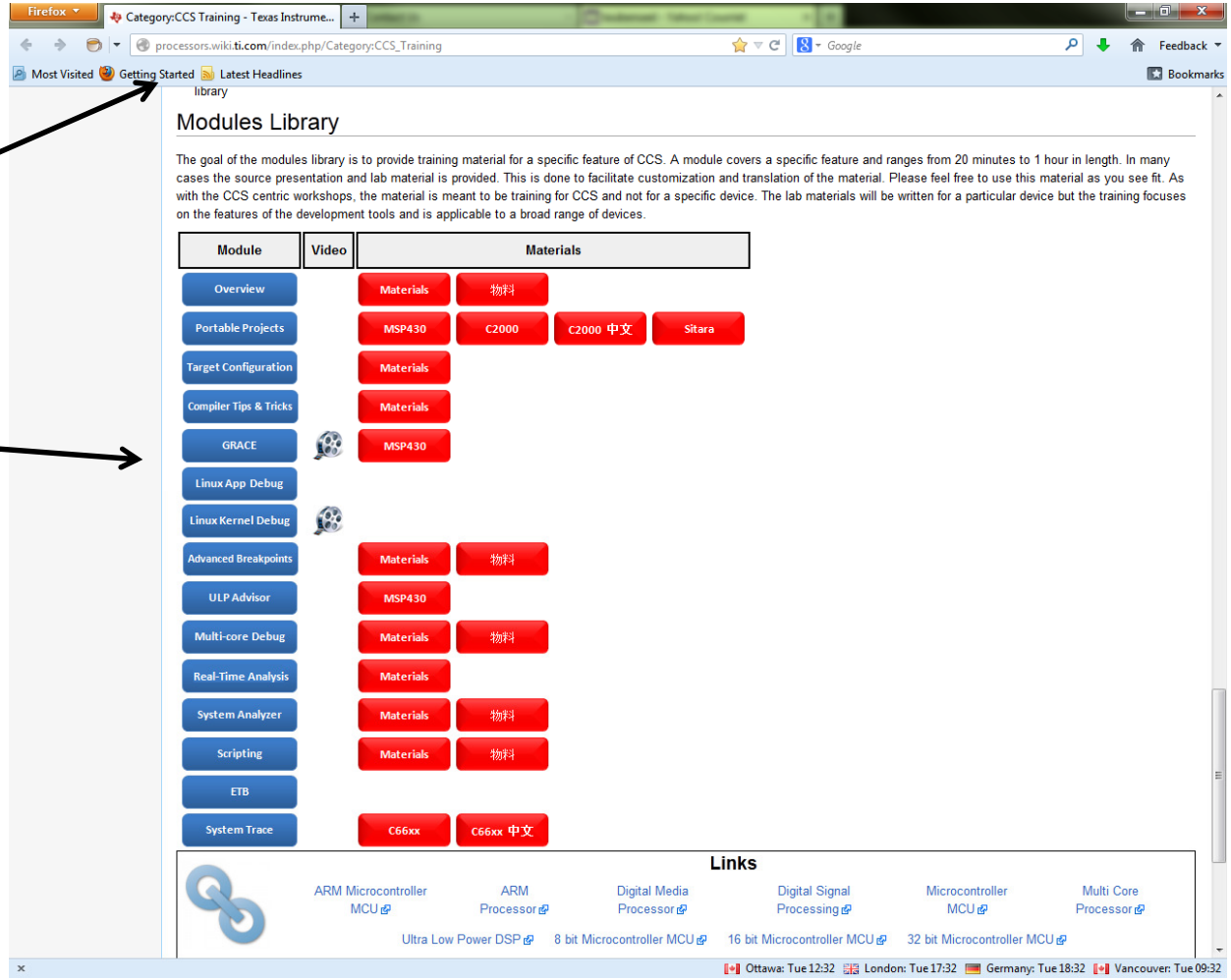
Check the status of bugs or enhancement requests.



Training

Further up this wiki many workshops, videos and further helpful tools are available.

Training modules and video tutorials on specific features



http://processors.wiki.ti.com/index.php/Category:CCS_Training

Why use Code Composer Studio?

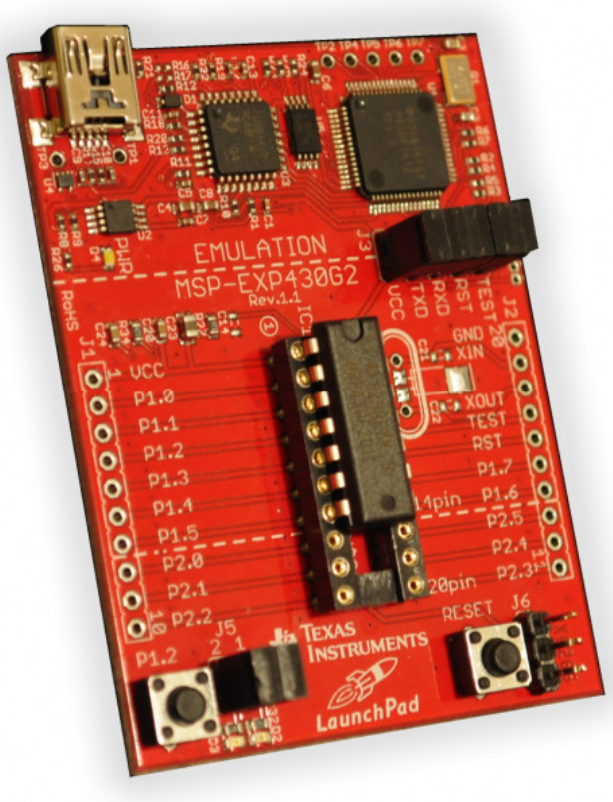
Code development environment	eases and speeds design and troubleshooting.
Advanced GUI framework	simplifies project management.
Multi-processor debugging	manages status and information from multiple cores.
Flexible project environment	aids control of compiler & SYS/BIOS™ sw kernel foundation.
Debug server scripting interface	enables the automation of common tasks.
Update manager tool	automatically manages tool updates.

Getting Started with CCSv5 and LaunchPad

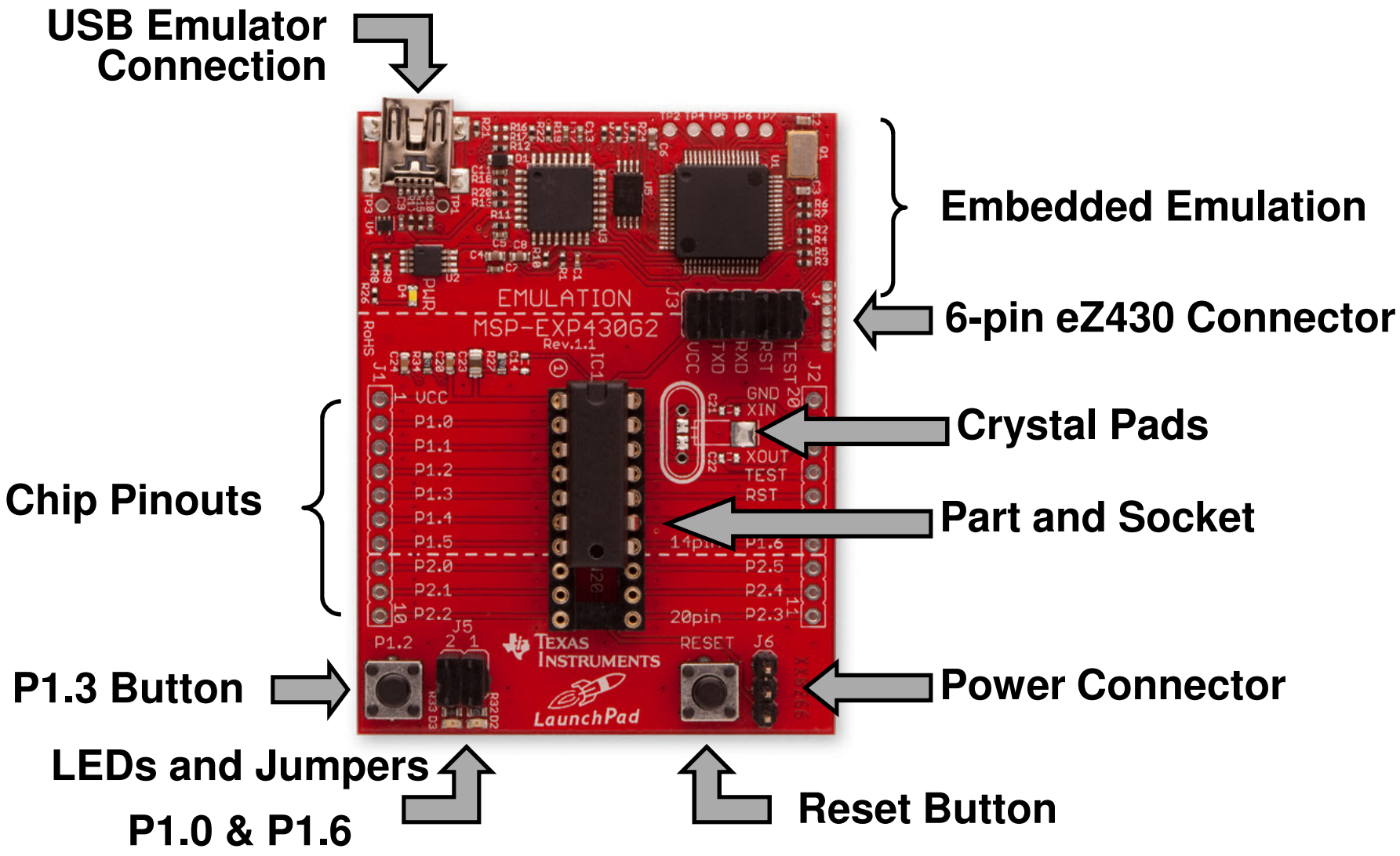
What is LaunchPad?

- Low cost (under $5), easy-to-use development tool intended for beginners and experienced users alike for the MSP430G2xx Value Line series devices

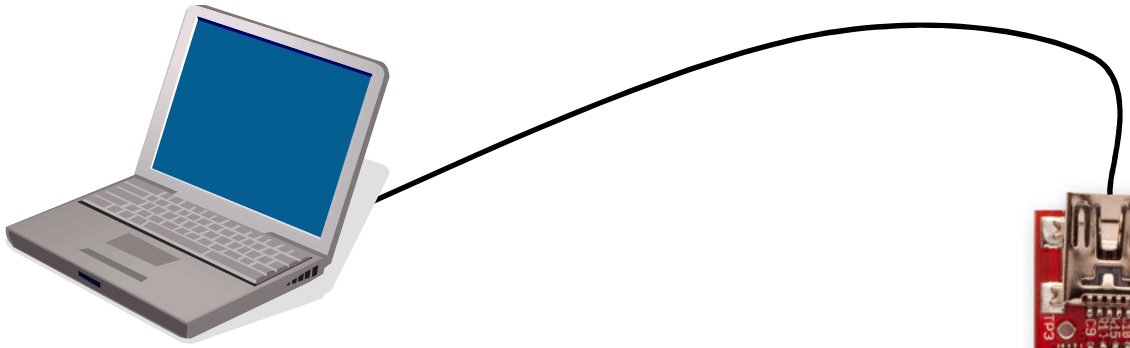
- Complete development environment that features integrated USB-based emulation and all of the hardware and software necessary to develop applications



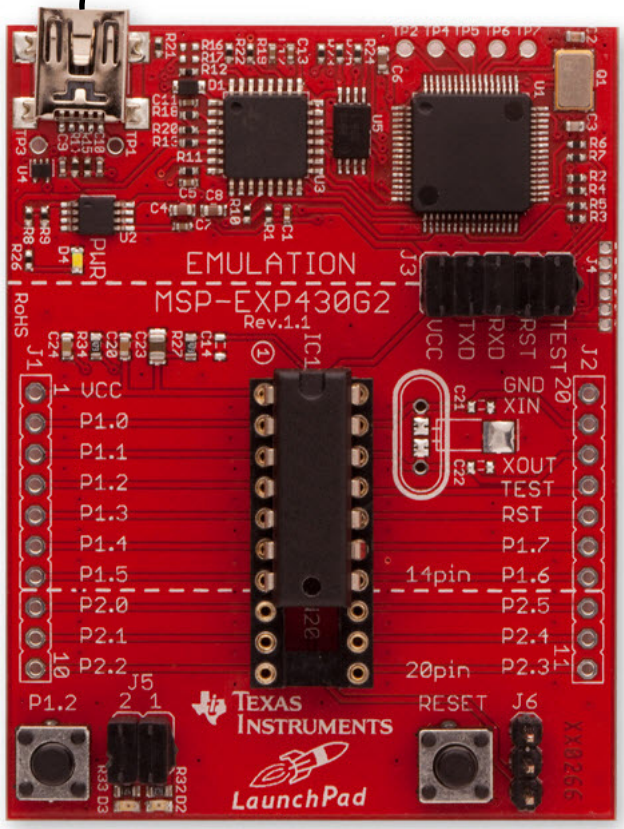
LaunchPad: Development Board



LaunchPad: Hardware Setup



Connect mini-USB cable from PC to board



LaunchPad

- Want to learn more about LaunchPad?
  - Check out: http://processors.wiki.ti.com/index.php/MSP430_LaunchPad_%28MSP-EXP430G2%29

LAB conventions

- Lab steps are numbered for easier reference
  1. ...
  2. ...

- Explanations, notes, warnings are written in blue
  - Warnings are shown with
  - Information is marked with
  - Tips and answers are marked with
  - Questions are marked with

Blink LED Example: Exercise Summary

- Key Objectives
  - Create and build a simple program to blink LED1
  - Start a debug session and load/flash the program on the Launchpad
  - Run the program to blink LED1
- Tools and Concepts Covered
  - Workspaces
  - Welcome screen / Resource Explorer
  - Project concepts
  - Basics of working with views
  - Debug launch
  - Debug control
  - Profile Clock
  - Local History
  - Build Properties
  - Changing compiler versions

LAB 1: BLINK LED EXAMPLE

30 MINUTES

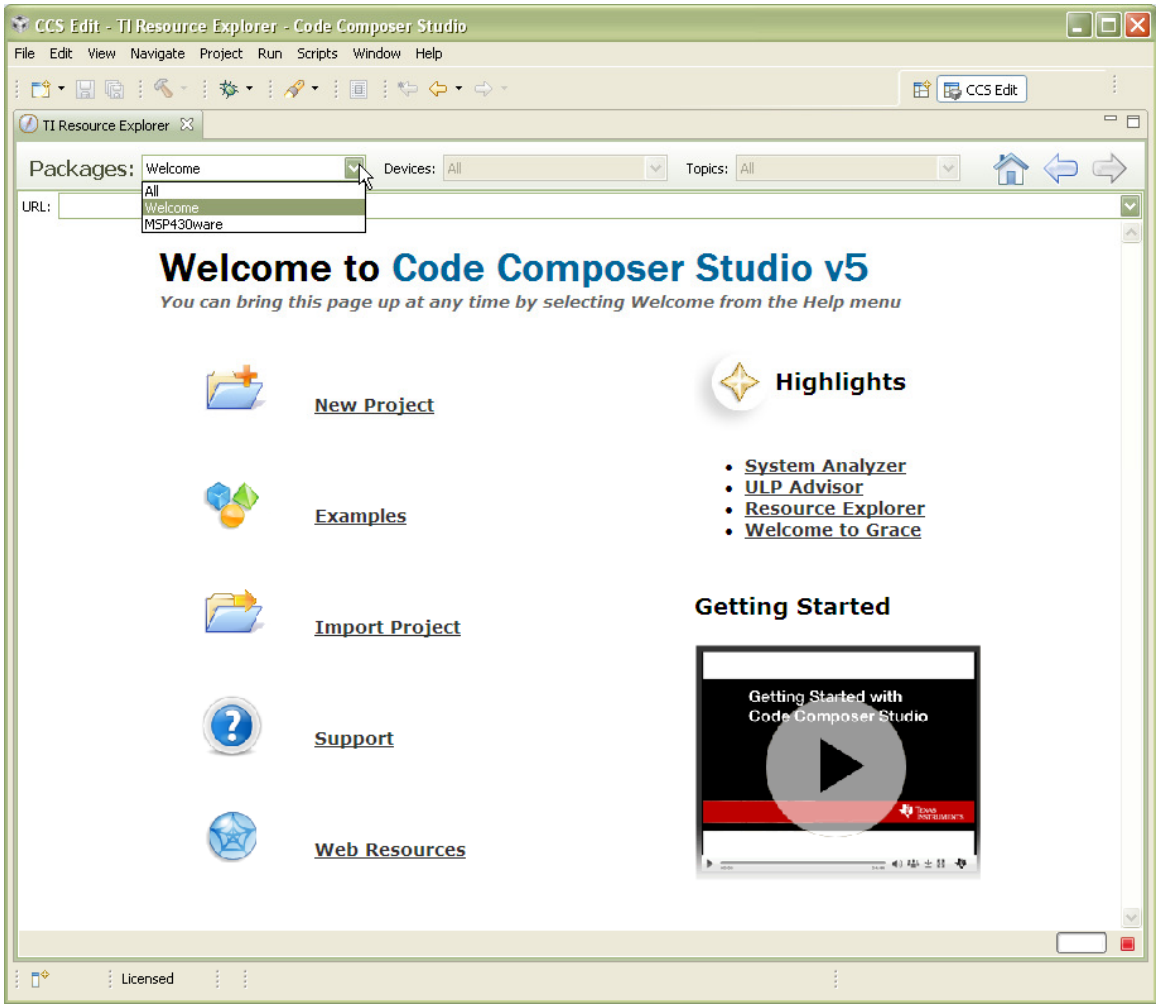
Open CCS and select the default workspace
You can close the TI Resource Explorer View (it will not be used)

Eclipse Concept: Workspaces

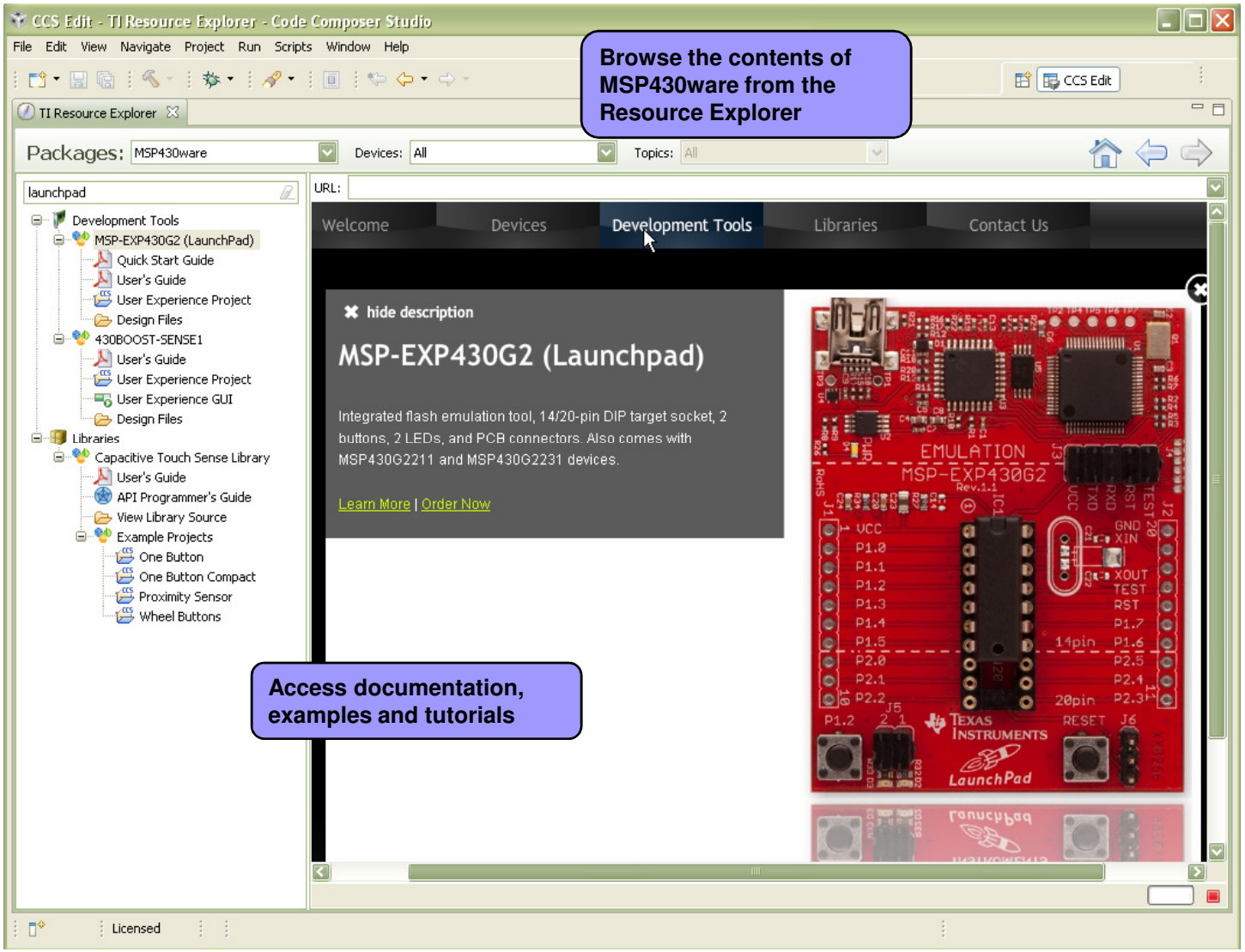
- Main working folder for CCS
- Contains information to manage all the projects defined to it
  - The default location of any new projects created
- User preferences, custom perspectives, cached data for plug-ins, etc all stored in the workspace
- Multiple workspaces can be maintained
  - Only one can 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

View: Resource Explorer

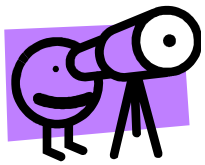
- The ‘Resource Explorer’ will display the ‘Welcome’ page the first time CCS is used with a new workspace
- Getting Started video introduces you to CCS
- Contains links to documentation, examples, support resources
- Buttons to create a new project or import an existing one into your workspace
- ‘Help->Welcome to CCS’ to return to the Resource Explorer in the future

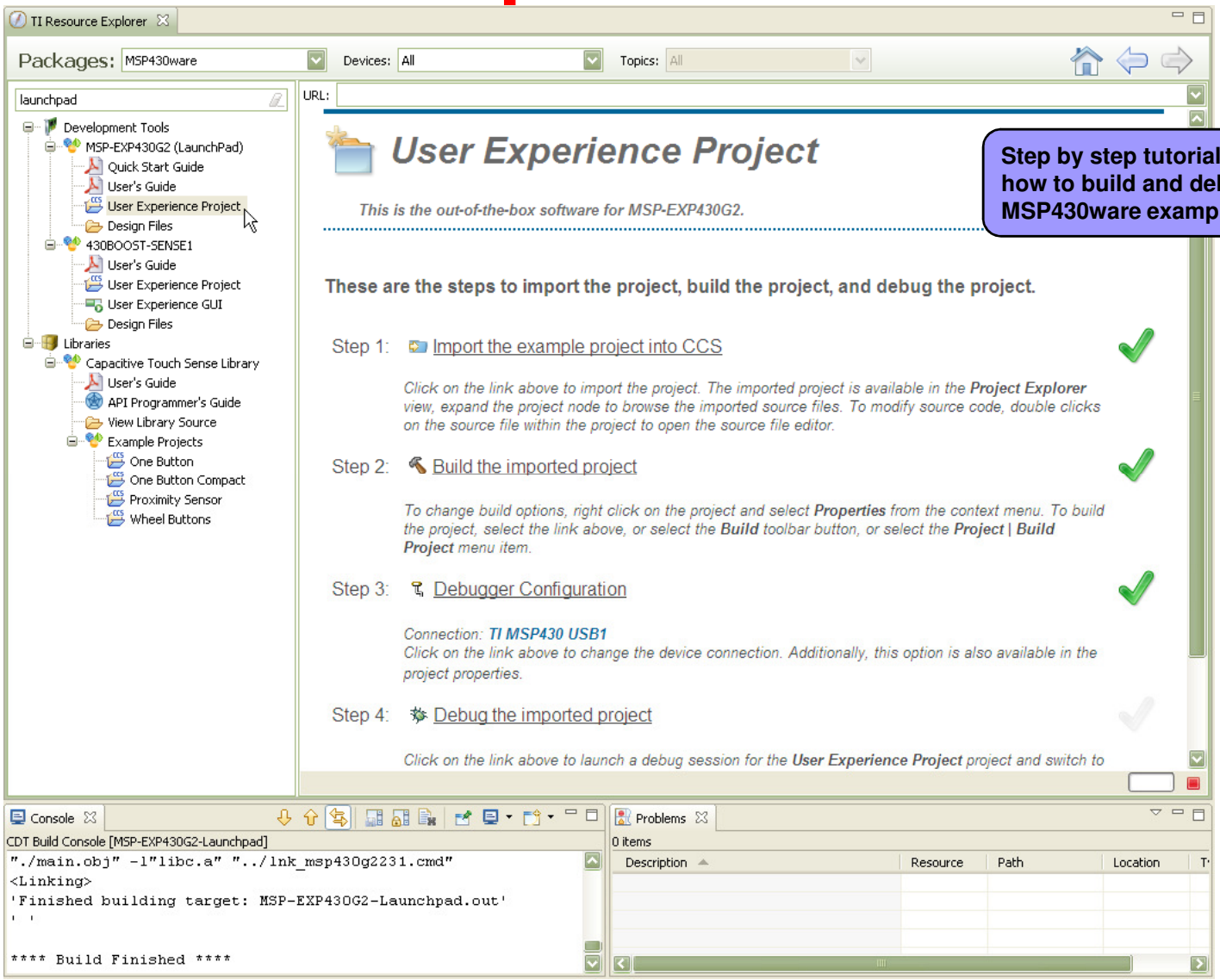


Resource Explorer: MSP430ware



Resource Explorer: Tutorials





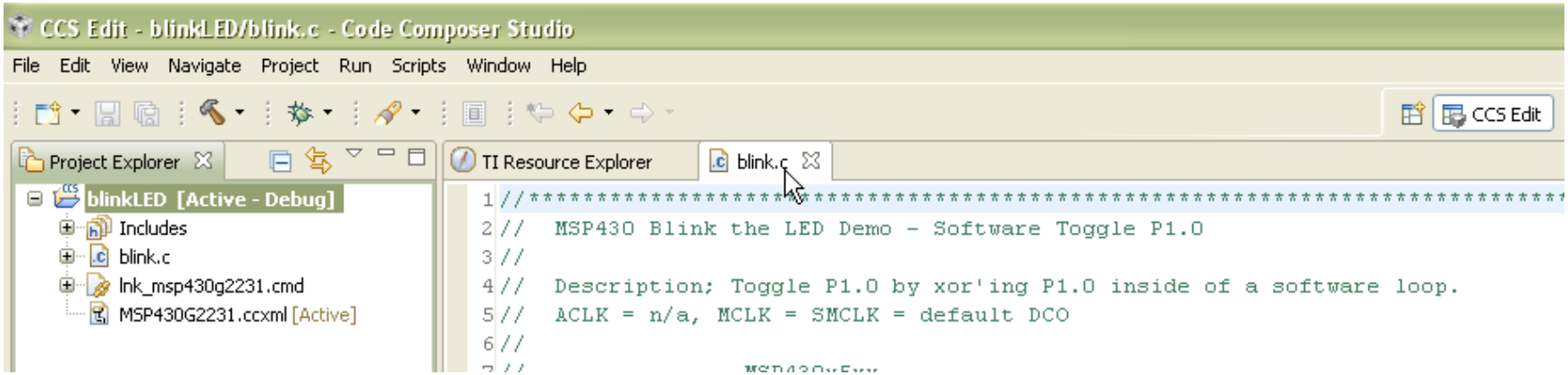
Step by step tutorials on how to build and debug MSP430ware examples

Eclipse Concept: Projects

- Projects map to directories in the file system
- Files can be added or linked to the project
  - Adding file to project
    - Copies the file into your project folder
  - Linking file to project
    - Makes a reference to the file in your project
    - File stays in its original location
- Projects are either open or closed
  - Closed Projects:
    - Still defined to the workspace, but it cannot be modified by the Workbench
    - The resources of a closed project will not appear in the Workbench, but the resources still reside on the local file system
    - Closed projects require less memory and are not scanned during routine activity
- Projects that are not part of the workspace must be imported into the active workspace before they can be opened
  - Both CCSv4/5, CCE projects and legacy CCSv3 projects can be imported into the workspace

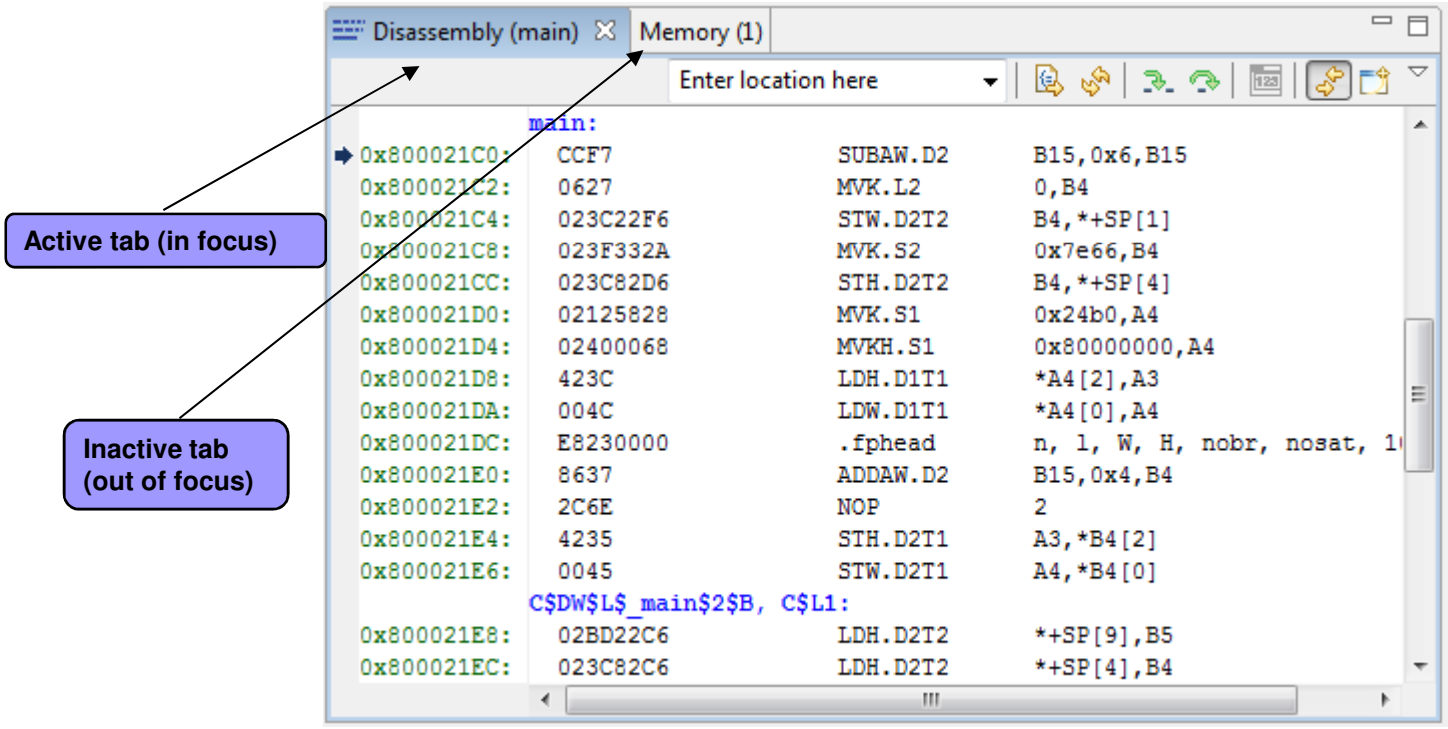
View: Project Explorer

- Displays all projects defined in the active workspace
- The view is mostly a representation of the file system of the project folder
  - Linked files are marked with a special link graphic in the icon
- Use filters to hide various file types to reduce clutter in the view
  - Default is to filter CCS generated project files (.*)



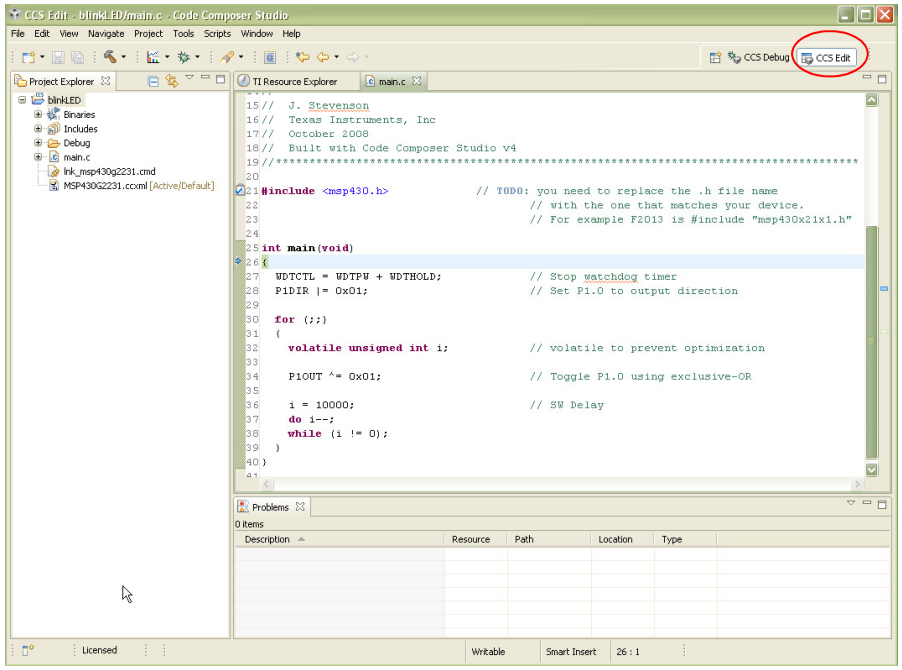
Eclipse Concept: Views

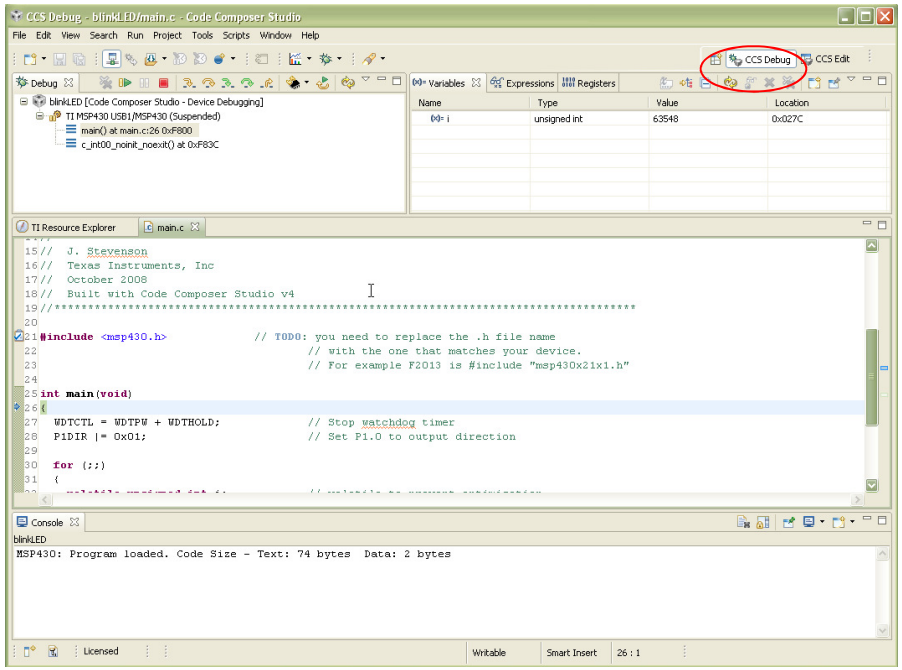
- Views are windows within the main Workbench window that provide visual representation of some specific information
  - Most views can be accessed via the 'View' menu
  - Views are context sensitive



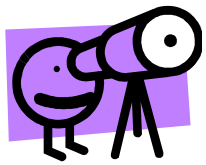
Eclipse Concept: Perspectives

- Defines the initial set and layout of views in the Workbench window
- Each perspective provides a set of functionality aimed at accomplishing a specific type of task ('CCS Edit' for project development, 'CCS Debug' for debugging, etc)
- Can create custom perspectives

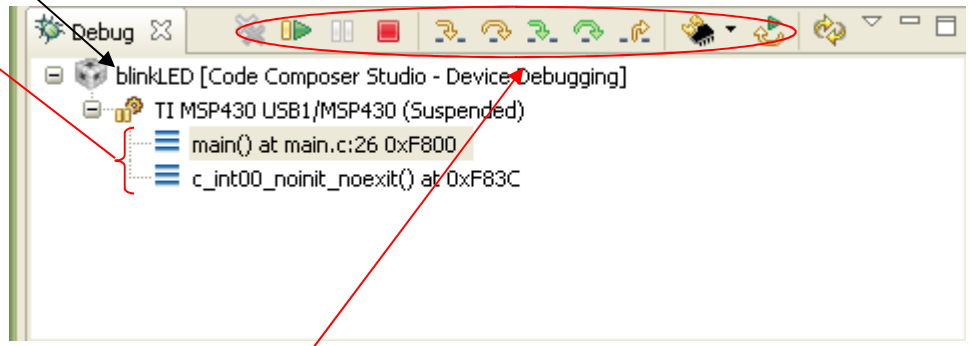




View: Debug

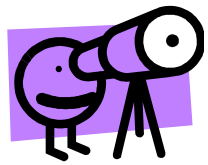


- The Debug view displays:
  - Target configuration or project
  - Call stack

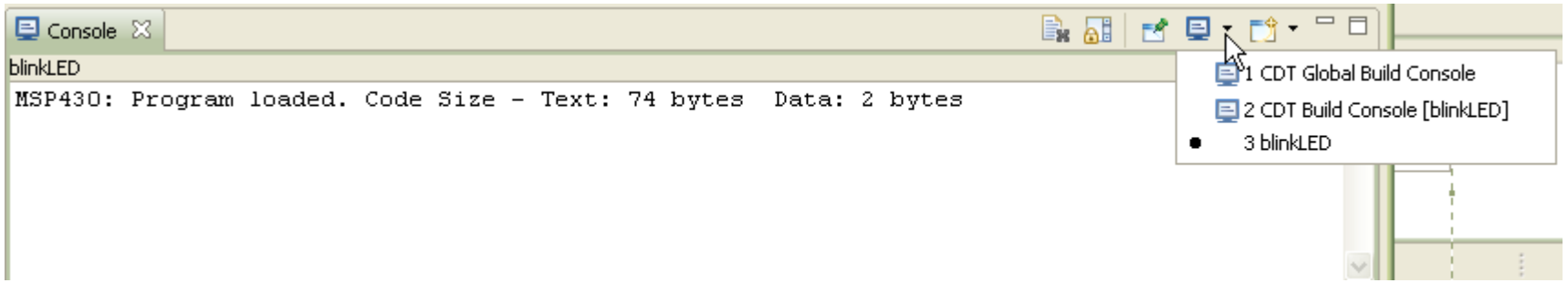


- Buttons to ‘run, halt, terminate (debug session), source and asm stepping, reset CPU, restart program

View: Console



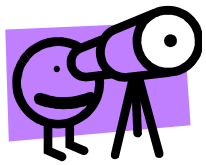
- Multiple contexts
  - Can display build messages or debug messages (including CIO) depending on which console is selected
  - Automatically switches contexts when a new message occurs
    - Can use the “Pin” option to prevent this
- You can open multiple console windows
  - CIO output in one and build messages in the other



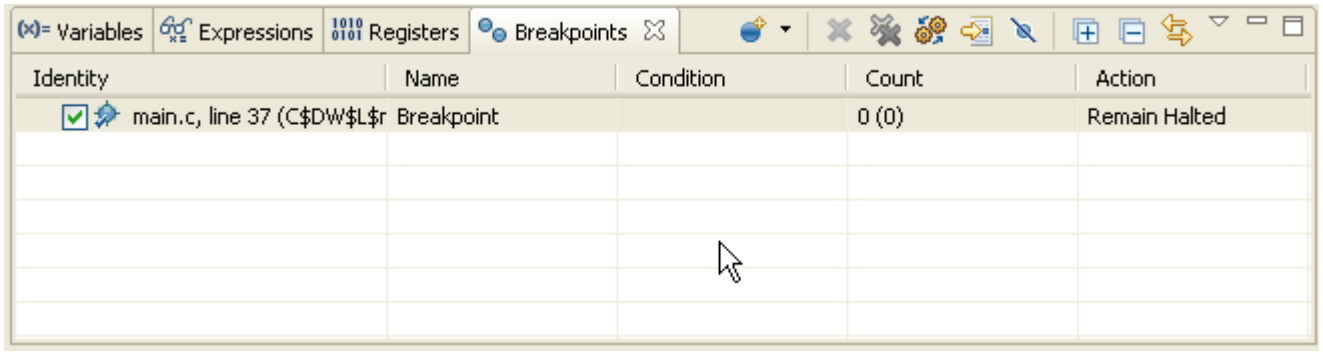
Breakpoint Types

- Software vs Hardware breakpoints
  - For software breakpoints the opcode in memory is replaced with a break opcode so that execution halts when that opcode is reached
  - For hardware breakpoints an address comparator is used on the device to compare the address being fetched with that stored in the comparator
- When placing breakpoints in Flash hardware breakpoints are used
  - MSP430 is special as we can use a single hardware breakpoint resource to enable unlimited “software” breakpoints in Flash
- You have a limited number of hardware breakpoints based on the number of comparators on the device
- Watchpoints are hardware breakpoints that are looking at data addresses instead of program addresses
  - i.e. break when a location or variable is accessed or written or written with a specific value (extra comparator required)

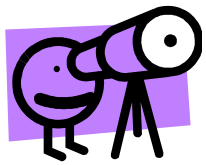
View: Breakpoints



- View all available breakpoints
- Can group breakpoints by CPU (multi-core device)
- Specify various actions when the breakpoint is triggered
  - Control Profiling (set profile halt/resume points)
  - File I/O
  - Run a GEL expression
  - Set a Watchpoint
  - Control CPU trace (on selected ARM & DSP devices)

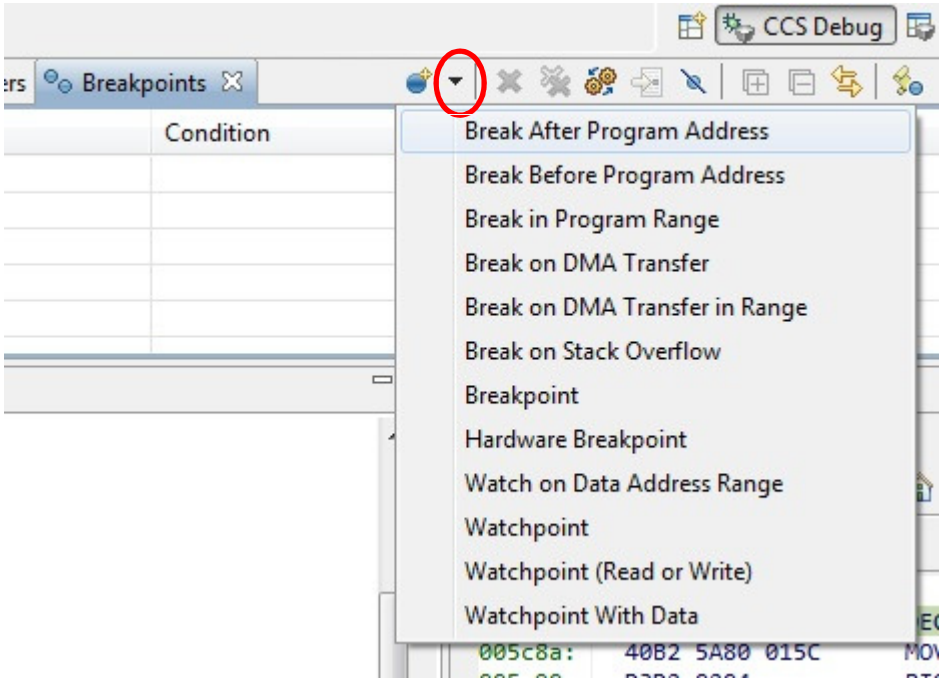


Further on Breakpoints



- Although not applicable in this lab, some more advanced MSP430 devices offer further breakpoint options (see the screen capture)
- These include the ability to use operators as well as set a range, eg for a data, program address or DMA based breakpoint, etc
- Watchpoints on read/write are also available on certain more advanced msps

- Further examples and information are to be available from the processor wiki site
- http://processors.wiki.ti.com/index.php/Main_Page



Debugger Options

- Many debugging features can be enabled/disabled from the Debugger Options
- During a debug session in the ‘CCS Debug’ perspective:
  - ‘ Tools->Debugger Options->Generic Debugger Options’
- Configure a variety of debug options like enable/disable:
  - auto-run to main
  - auto-connect to a HW target
  - real-time options (on supported HW)
  - program verification on load
  - etc…
- Use the “Remember My Settings” option to have the settings apply for subsequent debug sessions

Temperature Sense Demo: Briefing

- Key Objectives
  - Create and build the Temperature Sense Demo
  - Start a debug session and load/flash the program on the LaunchPad
  - Run the program to start Temperature Sense Demo
- Tools and Concepts Covered
  - Adding source files
  - “Focus” concept
  - Loading Symbols
  - Changing Build Options
  - Changing Compiler Version

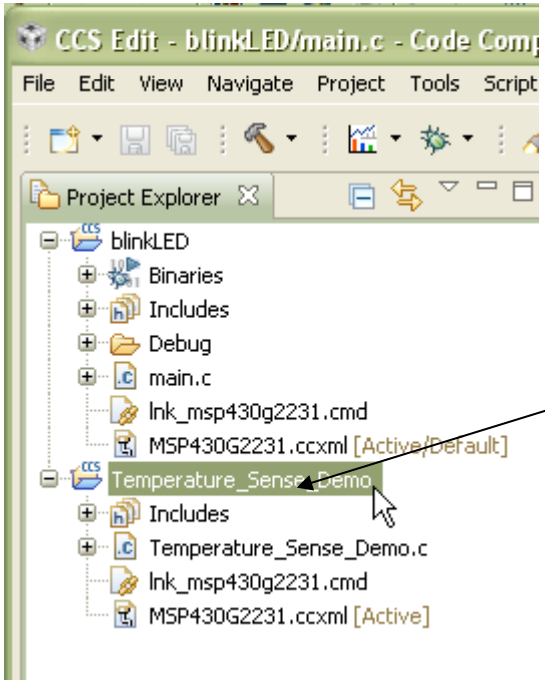
LAB 2: TEMPERATURE SENSE DEMO

30 MINUTES

Open CCS and select the default workspace
You can close the TI Resource Explorer View (it will not be used)

Eclipse Concept: Focus

- Focus refers to the highlighted portion of the workbench
  - Can be an editor, a view, a project, etc.
- This concept is important since several operations inside Eclipse are tied to the element in focus
  - Project build errors, console, menu and toolbar options, etc.



'Temperature_Sense_Demo' project is in 'Focus' since it has been selected. So pressing the 'Debug' button will build the project and start the debugger for the 'Temperatrue_Sense_Demo' project

Compiler Versions

- Each project specifies which version of the compiler to use
  - Actually set on a per configuration level
  - I.e. Debug can use one version and release another
- When you download a new compiler if you want to use it you have to change the compiler version specified by your project
- CCS will allow you to select from all the compilers that it knows about on your computer
- When you install a new compiler via the Update Manager in CCS it will automatically know where the computer is
  - However you can also tell CCS where another version of the compiler is located

Blink LED Example: Exercise Summary

- At this point you experimented the following concepts:
  - Workspaces
  - Welcome screen / Resource Explorer
  - Project concepts
  - Basics of working with views
  - Debug launch
  - Debug control
  - Profile Clock
  - Local History
  - Build Properties
  - Changing compiler versions
  - Debug symbols
  - Profile

SHARING PROJECTS

Sharing Projects

- Sharing “Simple” projects (all source/header files are contained in the project folder)
- Sharing “Linked file” projects only (source is outside of project)
  - Typical for non-trivial projects
  - Very common when using source control to have a source repository that is in a location outside of the project
  - Effort involves making the project “portable”
    - Eliminate absolute paths in the project
  - **This is the most common use case**

Simple Projects

- USE CASE: Wish to share (give) a project folder and all needed source files to build the project. All source files are inside the project folder.
- Easy to share projects with no linked files:
  - The entire project folder can be distributed to another "as-is"
  - The user who receives the project can import it into their workspace going to menu *Project* → *Import Existing CCE/CCS Project* and selecting the copied folder
  - Works well for simple projects that only reference files inside the project folder

“Linked file” Projects

- USE CASE:
  - Wish to share (give) a project folder only. The person receiving the project file already has a copy of the source files
  - Wish to check the project folder/files into source control
- Most use cases involve sharing JUST the projects instead of bundling all the source files
  - People will have their own local copies of the source files
- Need to make the project portable to make sure the project is easily shared
- Portable projects avoid any absolute paths
- Ideal portable projects should be usable without modifying any of the project files
  - This is required for projects maintained in source control

Linked Path Variables

- Used when linking files to projects
  - Linked files are those outside of the project folder
- By default CCS (Eclipse) uses absolute paths for linked files
  - Obviously that does not work well if a project needs to be shared
- Instead use variables
  - Define a set of variables as part of your personal workspace
  - When linking the file choose to use a variable and CCS will automatically convert the absolute path to a path using the variable

Build Variables

- Paths are used in project options
  - Location of generated files
  - Search paths
- You can use absolute, relative or paths based on variables
- Depending on how your project is structured and what your include statements look like it is often best to use variables
- Types of variables
  - There are a number of pre-defined variables that you can use such as the project location, workspace location, compiler location…
  - You can define your own custom variables

Project vs Workspace Level Variables

- Linked Resource Path Variables and Build Variables can be set at the project or workspace level
- This current lab set these variables at the workspace level
- What is the benefit of setting these variables at the workspace level instead of the project level?
  - All projects can reuse the same variable (set it once)
  - Do not need to modify the project!
    - This is important for projects checked into source control and to avoid constant checkouts so the project can be written to!

LAB 3: PORTABLE PROJECTS

30 MINUTES

Open CCS and select the default workspace
You can close the TI Resource Explorer View (it will not be used)

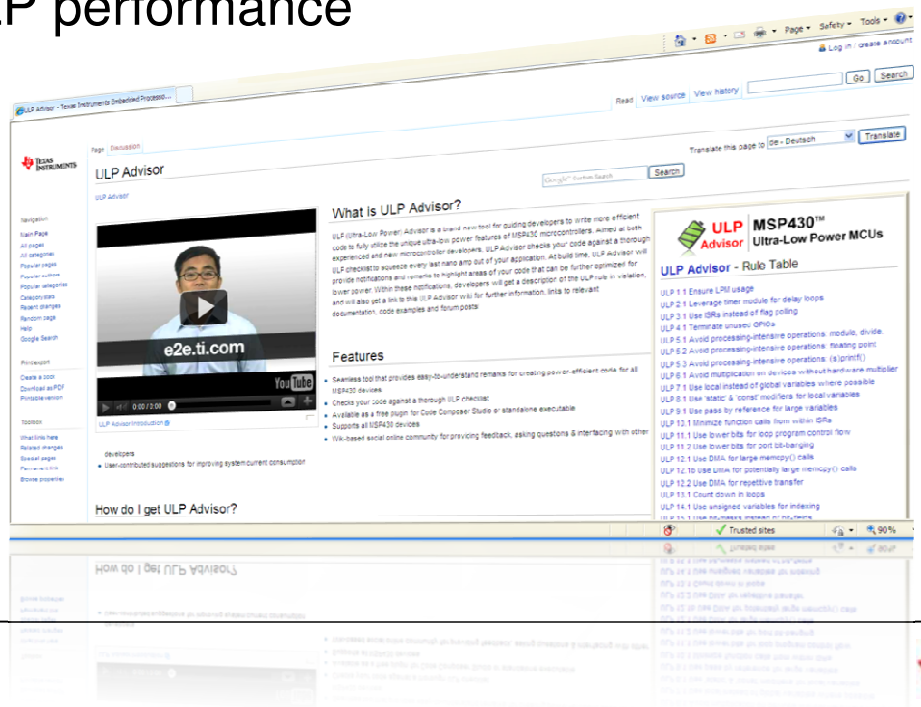
Portable Projects: Briefing

- Key Objectives
  - Create a project that uses a linked source file
  - Make the project portable
- Tools and Concepts Covered
  - Linked Resources
  - Linked Resource Path Variables
  - Build Variables

ULTRA LOW POWER ADVISOR

ULP (Ultra-Low Power) Advisor

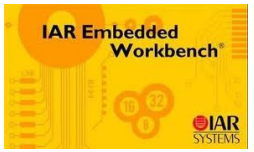
- Guides developers on how to write code that fully utilizes the unique Ultra Low Power features of MSP430 devices
- First true software-based teaching tool in the industry for enabling developers to always write code with Ultra-Low Power in mind
- Offers unique and not-easily-discovered ULP tips and tricks
- Developers will understand where they can improve their code line-by-line for ULP performance



www.ti.com/ulpadvisor

Texas Instruments

ULP Advisor | Integrated SW

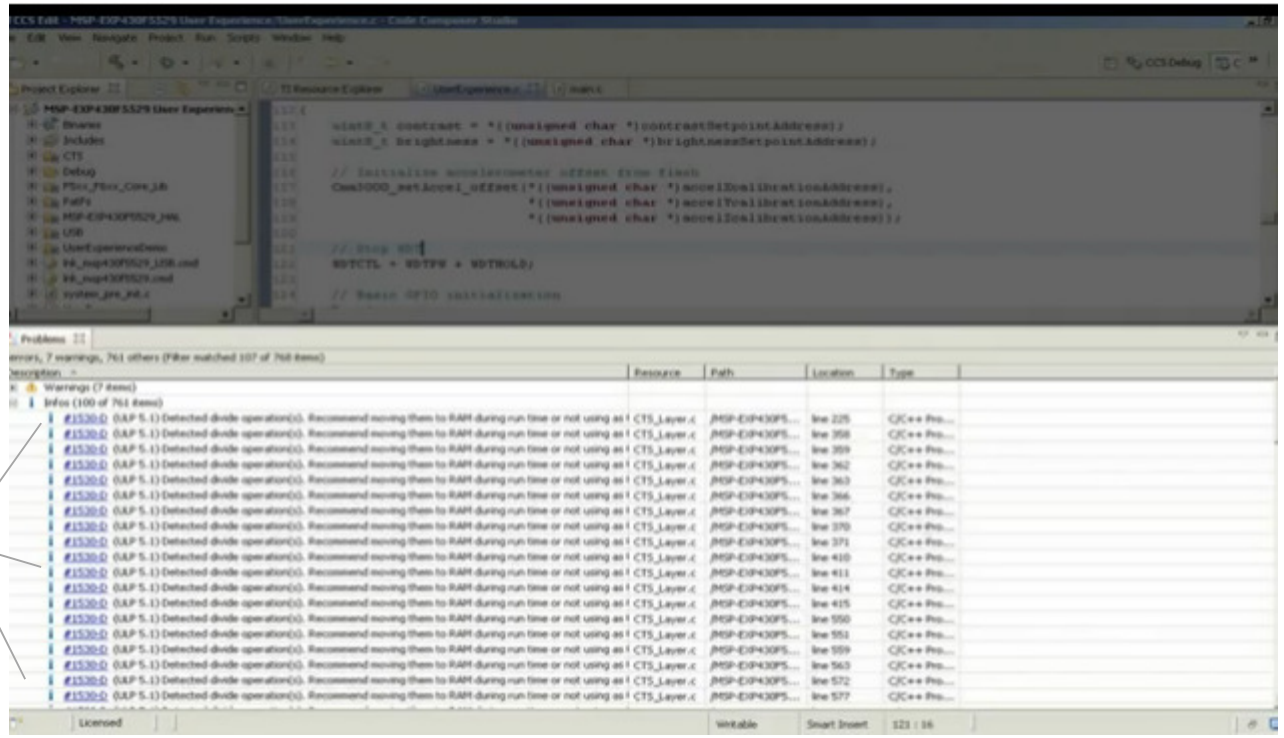


- IDE Integration
  - Comes with Code Composer Studio v5.2
  - Pre-installed as a plug-in for IAR
- Stand-alone version coming soon
  - Support for other IDEs, such as Open Source MSPGCC
- Automatically enabled for new projects

ULP Advisor | Operation

- Activated when compiling a project
- Scans through project's code files against a ULP checklist
- Highlights ULP violations and reports in **Problem View**

ULP Remarks



ULP Advisor | Wiki

- Each remark provides:
  - Brief suggestion
  - Hyperlink to wiki with detailed information on the rule
- Customers can find:
  - Background information, why the rule is important
  - What ULP Advisor checked to issue this remark
  - Now that I know why and whether or not:
    - what is wrong with my code or
    - understand why my code intentionally consumes such power
  - Code examples
  - Videos
  - Tutorials
  - Links to relevant documentation
  - Leverages TI's huge e2e online community.

ULP Advisor | Operation

- Works on all MSP430 devices
- Works on all MSP430 C code

- ULP Advisor supports all MSP430 devices and its power-savings can be leveraged across all applications.

LAB 4: ULTRA LOW POWER ADVISOR

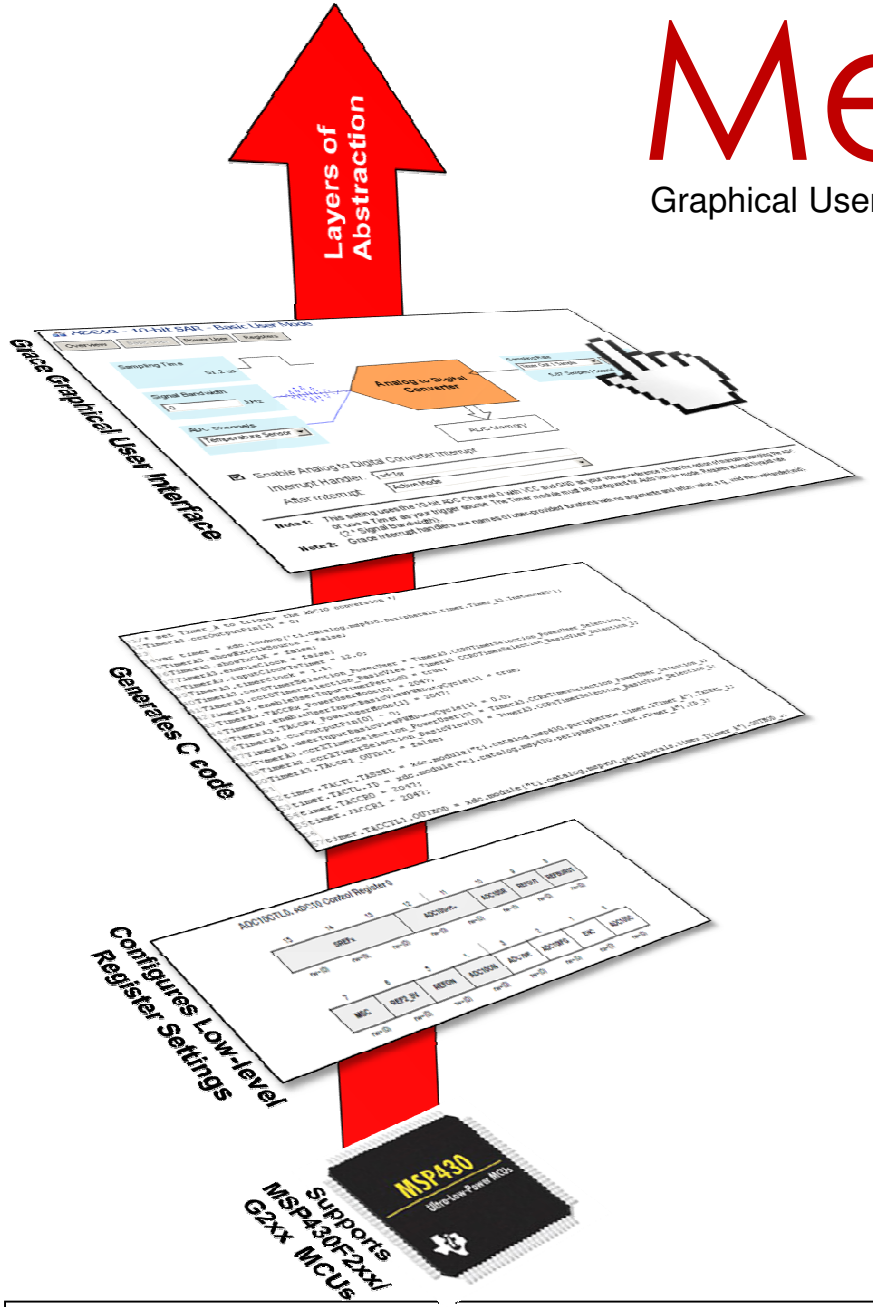
20 MINUTES

Open CCS and select the default workspace
You can close the TI Resource Explorer View (it will not be used)

GRACE

Meet Grace

Graphical User Interface for enabling and configuring MSP430 peripherals.



- **Visually configure MSP430 peripherals.**
  Enable and configure ADCs, DACs, timers, clocks, serial communication modules and more, by navigating buttons, drop-down menus, and text fields.
- **Generate code that is inserted directly into your Code Composer Studio project.**
  This code can be debugged and downloaded to your MSP430 device just like traditionally created code!
- **Helpful popups and tooltips are based on MSP430 documentation.**
  Spend less time flipping through pages of datasheets and user guides, and learn as you go!
- **Grace is seamlessly integrated into development environment**
  Grace supports all F2xx and G2xx Value Line microcontrollers, as well as MSP430's most popular hardware kits, including the $4.30 LaunchPad, the eZ430-F2013 and eZ430-RF2500.

www.ti.com/grace

Texas Instruments

What is Grace?

- Grace is a tool that allows you as an MSP430 developer to generate the peripheral's set up code within minutes

Enable peripherals with a single click

Interactive elements for peripheral configuration

Generates fully-commented, easy-to-read C code

Grace

- Fully harness MSP430 MCUs integrated analog and digital peripherals with Grace Software
- Grace software supports all MSP430F2xx and G2xx Value Line MCU devices
- Seamless integration Code Composer Studio

LAB 5: GRACE

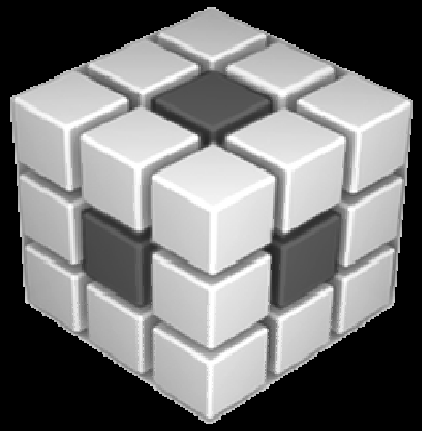
20 MINUTES

Open CCS and select the default workspace
You can close the TI Resource Explorer View (it will not be used)

Resources

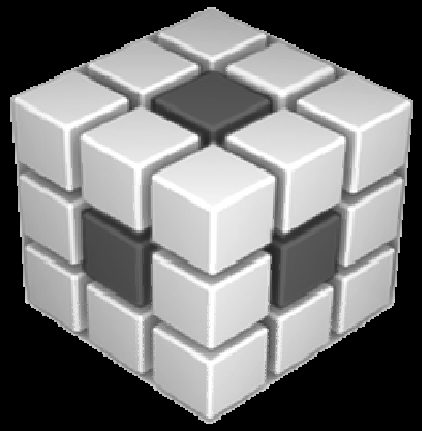
- http://processors.wiki.ti.com/index.php/Download_CCS
- http://processors.wiki.ti.com/index.php/Main_Page
- http://processors.wiki.ti.com/index.php/Category:CCS
- http://processors.wiki.ti.com/index.php/Category:CCS_Training
- http://processors.wiki.ti.com/index.php/MSP430

Thank you!



Texas Instruments

Questions?



Texas Instruments