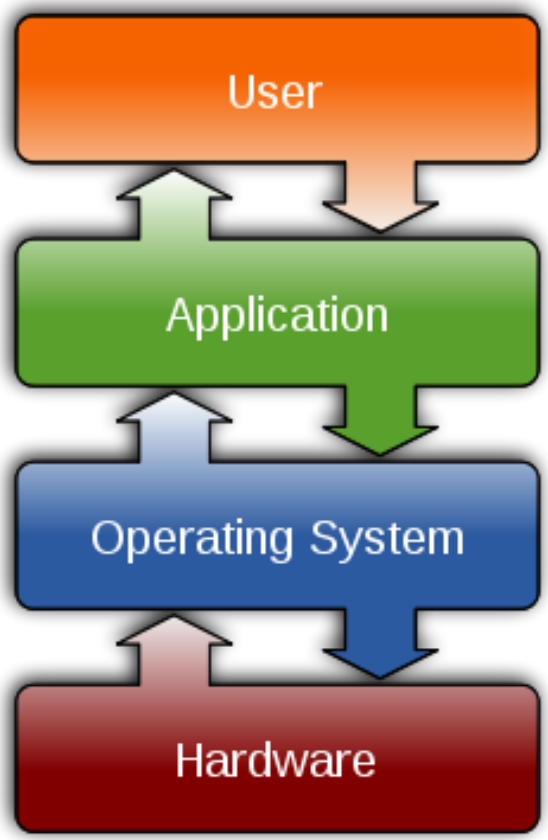
Introduction to CCSv5

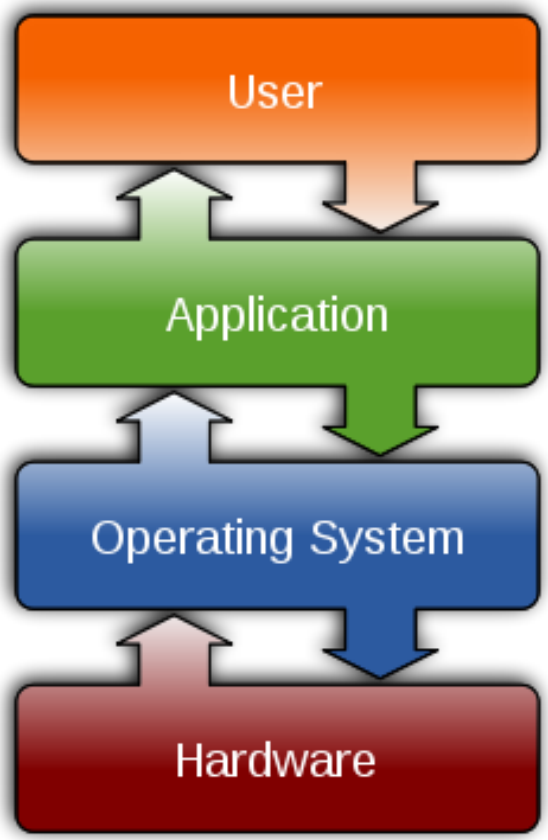
Outline

- **Intro to CCSv5**
  - Functional Overview
  - Perspectives
  - Projects
  - Target Configuration
  - Build Config & Options
  - Licensing/Pricing
  - CCSv5 – For More Info…

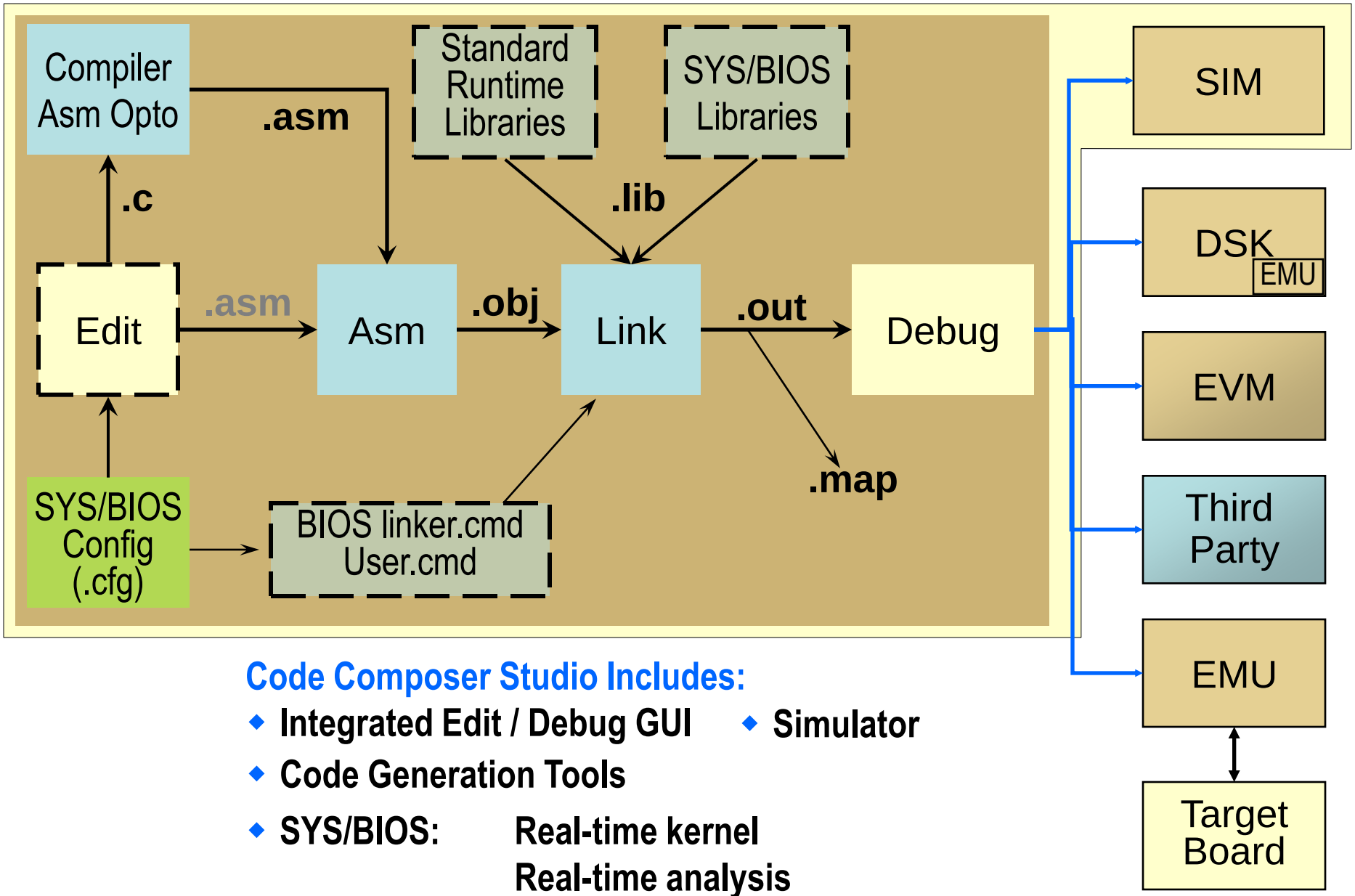


Outline

- Intro to CCSv5
  - **Functional Overview**
  - Perspectives
  - Projects
  - Target Configuration
  - Build Config & Options
  - Licensing/Pricing
  - CCSv5 – For More Info…



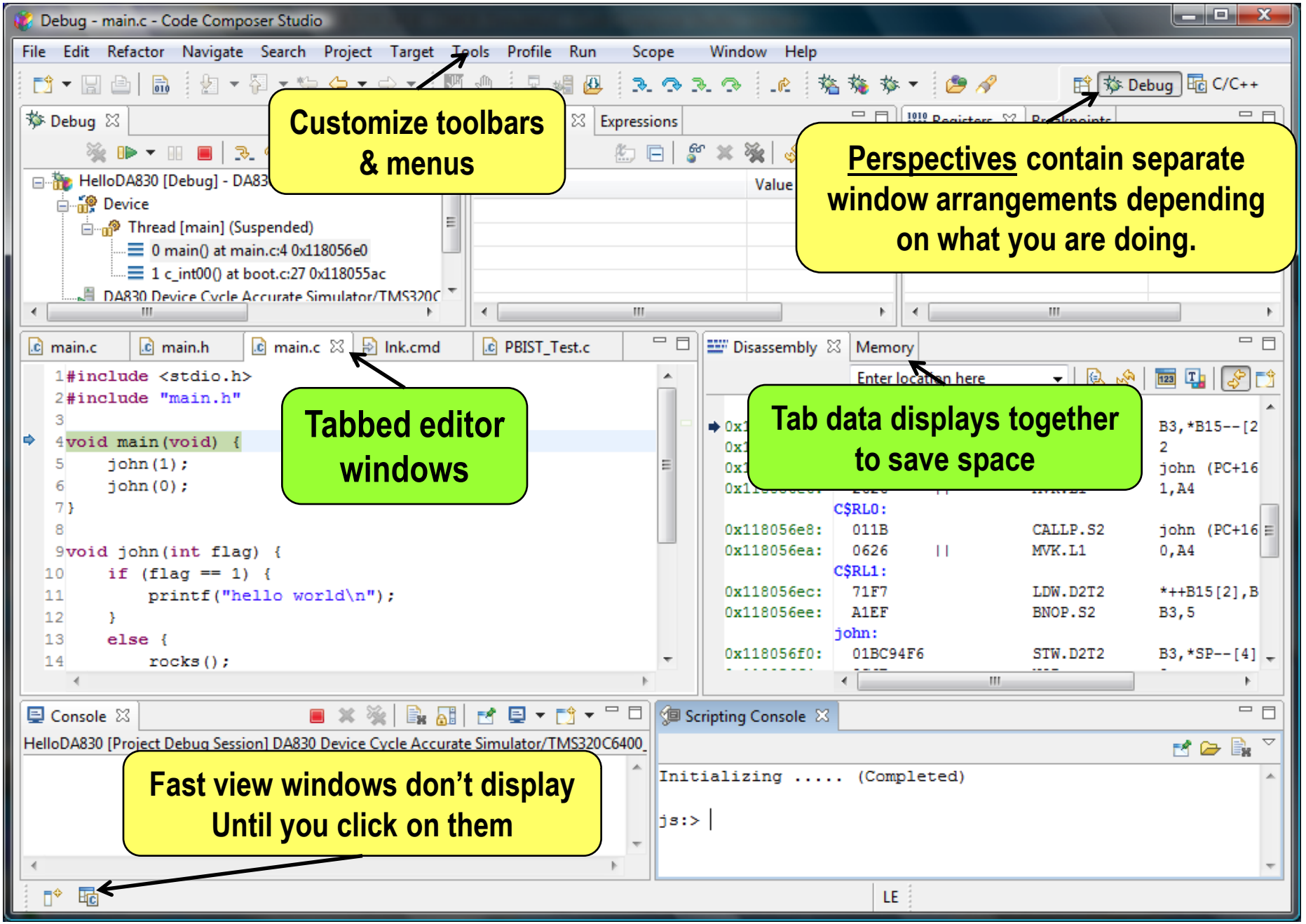
CCS Functional Overview



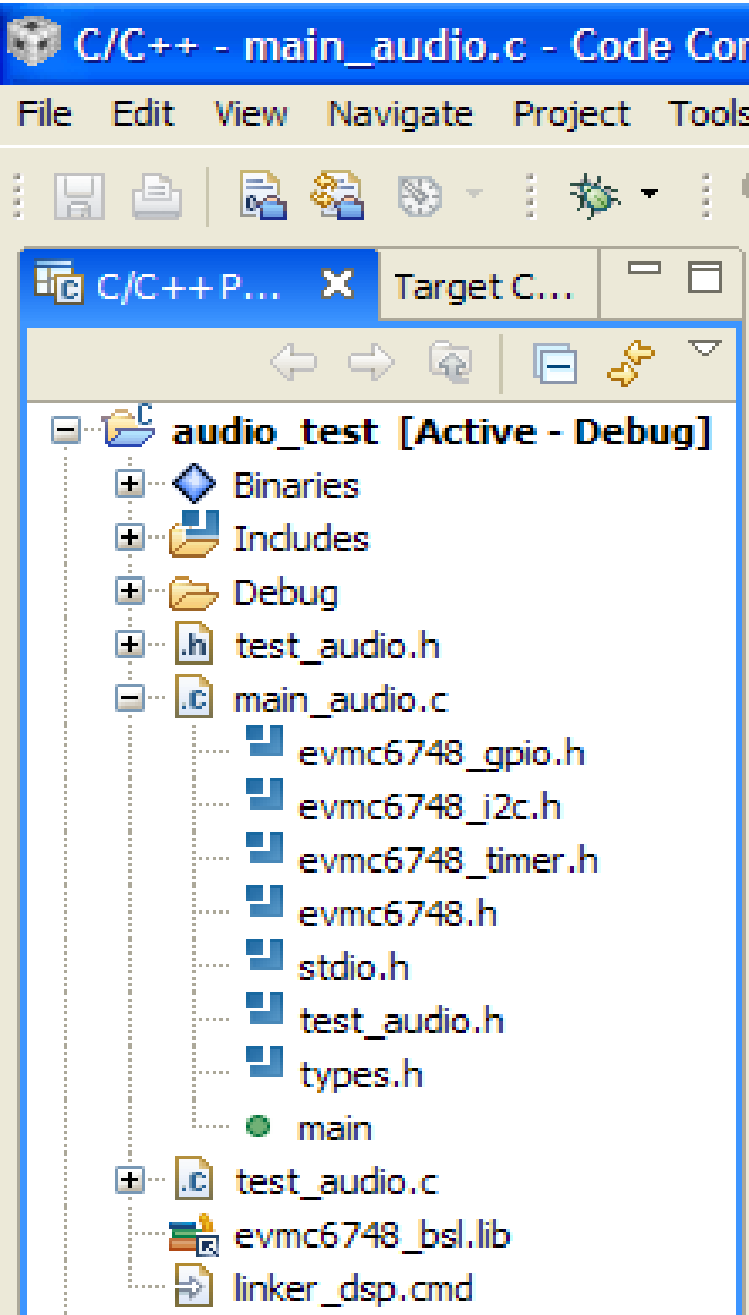
Code Composer Studio Includes:

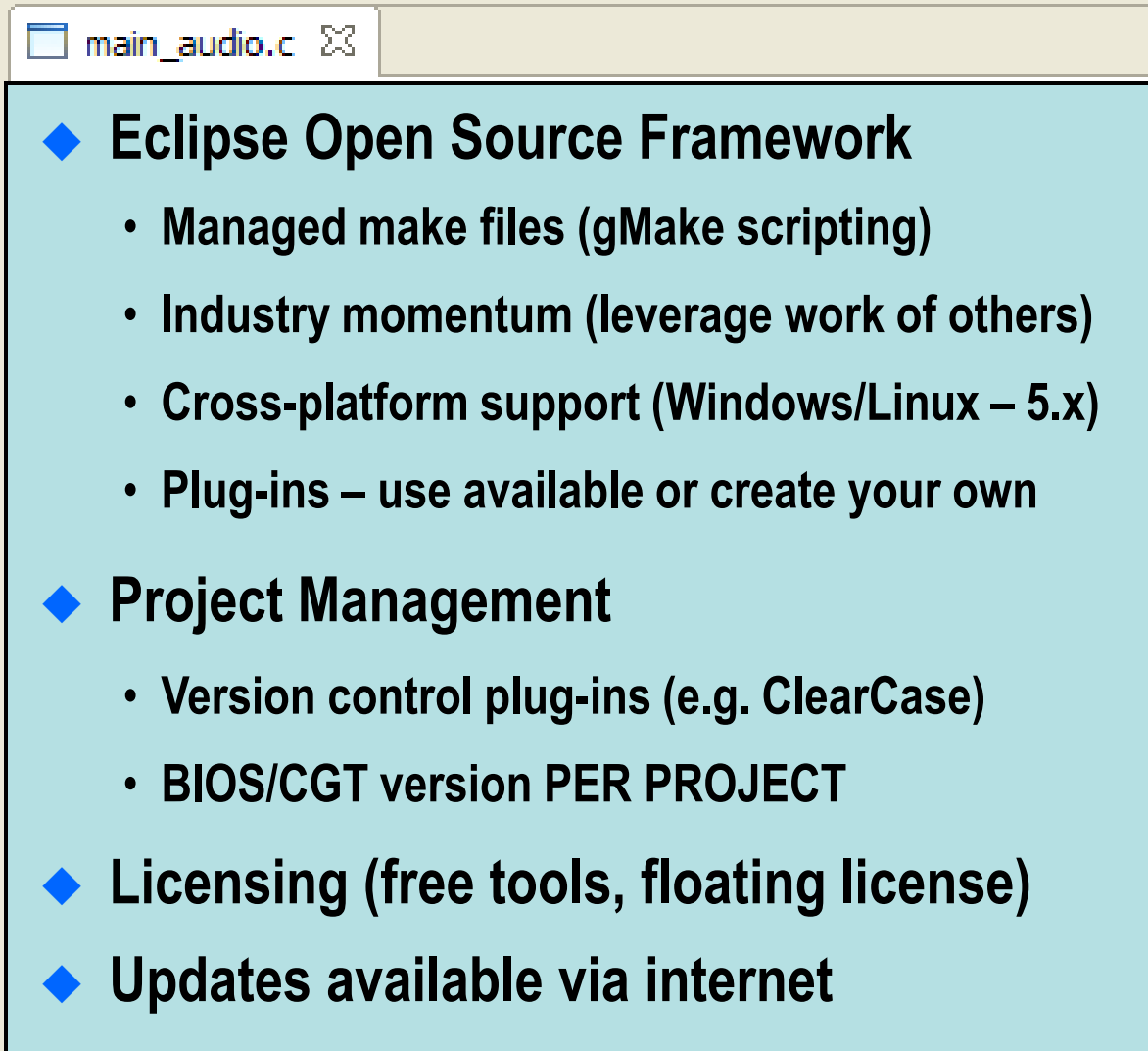
- Integrated Edit / Debug GUI
- Simulator
- Code Generation Tools
- SYS/BIOS: Real-time kernel
  Real-time analysis

CCSv5 “GUI” Environment – Space Saving



CCSv5 (Eclipse) Benefits

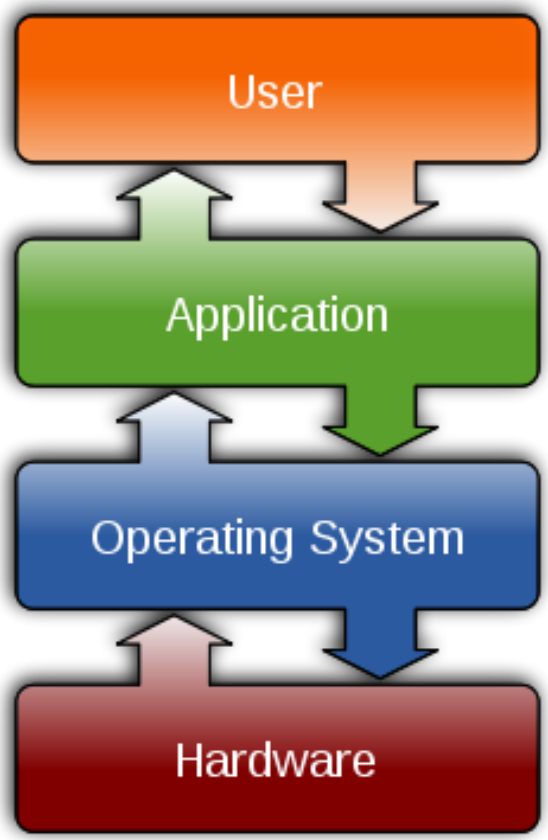




- **Eclipse Open Source Framework**
  - Managed make files (gMake scripting)
  - Industry momentum (leverage work of others)
  - Cross-platform support (Windows/Linux – 5.x)
  - Plug-ins – use available or create your own
- **Project Management**
  - Version control plug-ins (e.g. ClearCase)
  - BIOS/CGT version PER PROJECT
- **Licensing (free tools, floating license)**
- **Updates available via internet**

Outline

- Intro to CCSv5
  - Functional Overview
  - **Perspectives**
  - Projects
  - Target Configuration
  - Build Config & Options
  - Licensing/Pricing
  - CCSv5 – For More Info…

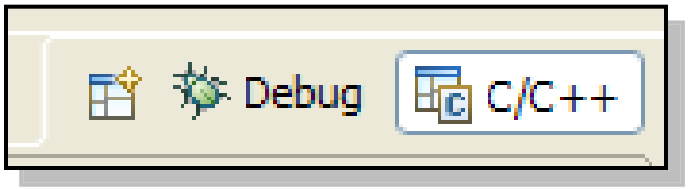


Perspectives

- ***Perspectives*** – a set of windows, views and menus that correspond to a specific set of tasks
- Two ***default perspectives*** are provided with CCSv5:

Debug

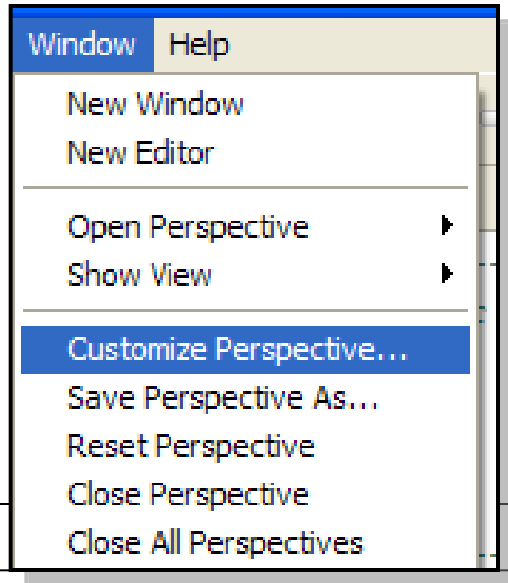
- Debug Views
- Watch/Memory
- Graphs, etc.

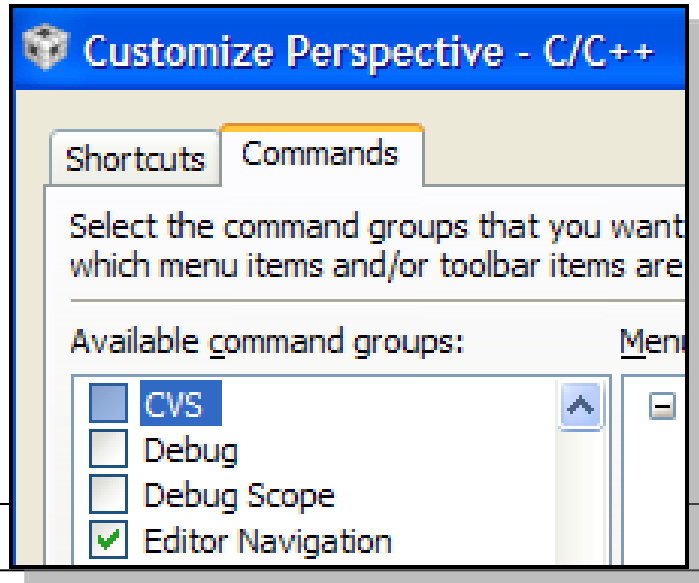


C/C++

- Code Dev't Views
- Project Contents
- Editor

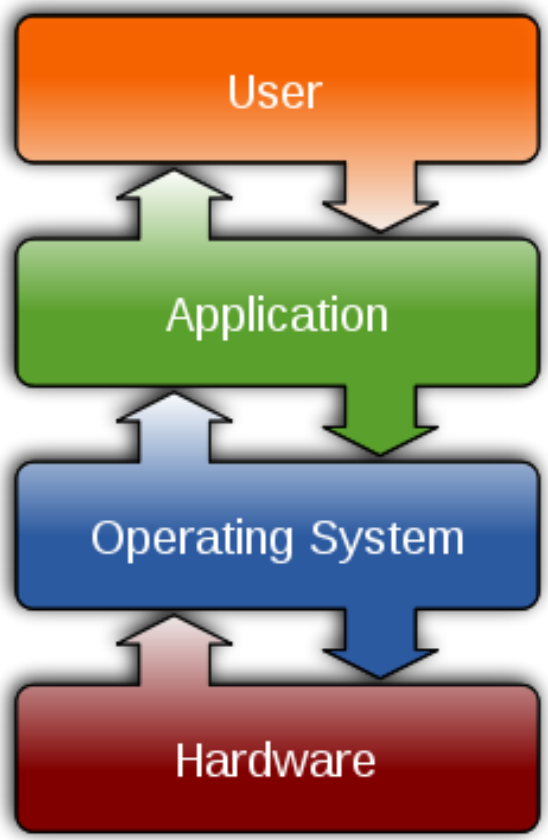
- Users can ***customize perspectives*** and save them:



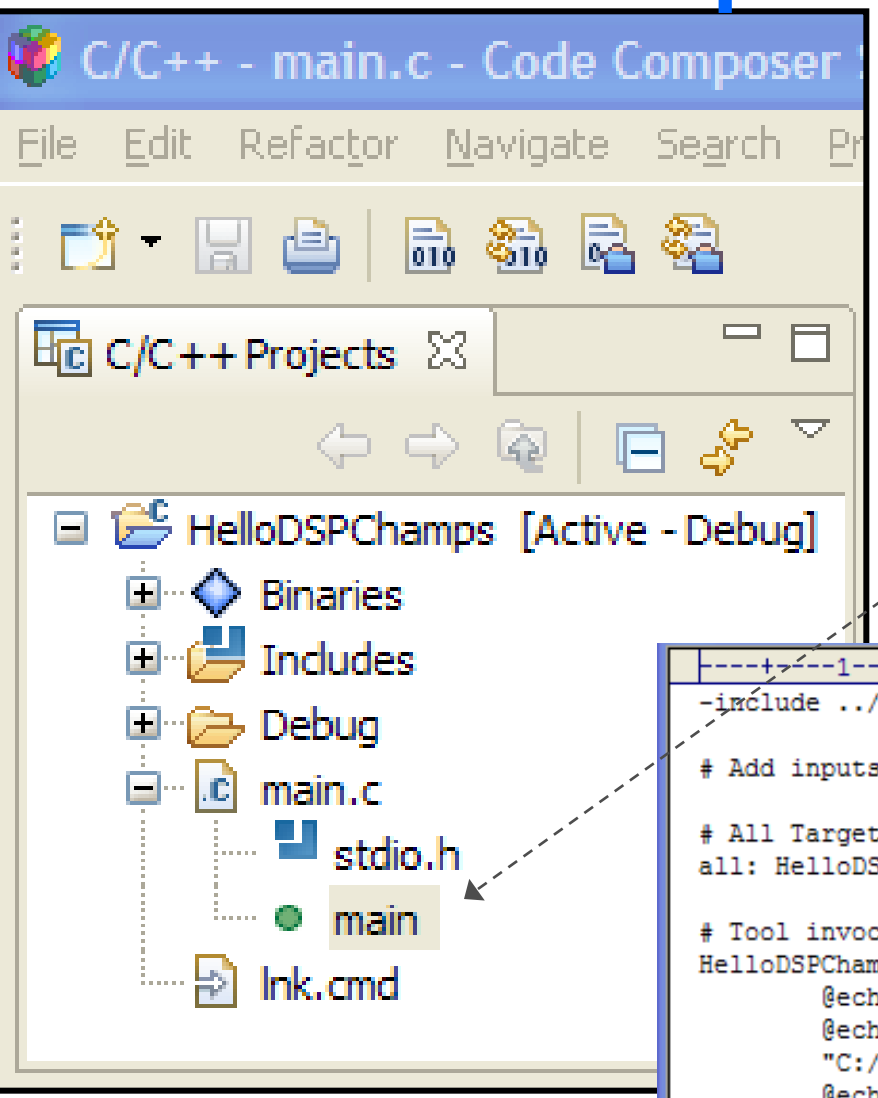


Outline

- Intro to CCSv5
  - Functional Overview
  - Perspectives
  - **Projects**
  - Target Configuration
  - Build Config & Options
  - Licensing/Pricing
  - CCSv5 – For More Info…



Eclipse "Projects"



- **CCSv5 is PROJECT-centric**
- **Eclipse uses *managed makefiles* as their build scripts – as opposed to *pjt* files**
- **Eclipse projects are folder based**
  - "Adding file" copies it to folder
  - "Linking file" references original file
  - Project explorer shows folder contents
- **Project explorer lists functions**

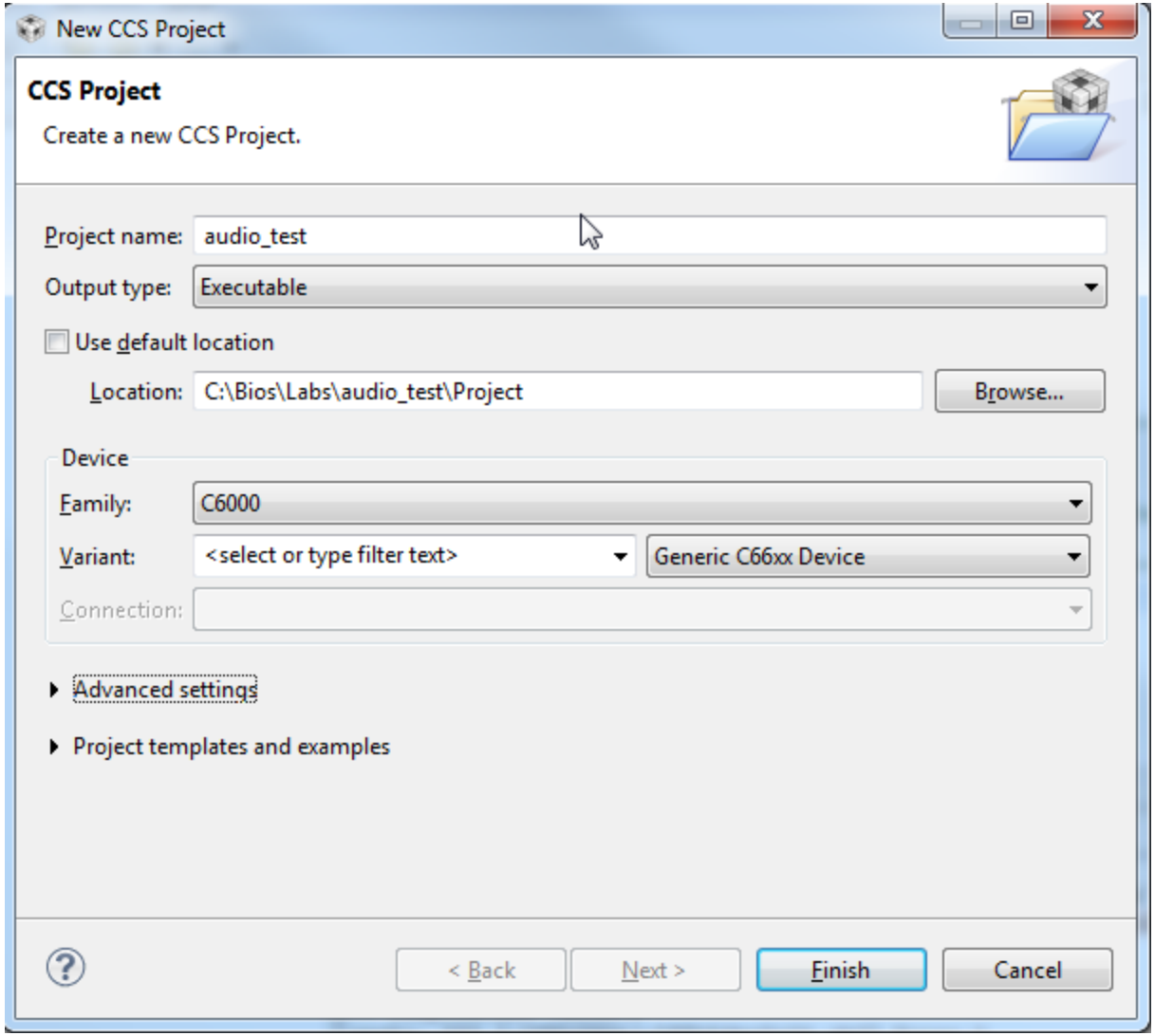
```
-include ../makefile.defs

# Add inputs and outputs from these tool invocations to the build variables

# All Target
all: HelloDSPChamps.out

# Tool invocations
HelloDSPChamps.out: $(GEN_CMDS) $(CMD_SRCS) $(OBJS)
        @echo 'Building target: $@'
        @echo 'Invoking: C6000 Linker v6.1'
        "C:/texas_instruments/ccsv4/tools/compiler/C6000/bin/cl6x" -z --rom_m
        @echo 'Finished building target: $@'
        @echo ' '

# Other Targets
clean:
        -$(RM) $(S62_DEPS)$(ASM_DEPS)$(C55_DEPS)$(C6000_EXECUTABLE_OUTPUTS)$(
```

make file

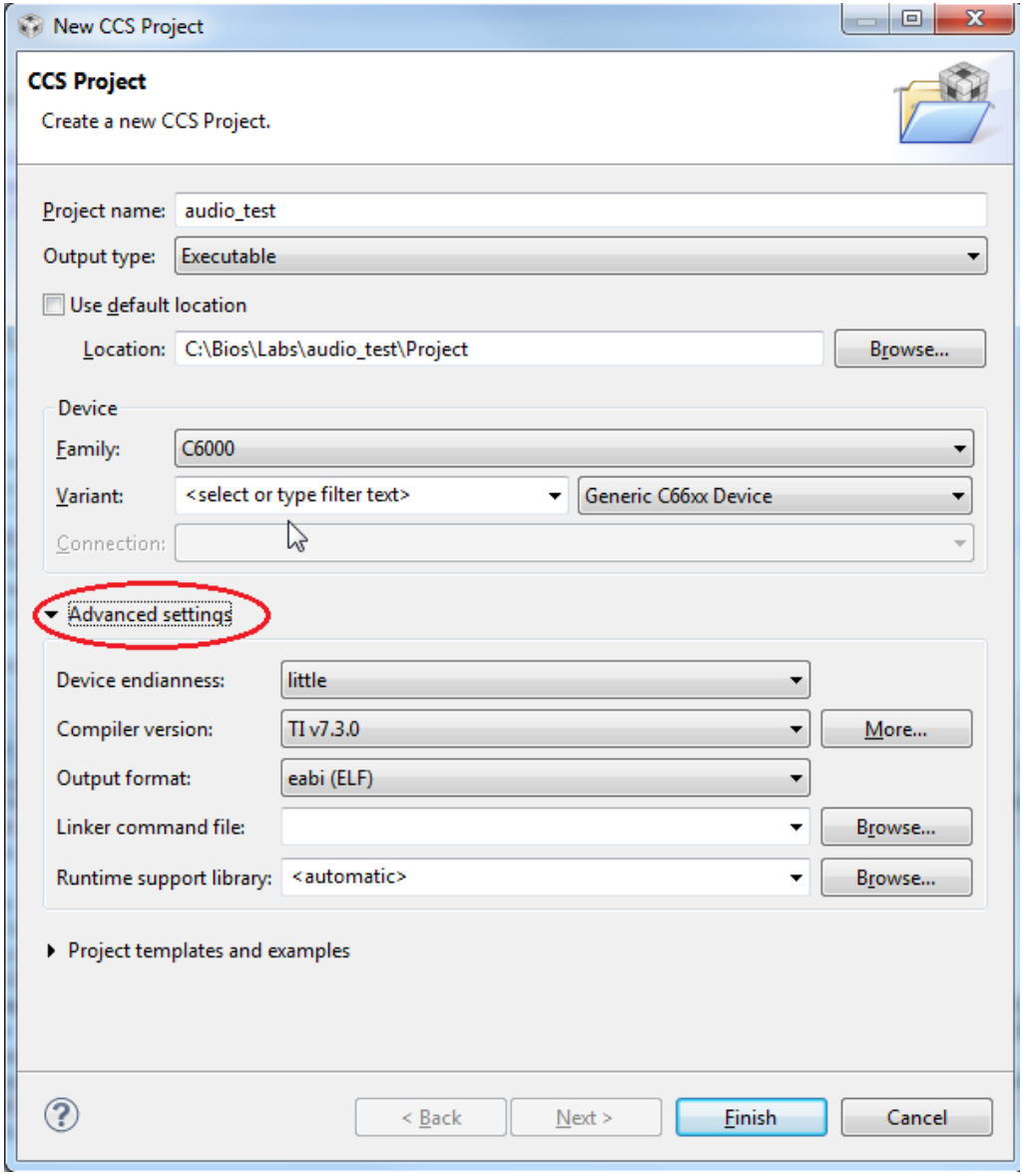
How do we create a NEW project?

Creating a New Project (1)

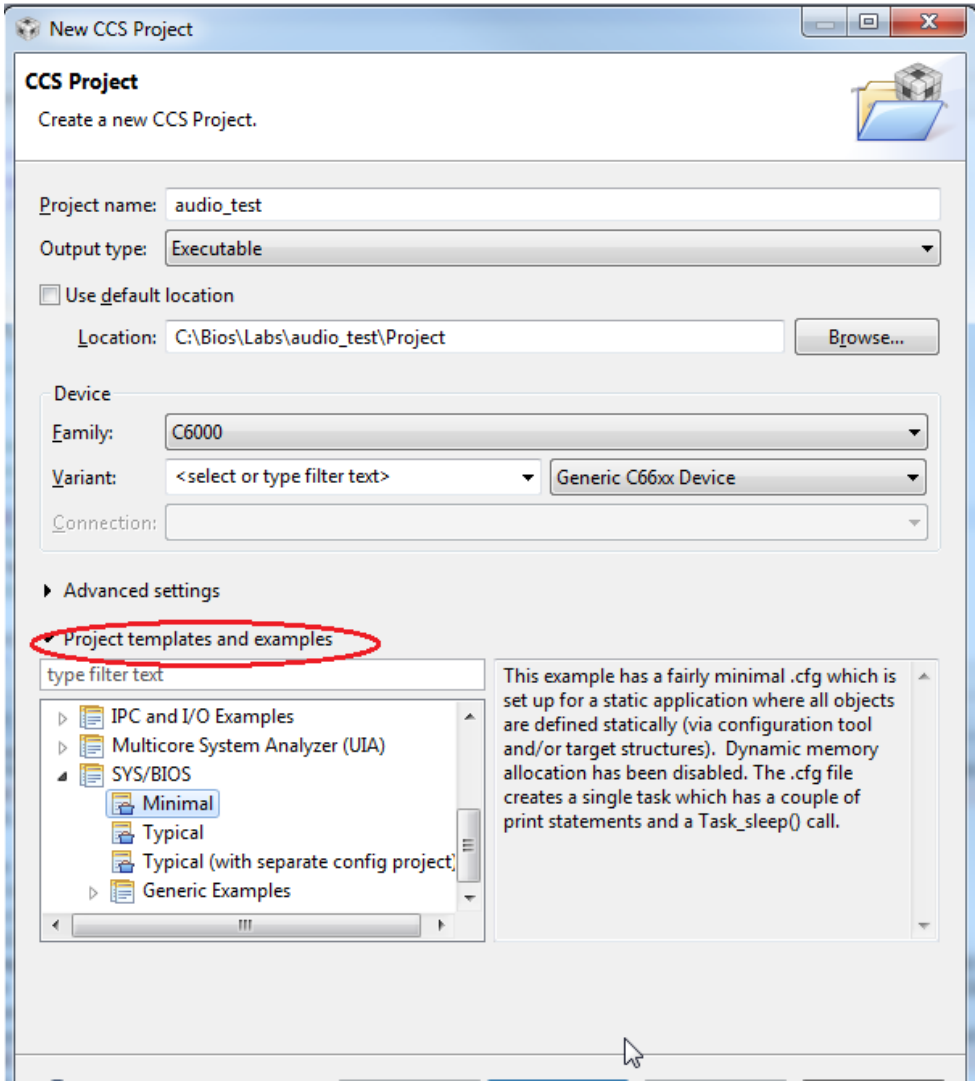
`File → New → CCS Project` (in C++ perspective)



Creating a New Project (2)



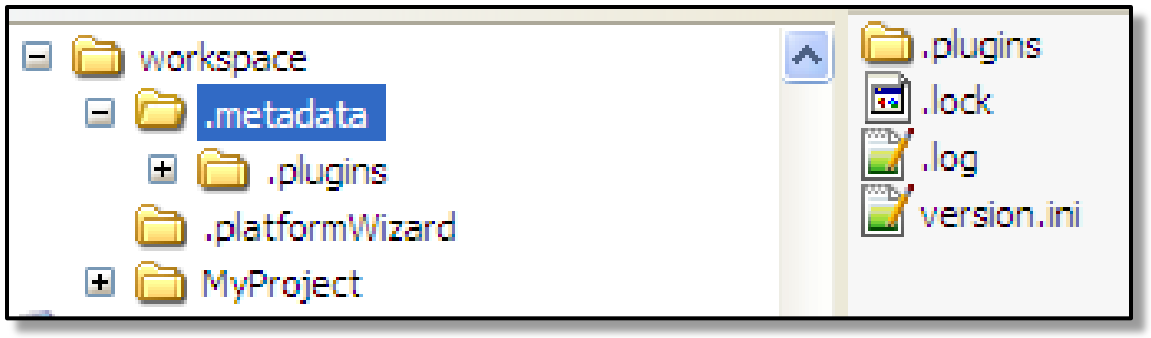
Creating a New Project (3)



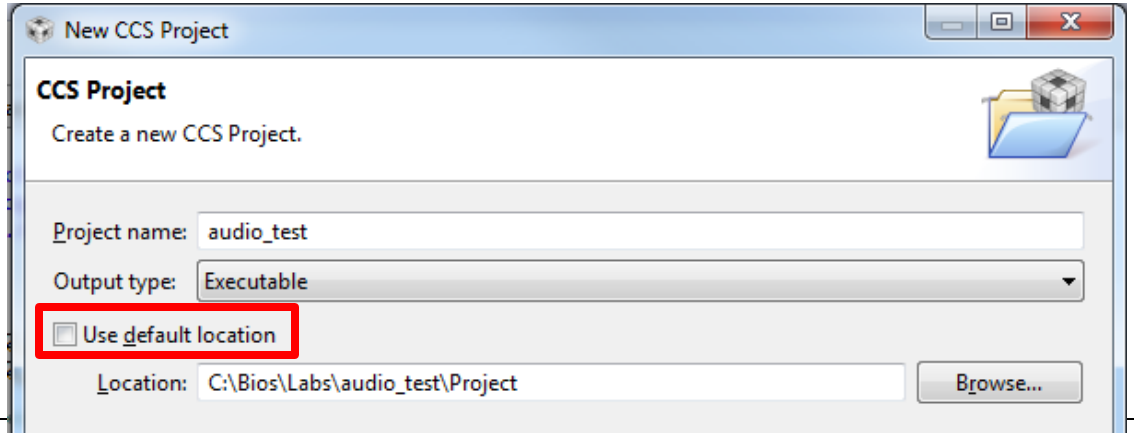
- Not using SYS/BIOS?
  - Choose "Empty Project"
- Using SYS/BIOS?
  - Choose "Minimal" under SYS/BIOS

Eclipse "Workspace"

- ***Workspace*** – a "container" for Eclipse metadata and the default location for all projects
- Default Location: `\My Documents\workspace`:

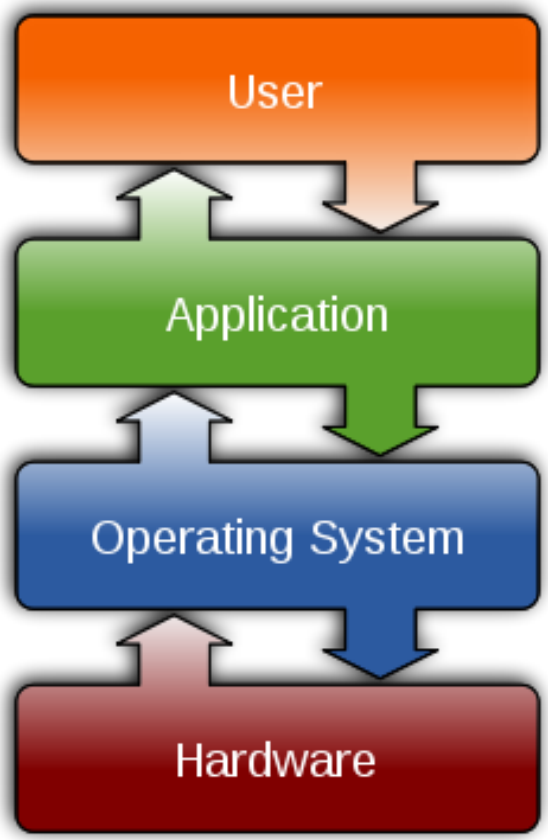


- Can change "default" workspace location if desired
- User can also locate projects in specific folders:



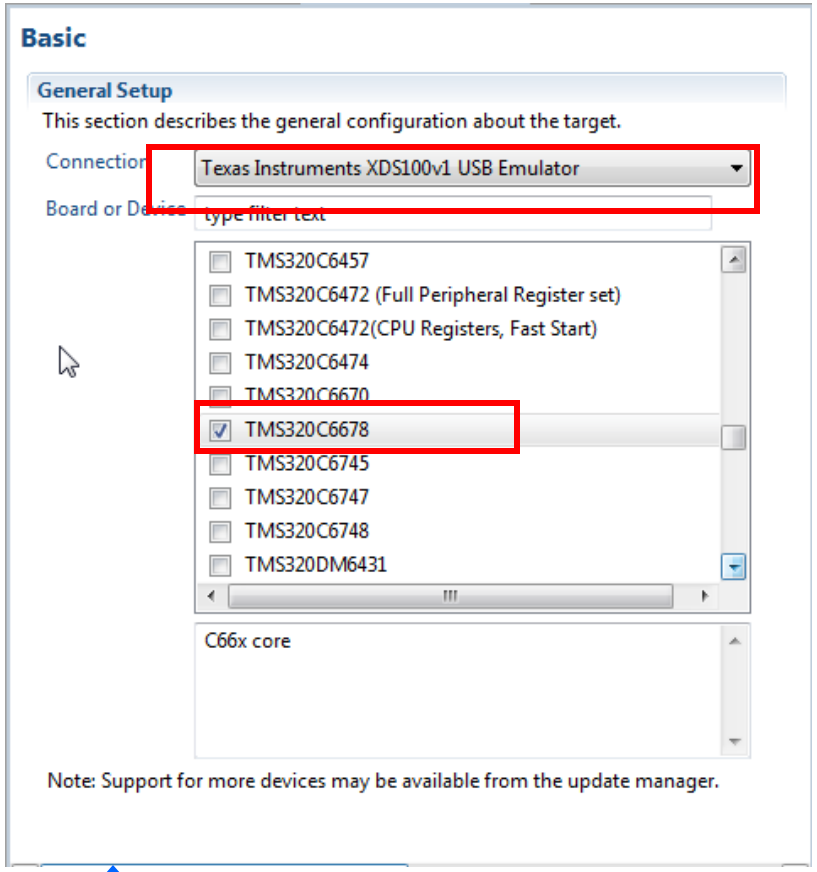
Outline

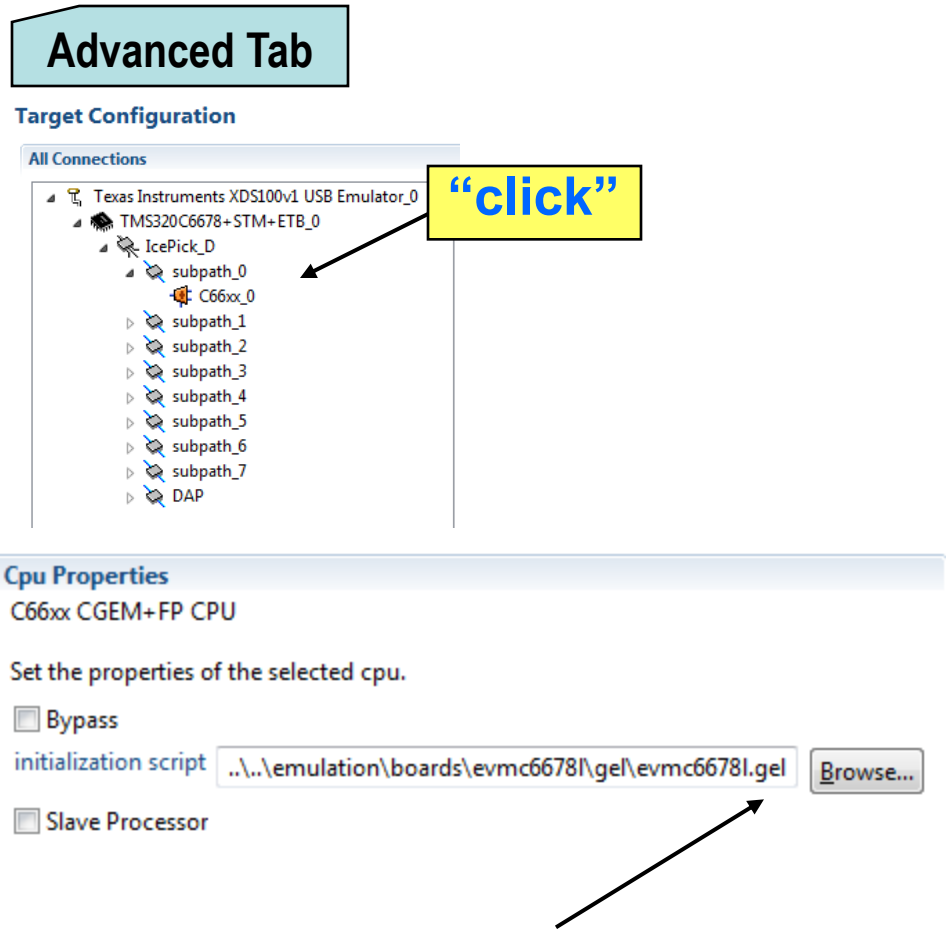
- Intro to CCSv5
  - Functional Overview
  - Perspectives
  - Projects
  - **Target Configuration**
  - Build Config & Options
  - Licensing/Pricing
  - CCSv5 – For More Info…
- Intro to SYS/BIOS



Creating a New Target Config File (.ccxml)

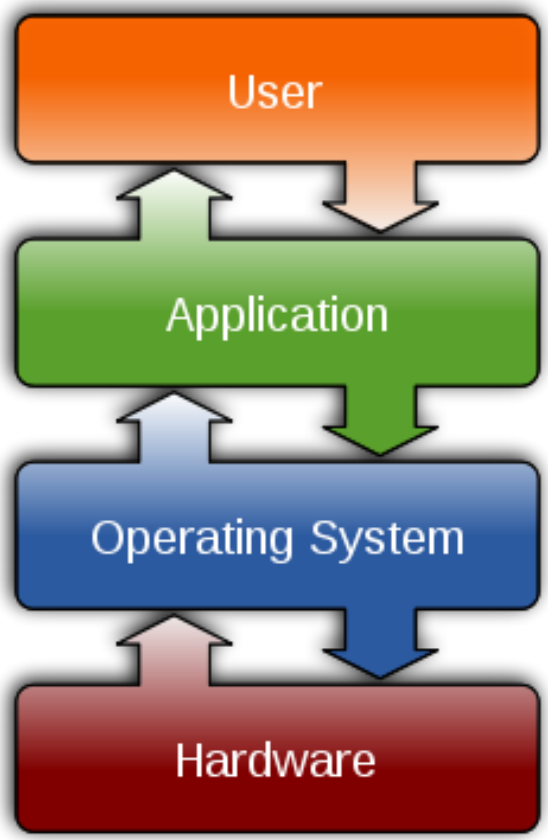
- ***Target Configuration*** – **defines your "target" – i.e. emulator/device used, GEL scripts (replaces the old CCS Setup)**
- **Use on a per-project basis (add to project or create User Defined)**





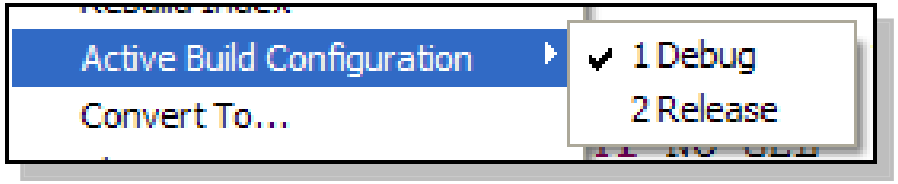
Outline

- Intro to CCSv5
  - Functional Overview
  - Perspectives
  - Projects
  - Target Configuration
  - **Build Config & Options**
  - Licensing/Pricing
  - CCSv5 – For More Info…



Two Default Build Configurations

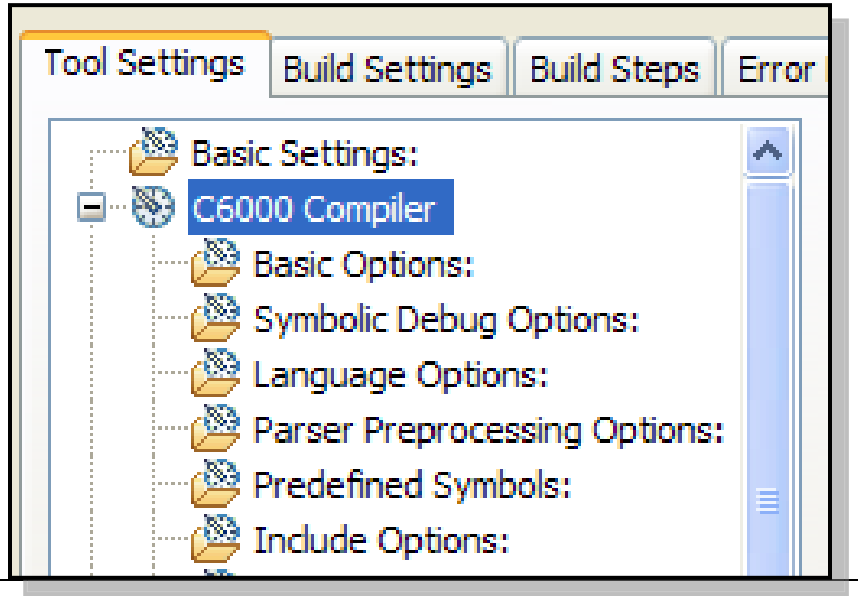
- ***Build Configuration*** **– a set of build options for the compiler and linker (e.g. optimization levels, include DIRs, debug symbols, etc.)**
- **CCSv5 comes std with two DEFAULT build configs: Debug & Release:**



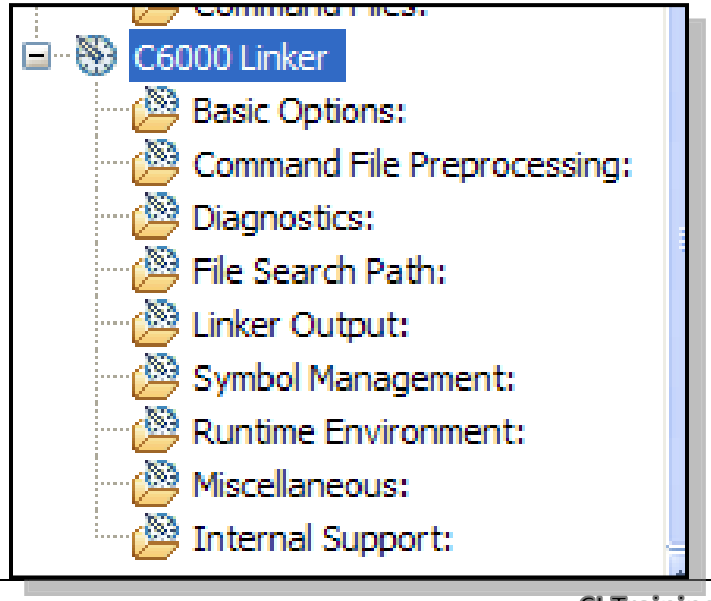
User can create their own config if desired

- **User can modify compiler/linker options via "Build Properties":**

Compiler

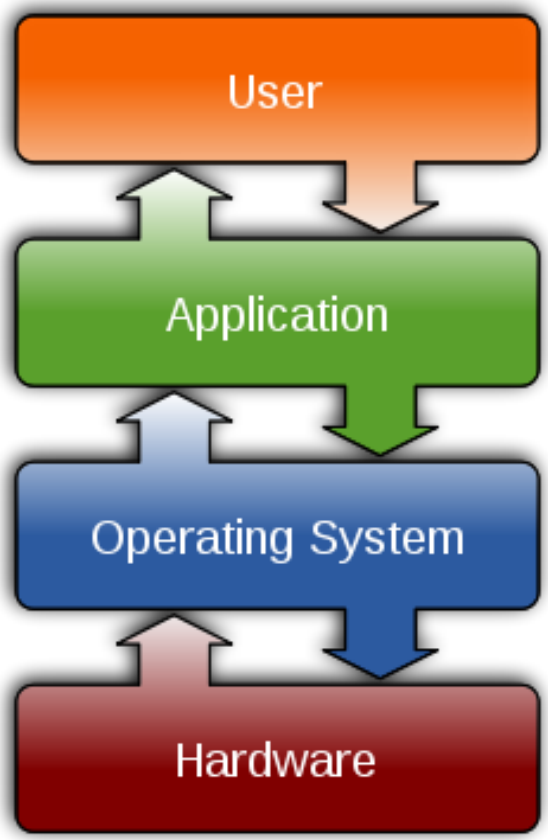


Linker



Outline

- Intro to CCSv5
  - Functional Overview
  - Perspectives
  - Projects
  - Target Configuration
  - Build Config & Options
  - Licensing/Pricing
  - CCSv5 – For More Info…



CCSv5 Licensing & Pricing

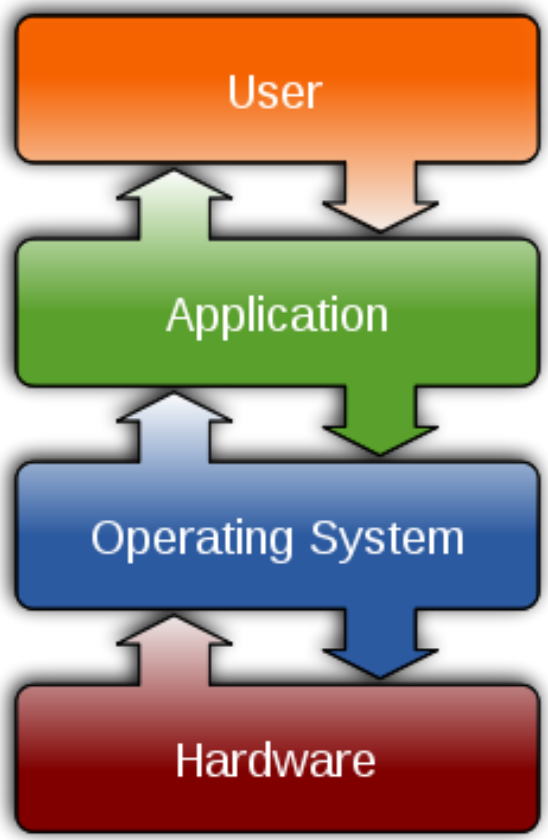
- **Licensing**
  - Wide variety of options (node locked, floating, time based…)
  - All versions (full, DSK, free tools) use same image
  - Updates readily available via the internet
- **Pricing**
  - Reasonable pricing – includes FREE options noted below

Item	Description	Price
Platinum Eval Tools	Full tools with 30 day limit (all EMU)	FREE
Platinum Bundle	EVM, sim, XDS100 use	FREE ☺
Platinum Node Locked	Full tools tied to a machine	$495 (1)
Platinum Floating	Full tools shared across machines	$795 (1)
Microcontroller Core	MSP/C2000 code size limited	FREE
Microcontroller Node Locked	MSP/C2000	$445

☺ - recommended option: purchase Dev Kit, use XDS100v1-2, & Free CCSv5

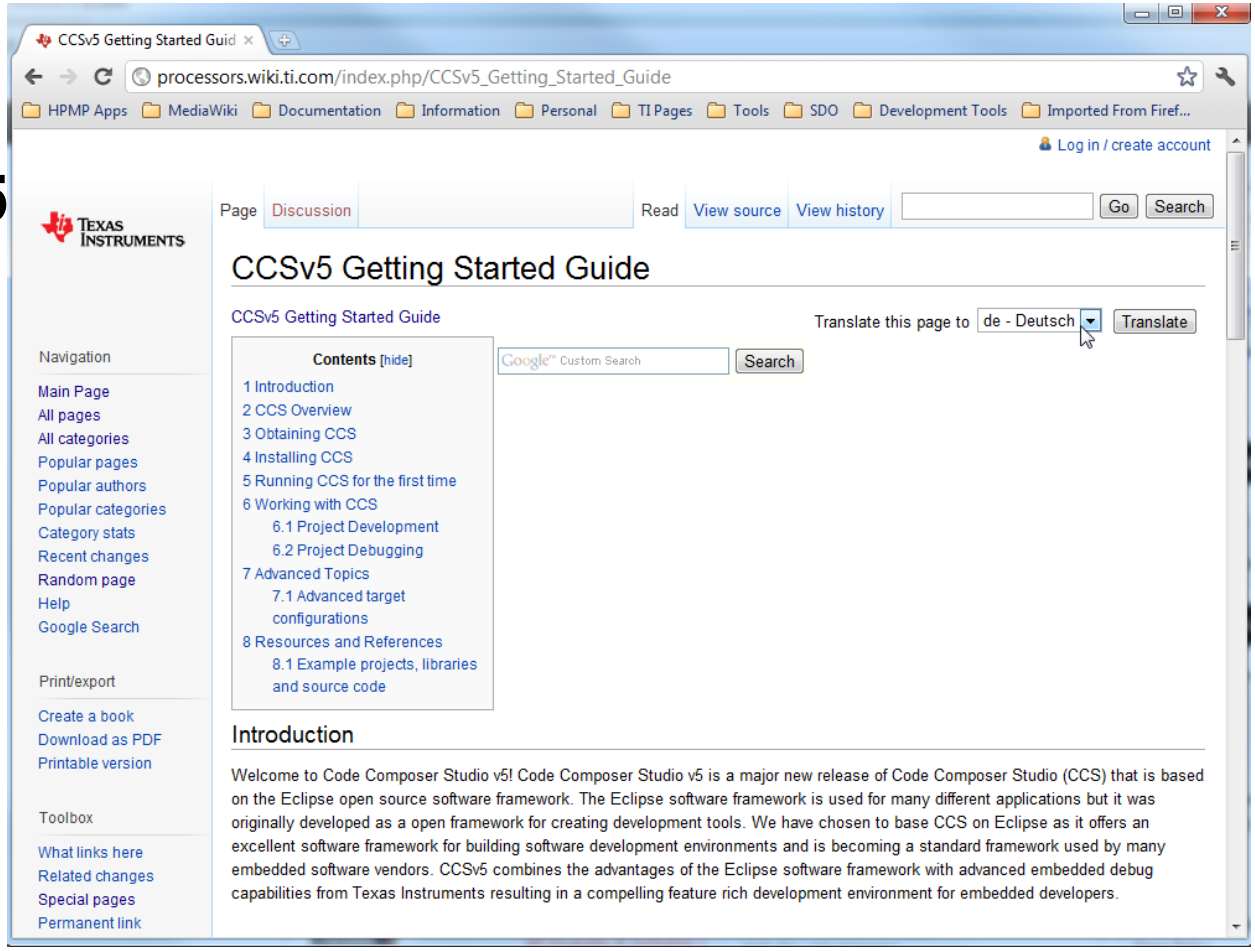
Outline

- Intro to CCSv5
  - Functional Overview
  - Perspectives
  - Projects
  - Target Configuration
  - Build Config & Options
  - Licensing/Pricing
  - CCSv5– For More Info…



CCSv5 – For More Info…

- **Links for:**
  - Downloading CCSv5
  - Installation Help
  - Licensing
  - Tutorials
  - BIOS Projects
  - Etc.



Questions?