Code Composer Studio v5 Workshop

What is Code Composer Studio?

- **Integrated development environment for TI embedded processors**
  – Includes debugger, compiler, editor, operating system…
  – The IDE is built on the Eclipse open source software framework
  – Extended by TI to support device capabilities
- **CCSv5 is based on "off the shelf" Eclipse**
  – Going forward CCS will use unmodified versions of Eclipse
    - TI contributes changes directly to the open source community
  – Drop in Eclipse plug-ins from other vendors or take TI tools and drop them into an existing Eclipse environment
  – Users can take advantage of all the latest improvements in Eclipse
- **Integrate additional tools**
  – OS application development tools (Linux, Android…)
  – Code analysis, source control…

Upgraded user interface for fast, intuitive and easy development

Simplified user interface shows developers what and when features are needed.

Resource Explorer facilitates use of example code.

TI Resource Explorer

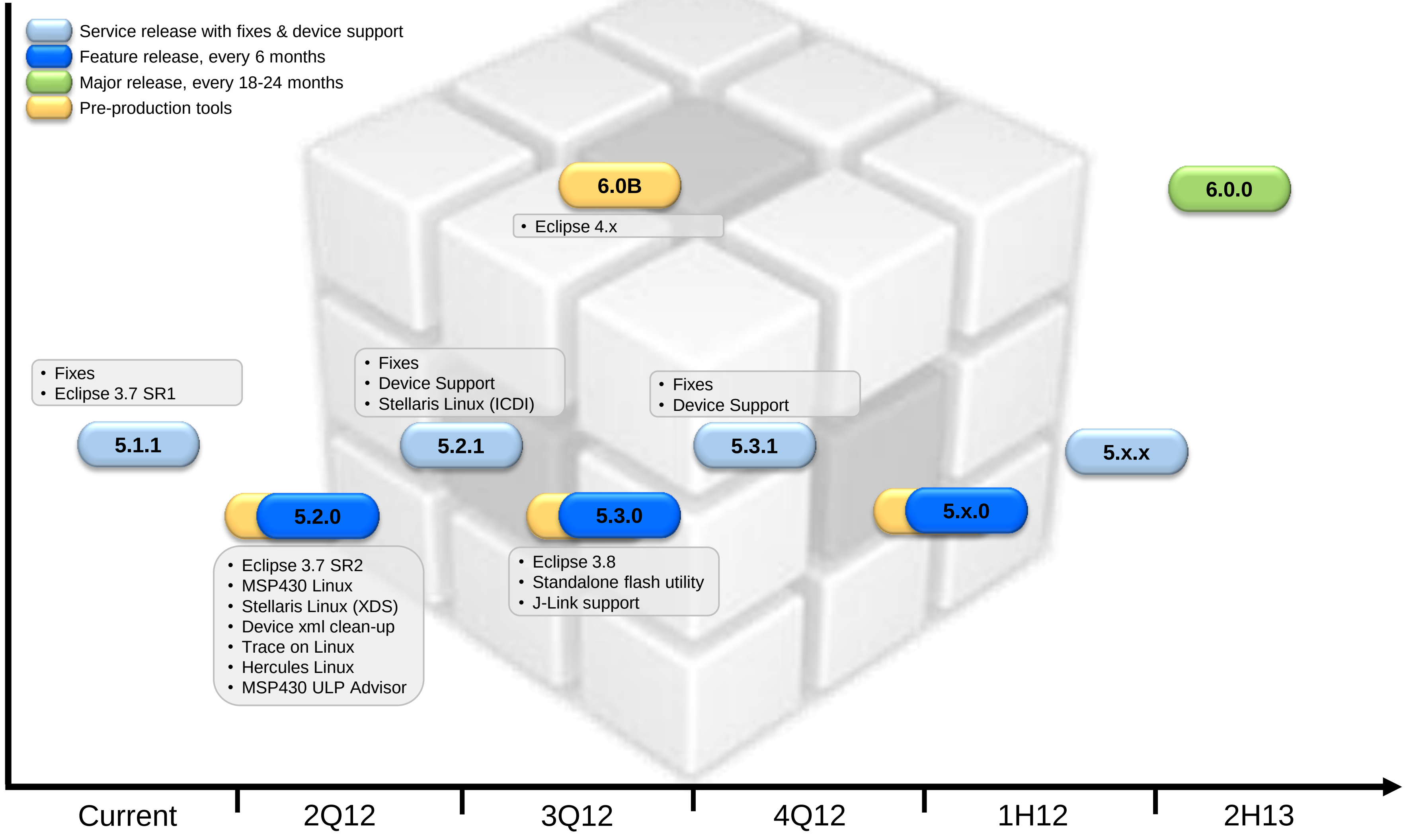
Development tools for Windows and now Linux operating systems.*

Eclipse open source framework 3.7 enables customization via latest third-party plug-ins.

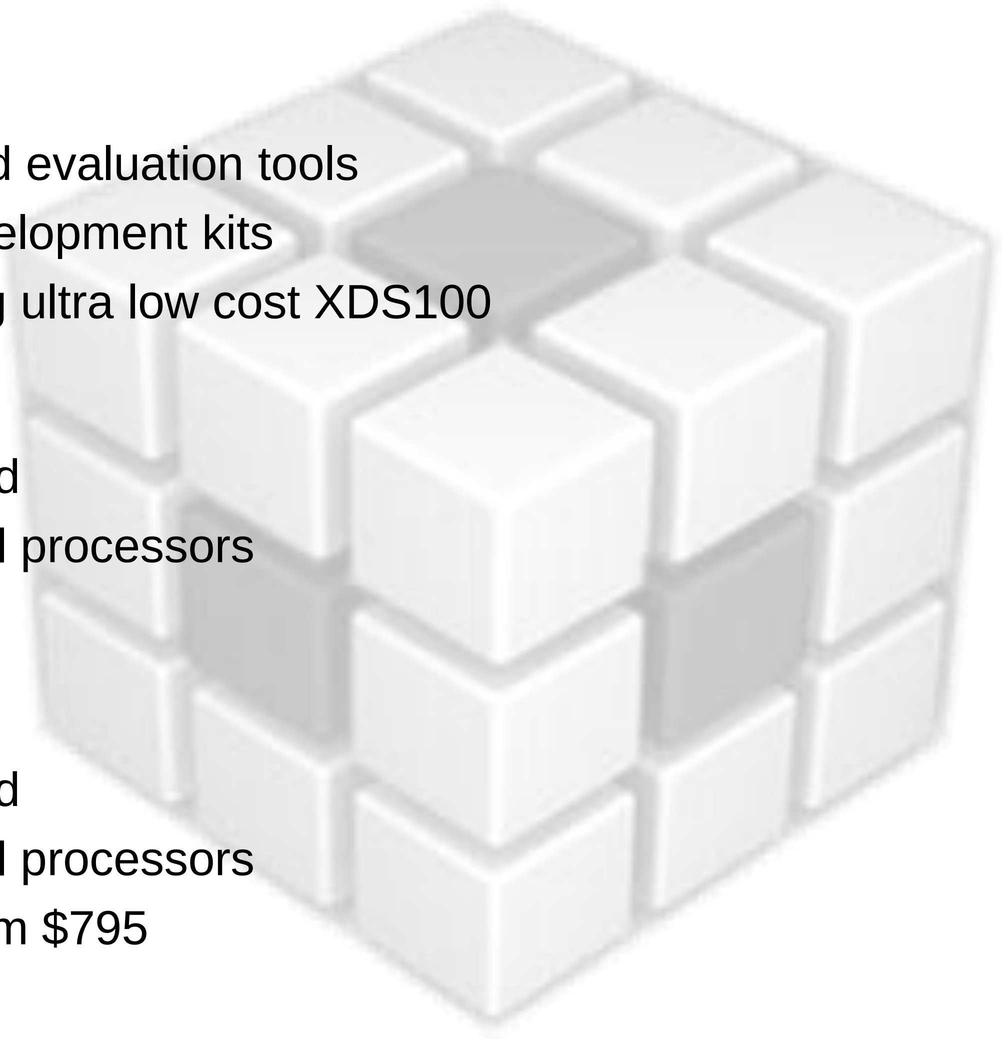
Video tutorials explain how to get the most out of features.

*MSP430, Stellaris MCU support available early 2012

Code Composer Studio™ Roadmap

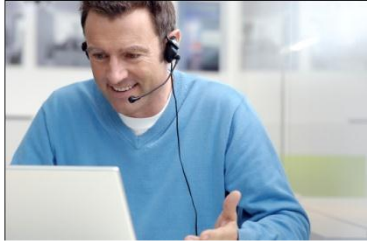


Pricing



- **Free**
  - Time limited evaluation tools
  - Tied to development kits
  - When using ultra low cost XDS100
- **Node locked**
  - Full featured
  - Supports all processors
  - $495
- **Floating**
  - Full featured
  - Supports all processors
  - Starting from $795

TI Software Support



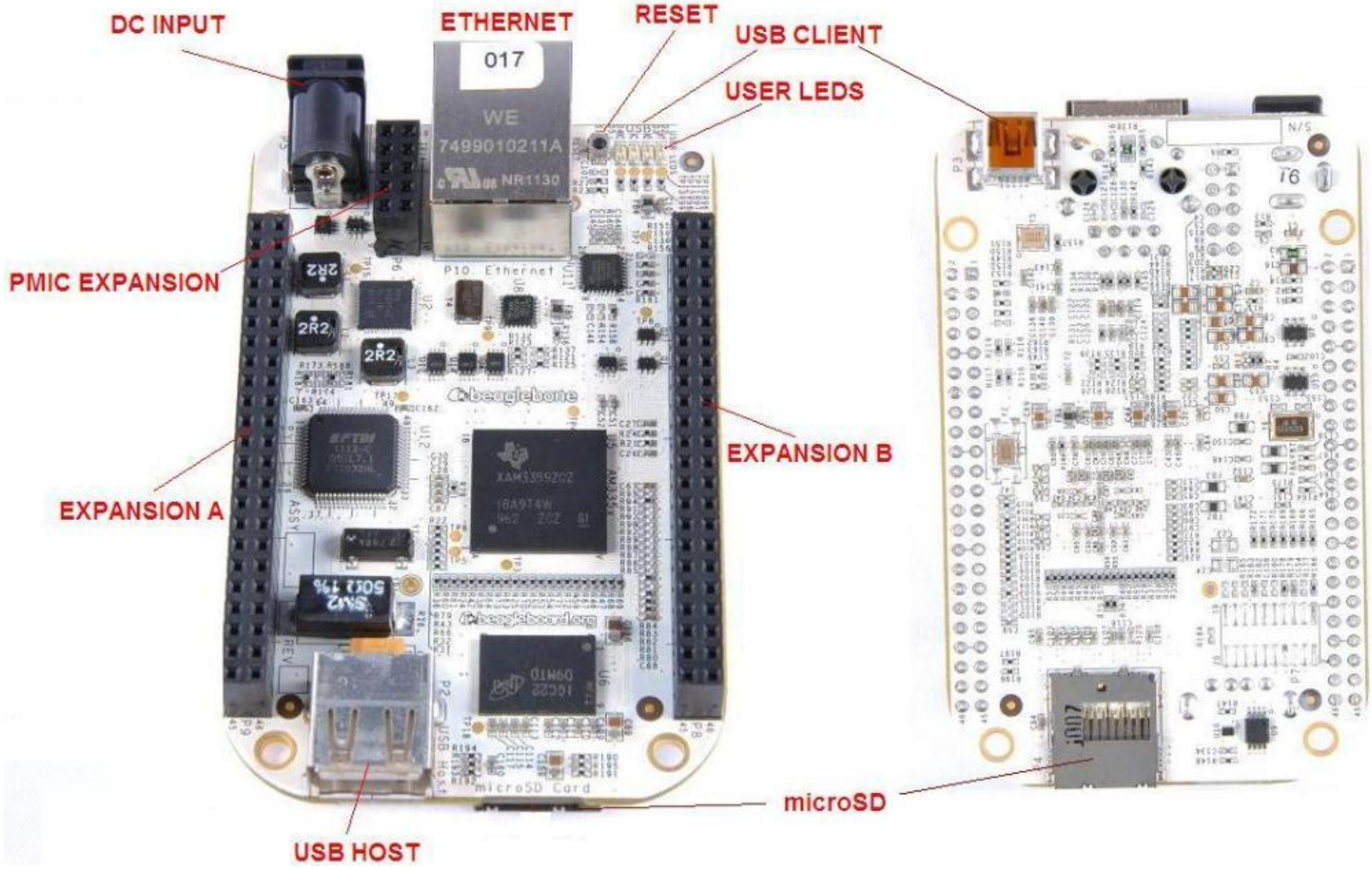
- **Submitting issues**
  - All questions, issues and enhancement requests should be submitted using the e2e Community Forums
- **Benefits of forums**
  - Connects users directly with the engineers developing & supporting TI products
  - Access an extensive knowledge base on TI products
- **Software related forums**
  - Development Tools
    - Any issues questions related to Code Composer Studio (CCS) or TI compilers
  - Embedded Software
    - Linux, Android, WinCE, BIOS and Codecs forums
- **Before posting a question check if it is already answered**
  - Check the FAQs and topics on the Embedded Processing Mediawiki
  - Search the e2e forums
- **Check status of issues**
  - Use SDOWP to see what release an issue will be addressed as well as the list of issues fixed in specific releases

GETTING STARTED WITH CCSV5 AND AM3359 BEAGLEBONE

What is AM3359 Beaglebone?

- **AM3359 Cortex A8 microprocessor**
- **Quick and easy evaluation of all of the advanced capabilities of the community board for just $89**
- **Multiple graphical development platforms for easy development (Cloud9, GateOne, CCS)**
- **Fits in an Altoids box!**
- **On-board emulation, access to several I/O pins (ADC, PWM, UART, I^{2}C, SPI, etc.)**
- **Main website: http://www.beagleboard.org**
  - Detailed example software and documentation
  - Complete hardware schematics and board files

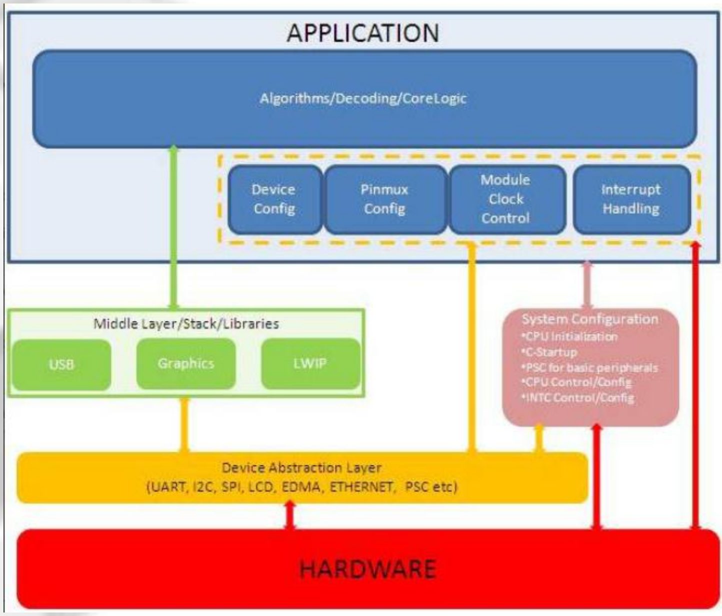
AM3359 BeagleBone



What is Starterware?

- **A quick and easy way to start development without a High level Operating System (Linux, WinCE, etc.)**
- **Contains example code for several peripherals (GPIO, UART, Ethernet, RTC, etc.)**
- **Supports multiple development platforms (CCS, IAR, GCC)**
- **Supports several processors:**
  - C6748, OMAPL138, AM1808, AM335x
- **Can be downloaded free at:**

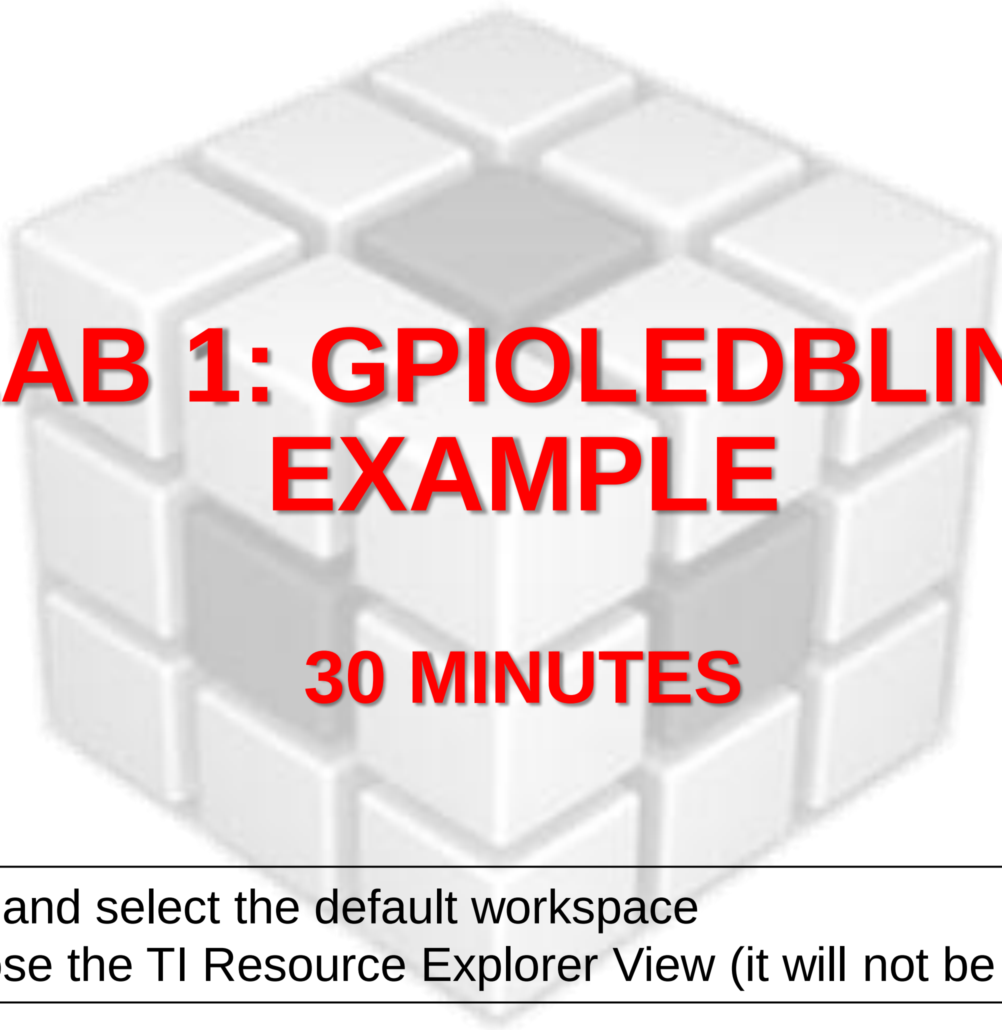
  http://processors.wiki.ti.com/index.php/StarterWare



GPIO LED BLINK EXAMPLE: BASIC PROJECT DEBUGGING

GPIO LED Blink Example: Exercise Summary

- **Key Objectives**
  - Create and build a simple program to blink USR2 LED (D4)
  - Start a debug session and load/flash the program on the BeagleBone
  - Run the program to blink USR2 LED
- **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



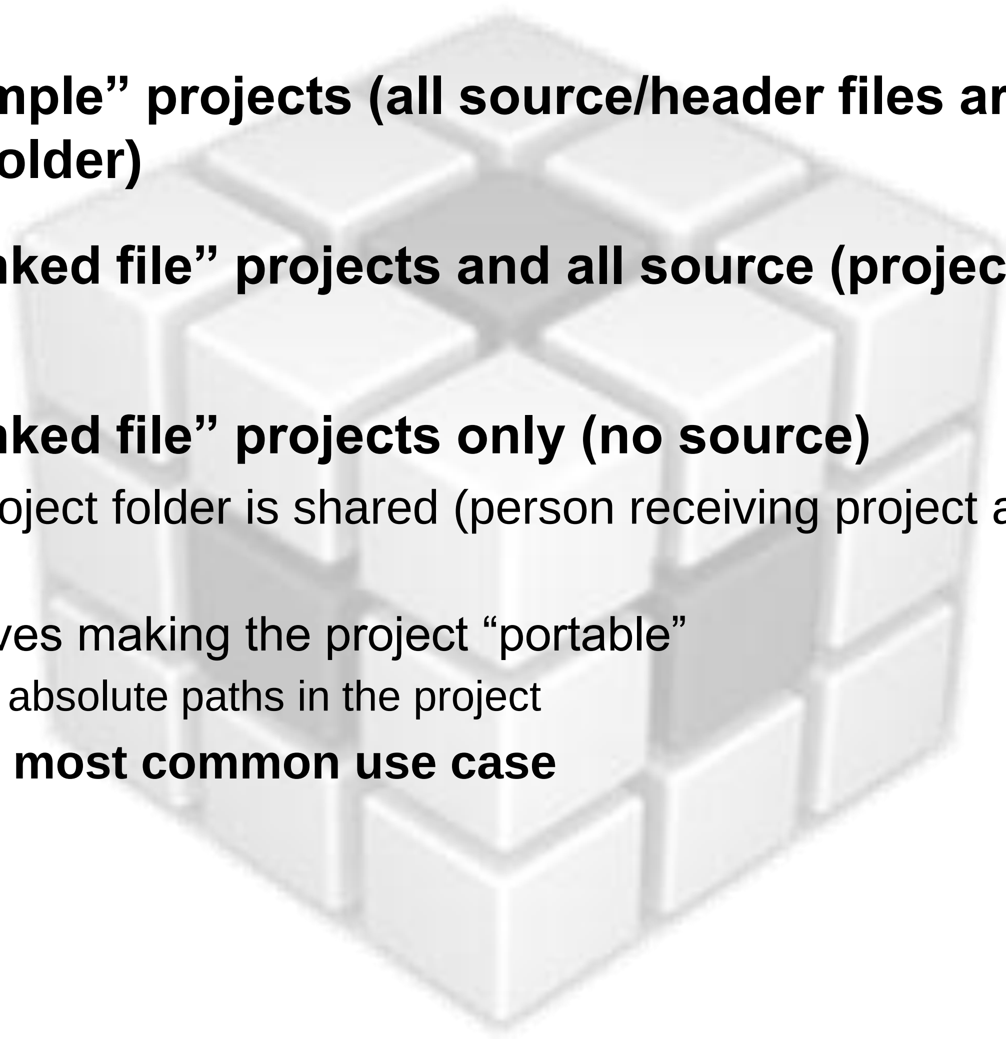
LAB 1: GPIOLEDBLINK EXAMPLE

30 MINUTES

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

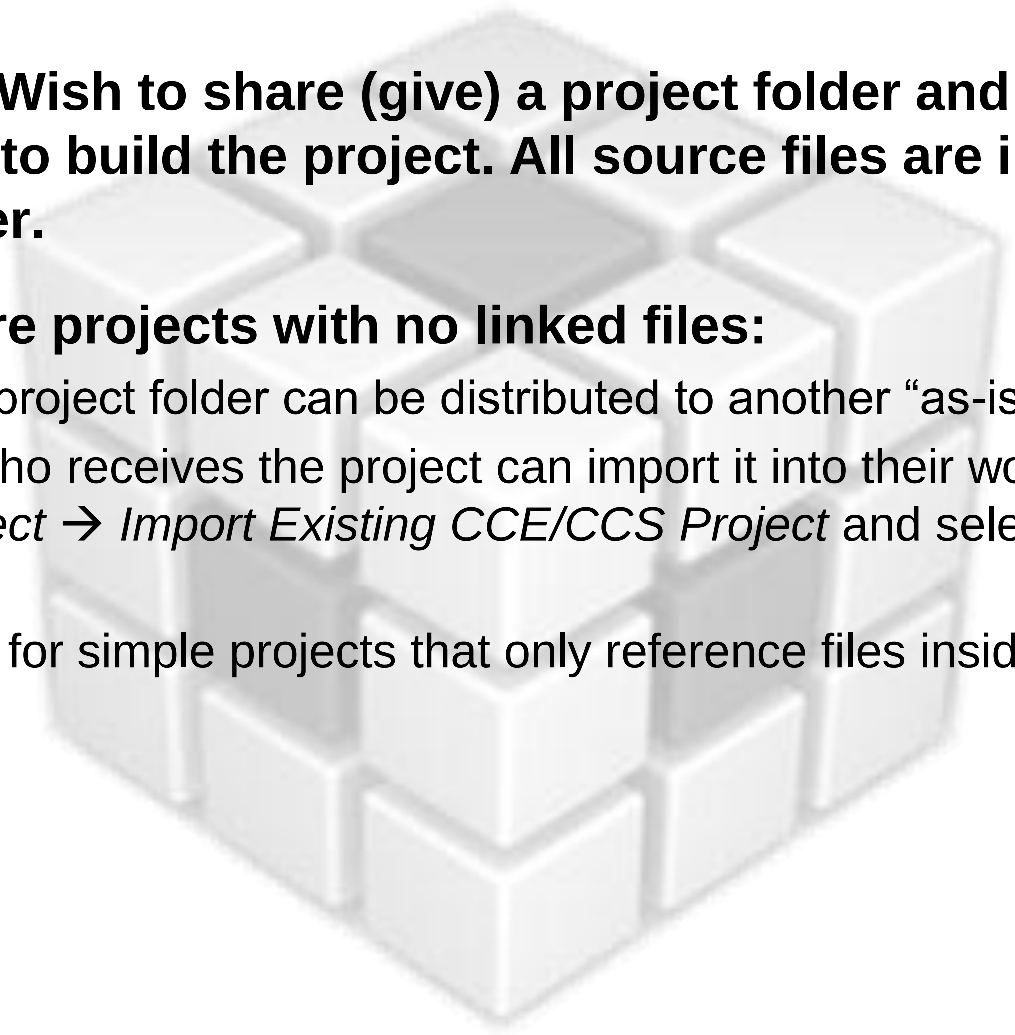
SHARING PROJECTS

Sharing Projects



- **Sharing "Simple" projects (all source/header files are contained in the project folder)**
- **Sharing "Linked file" projects and all source (project uses linked files)**
- **Sharing "Linked file" projects only (no source)**
  - Only the project folder is shared (person receiving project already has all source)
  - Effort involves making the project "portable"
    - Eliminate absolute paths in the project
  - **This is the most common use case**

Sharing Projects – Simple Projects



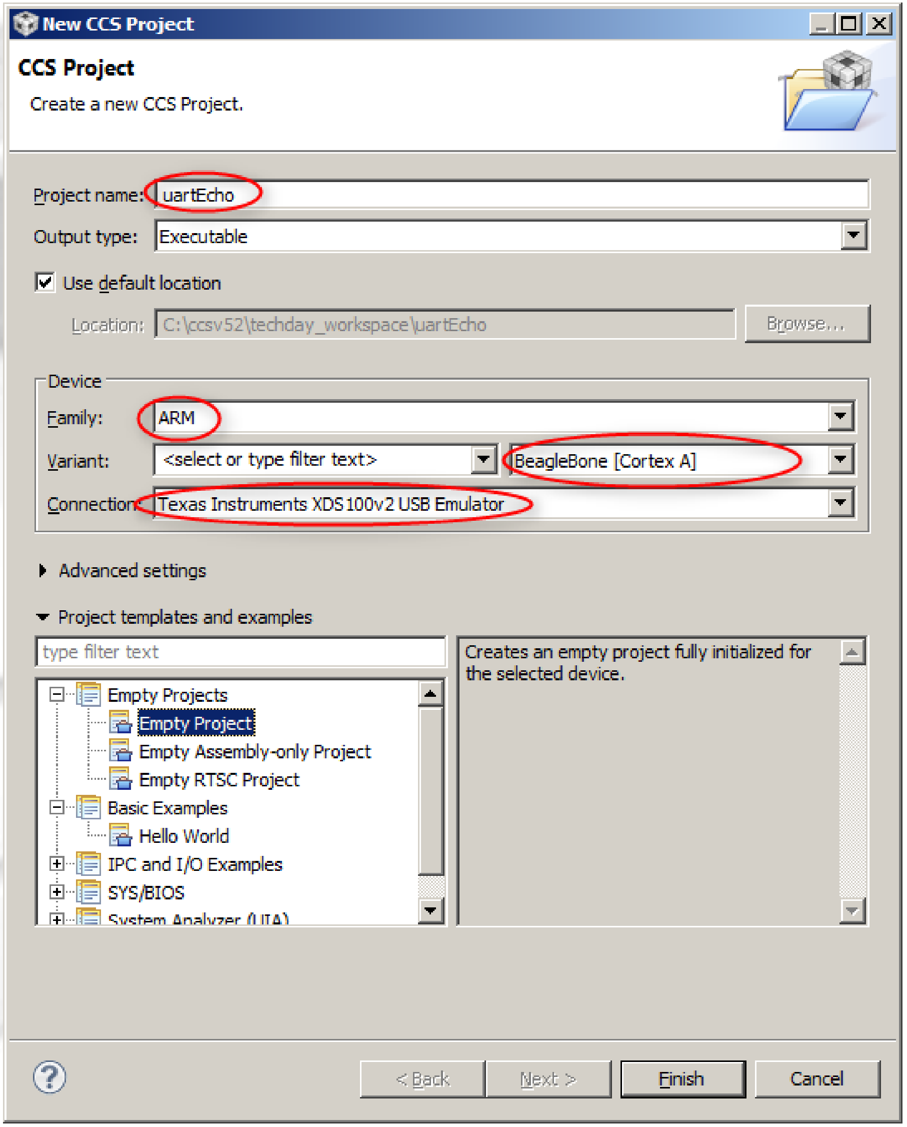
- **<u>USE CASE</u>: Wish to share (give) a project folder and all needed source files to build the project. All source files are inside the project folder.**
- **Easy to share projects with no linked files:**
  - The entire project folder can be distributed to another "as-is"
  - The user who receives 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

Sharing Projects – "Linked file" Projects

- **USE CASE(S):**
  - 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 ideal for projects maintained in source control

Create a New Project

- **A shared project is created the same way as a regular project**
- **Launch the *New CCS Project* Wizard**
  - Go to menu File → *New* → *CCS Project*
- **Fill in the fields as shown in the right**
- **Click *Finish* when done**
- **Generated project will appear in the Project Explorer view**
- **Remove the generated files <main.c> and <AM3358.cmd> from the project**


New CCS Project

CCS Project

Create a new CCS Project.

Project name: uartEcho

Output type: Executable

☑ Use default location

Location: C:\ccsv52\techday_workspace\uartEcho

Device

Family: ARM

Variant: <select or type filter text> BeagleBone [Cortex A]

Connection: Texas Instruments XDS100v2 USB Emulator

Advanced settings

Project templates and examples

type filter text

- Empty Projects
  - Empty Project
  - Empty Assembly-only Project
  - Empty RTSC Project
- Basic Examples
  - Hello World
- IPC and I/O Examples
- SYS/BIOS
- System Analyzer (UIA)

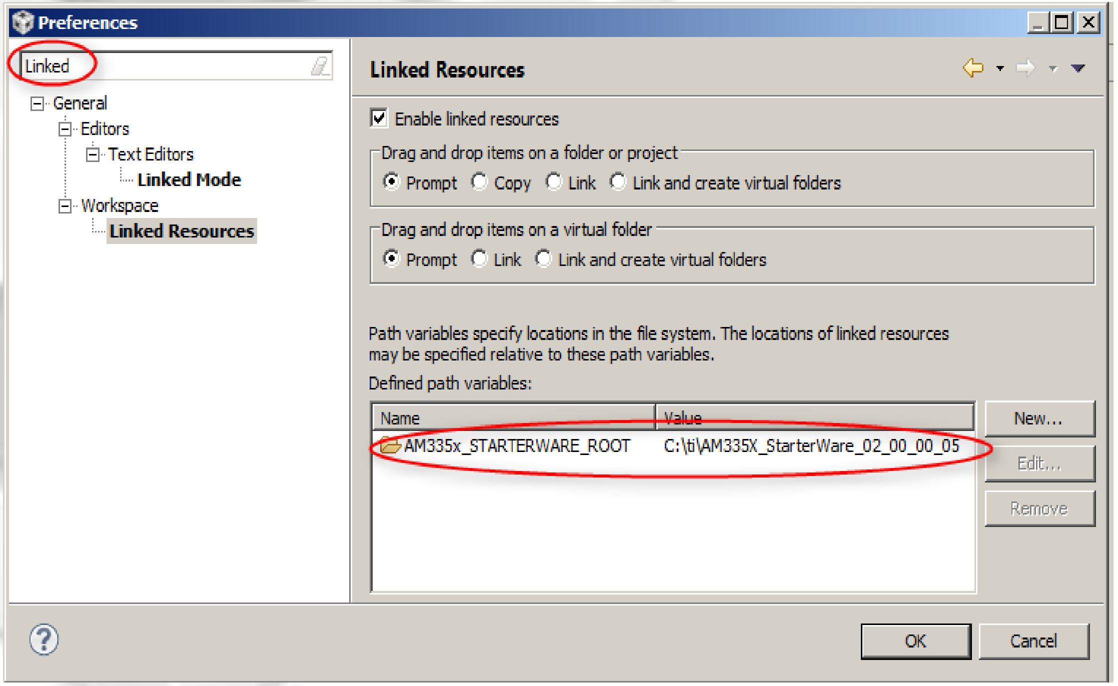
Creates an empty project fully initialized for the selected device.

Create a Linked Resource Path Variable

- **Here we will create the Linked Resource Path Variable which will be used when linking source files (resources) to the project**

- **Open the workspace preferences**
  - Menu *Window* → *Preferences*
- **Go to the *Linked Resources* preferences**
  - Type ‘Linked’ in the filter field to make it easier to find
- **Use the *New* button to create a ‘Linked Resource Variable’ (AM335x_STARTERWARE_ROOT) that points to the root location of the AM335x StarterWare directory**

C:\ti\AM335X_StarterWare_02_00_00_06


Preferences

Linked

- General
  - Editors
    - Text Editors
      - Linked Mode
  - Workspace
    - Linked Resources

Linked Resources

[✓] Enable linked resources

Drag and drop items on a folder or project
[✓ Prompt] Copy Link Link and create virtual folders

Drag and drop items on a virtual folder
[✓ Prompt] Link Link and create virtual folders

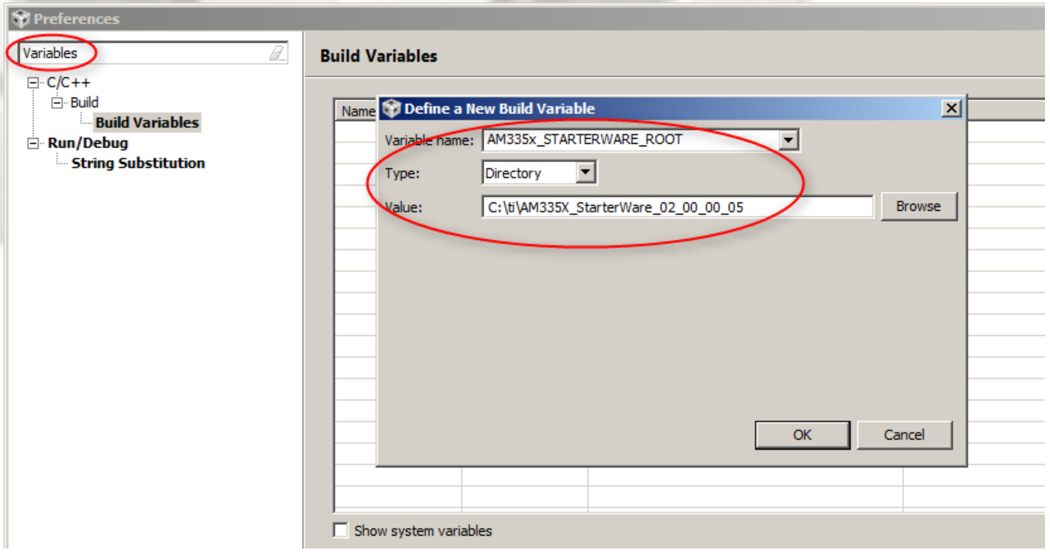
Path variables specify locations in the file system. The locations of linked resources may be specified relative to these path variables.

Defined path variables:

Name	Value
AM335x_STARTERWARE_ROOT	C:\ti\AM335X_StarterWare_02_00_00_05

Create a Build Variable

- **Here we will create the Build Variable which will be used when setting the project's compiler and linker options**
- **Go to the *Build Variables* preferences**
  - Type 'Variables' in the filter field to make it easier to find
- **Build Variables allow you to use variables in the project properties**
  - Linked Resource variables are only used for linked files
- **Use the *Add* button to create a 'Build Variable' (AM335x_STARTERWARE_ROOT) that points to the root location of the AM335x StarterWare directory**
- **Hit *OK* when done**



Variable name: AM335x_STARTERWARE_ROOT
Type: Directory
Value: C:\ti\AM335X_StarterWare_02_00_00_05

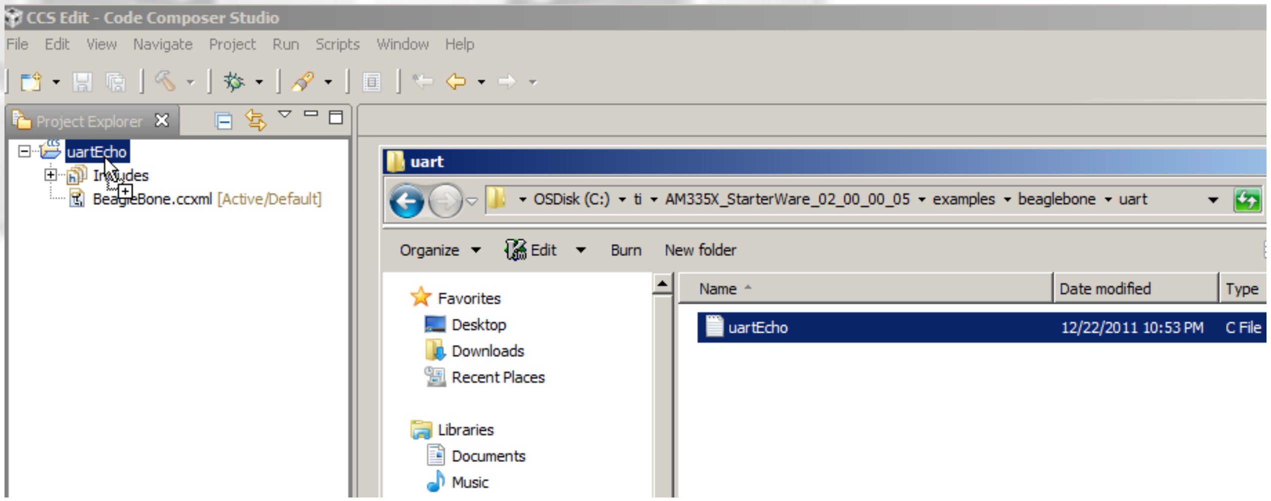
Link Source Files to Project

- **Here we will link the source file relative to the Linked Resource Path Variable previously created**
- **Open Windows Explorer and browse to:**

  C:\ti\AM335X_StarterWare_02_00_00_06\examples\beaglebone\uart

  – Drag and drop the following file into the new project in the CCS Project Explorer view

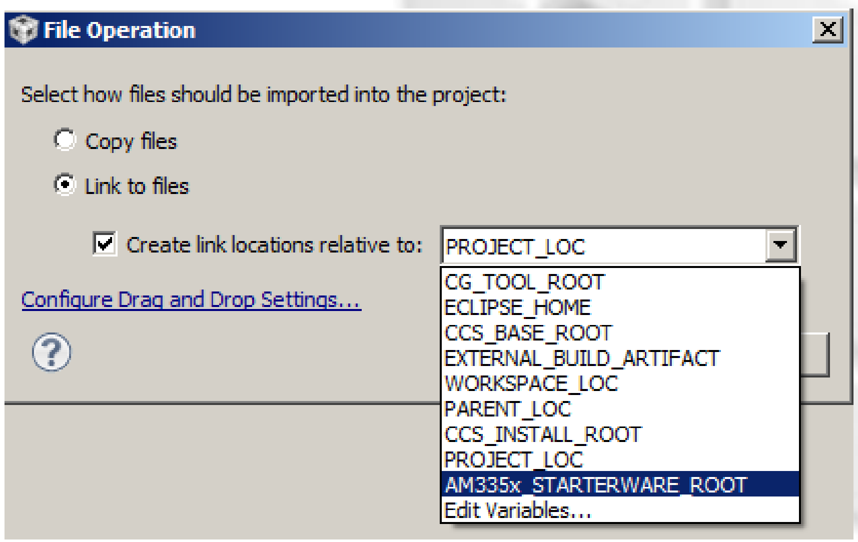
    – <uartEcho.c>


CCS Edit - Code Composer Studio
File Edit View Navigate Project Run Scripts Window Help
Project Explorer
uartEcho
Includes
BeagleBone.ccxml [Active/Default]
uart
OSDisk (C:) ▸ ti ▸ AM335X_StarterWare_02_00_00_05 ▸ examples ▸ beaglebone ▸ uart
Organize Edit Burn New folder
Favorites
Desktop
Downloads
Recent Places
Libraries
Documents
Music
Name
Date modified
Type

Name	Date modified	Type
uartEcho	12/22/2011 10:53 PM	C File

Link Source Files to Project

- **A dialog will appear asking if you wish to Copy or Link the files:**
  - Select *Link to files*
  - Select *Create link locations relative to:*
    - Use the new Linked Resource variable we created (**AM335x_STARTERWARE_ROOT**)
  - Hit *OK*
- **Files will now appear in the Project Explorer with the ‘link’ icon**



File Operation

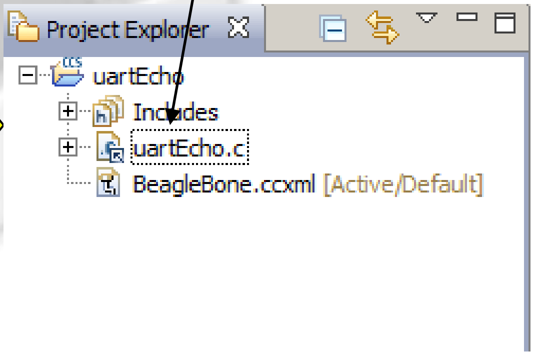
Select how files should be imported into the project:

( ) Copy files
(●) Link to files

[✓] Create link locations relative to: PROJECT_LOC

- CG_TOOL_ROOT
- ECLIPSE_HOME
- CCS_BASE_ROOT
- EXTERNAL_BUILD_ARTIFACT
- WORKSPACE_LOC
- PARENT_LOC
- CCS_INSTALL_ROOT
- PROJECT_LOC
- AM335x_STARTERWARE_ROOT
- Edit Variables...

Configure Drag and Drop Settings...

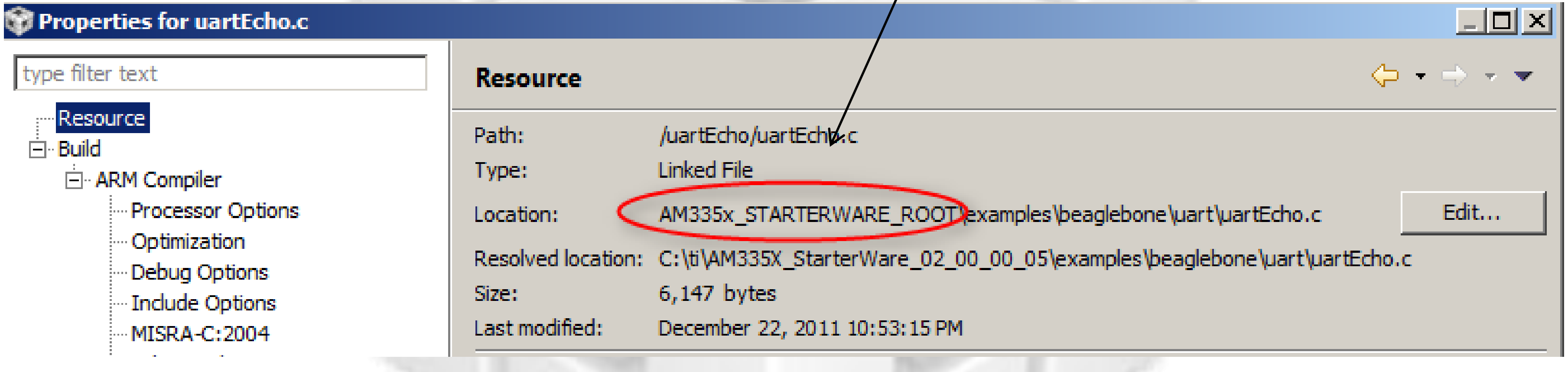


Project Explorer

- uartEcho
  - Includes
  - uartEcho.c
  - BeagleBone.ccxml [Active/Default]

Link Files to Project

- **Right-click on the C source file and check the *Properties***
  - See how the *Location* parameter references the Linked Resource Variable



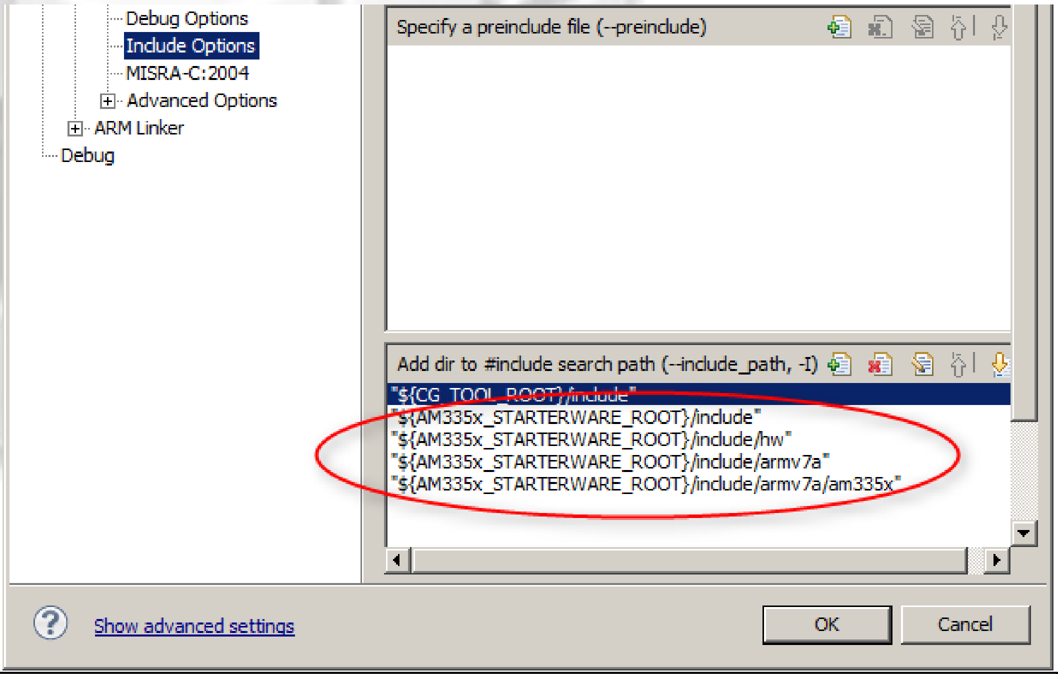
Properties for uartEcho.c

Resource

Path:	/uartEcho/uartEcho.c
Type:	Linked File
Location:	AM335x_STARTERWARE_ROOT\examples\beaglebone\uart\uartEcho.c
Resolved location:	C:\ti\AM335X_StarterWare_02_00_00_05\examples\beaglebone\uart\uartEcho.c
Size:	6,147 bytes
Last modified:	December 22, 2011 10:53:15 PM

Modifying Project Properties

- **Here we are adding paths to include files using the Build Variable**
- **Right-click on the project and select *Properties***
- **In the compiler *Include Options*, add the following entries to the list of include search paths:**
  - ${AM335x_STARTERWARE_ROOT}/include
  - ${AM335x_STARTERWARE_ROOT}/include/hw
  - ${AM335x_STARTERWARE_ROOT}/include/armv7a
  - ${AM335x_STARTERWARE_ROOT}/include/armv7a/am335x
- **Click *OK***
- **'${<BUILD VARIABLE>}' is the syntax to use a Build Variable in the project properties**
- **WARNING: Linked Resource Path Variables are only used when linking source files to a project. They cannot be used for build options. Use Build Variables when modifying build options**



Specify a preinclude file (--preinclude)

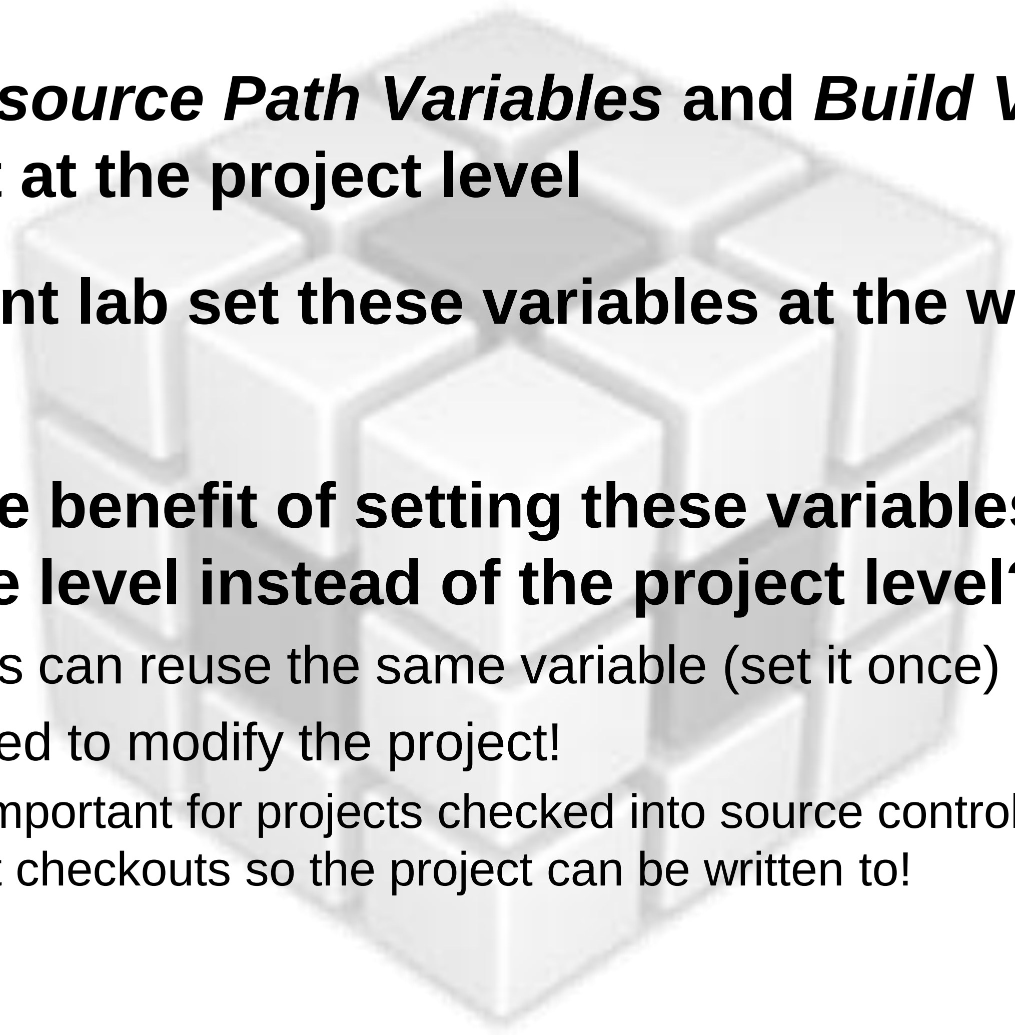
Add dir to #include search path (--include_path, -I)

```
"${CG_TOOL_ROOT}/include"
"${AM335x_STARTERWARE_ROOT}/include"
"${AM335x_STARTERWARE_ROOT}/include/hw"
"${AM335x_STARTERWARE_ROOT}/include/armv7a"
"${AM335x_STARTERWARE_ROOT}/include/armv7a/am335x"
```

CCS APPS

Texas Instruments

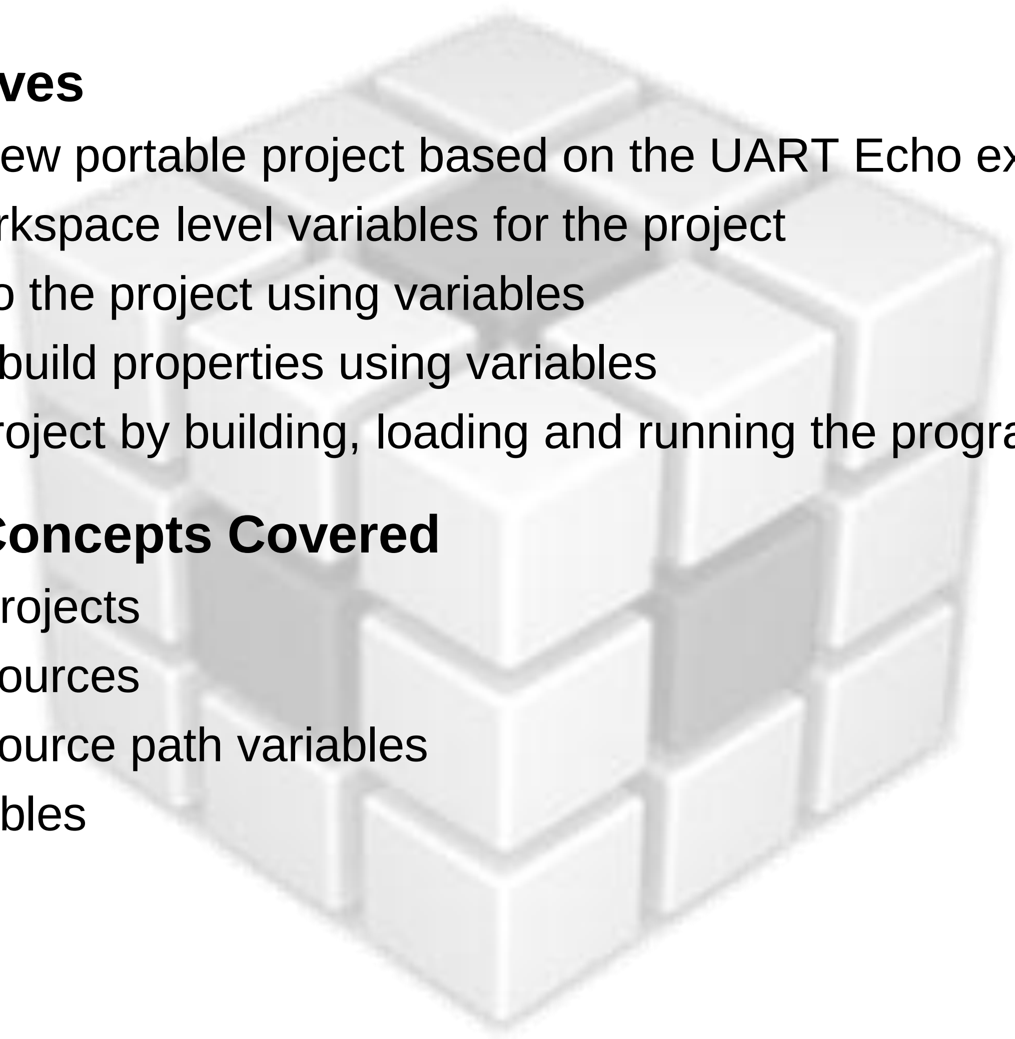
Project vs Workspace Level Variables



- ***Linked Resource Path Variables*** **and** ***Build Variables*** **can be set at the project 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!

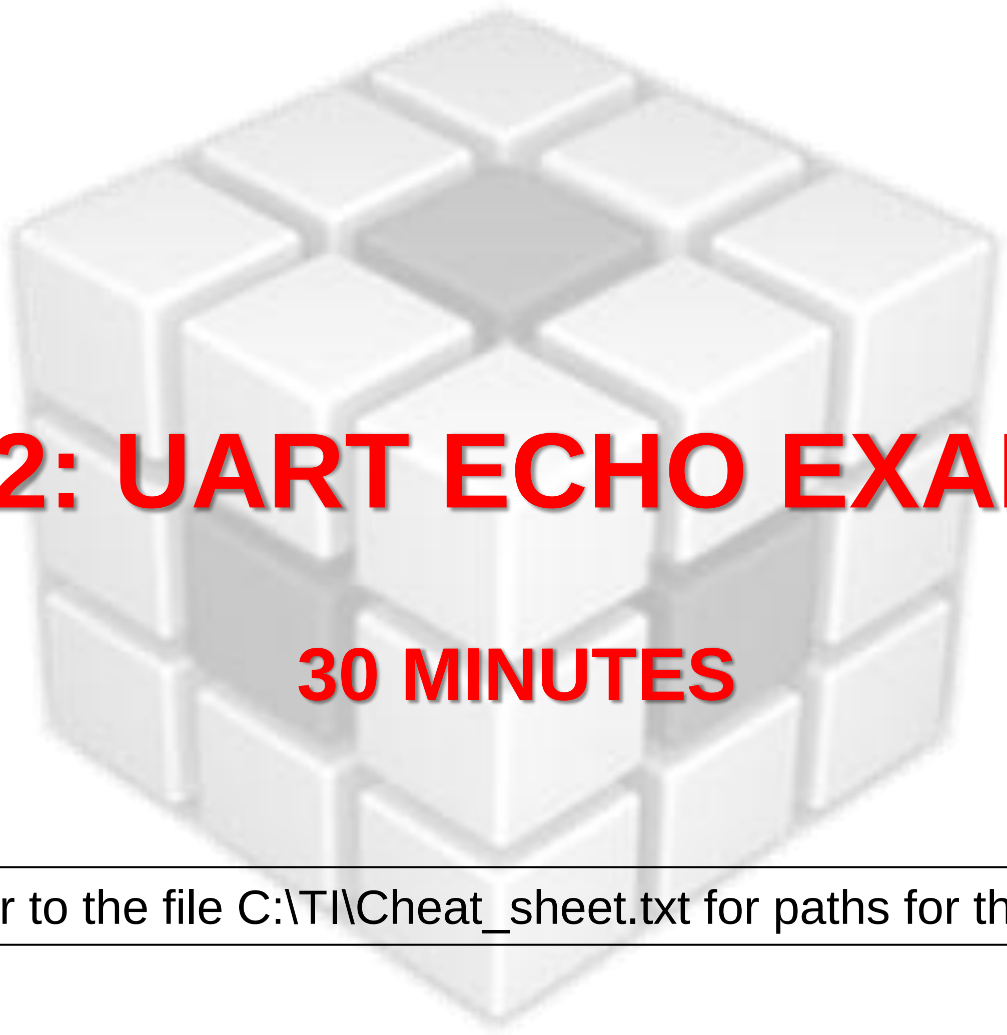
UART ECHO EXAMPLE: PORTABLE PROJECT

UART Echo Example: Exercise Summary



- **Key Objectives**
  - Create a new portable project based on the UART Echo example
  - Create workspace level variables for the project
  - Link files to the project using variables
  - Configure build properties using variables
  - Validate project by building, loading and running the program
- **Tools and Concepts Covered**
  - Portable Projects
  - Linked resources
  - Linked resource path variables
  - Build variables

Questions?



LAB 2: UART ECHO EXAMPLE

30 MINUTES

Please refer to the file C:\TI\Cheat_sheet.txt for paths for the variables